UNIVERSITY OF TARTU
FACULTY OF SCIENCE AND TECHNOLOGY
Institute of Computer Science
Computer Science Curriculum

Anti Ingel

Machine Learning in VEP-based BCI

Masters's Thesis (30 ECTS)

Supervisor: Ilya Kuzovkin, MSc
Co-Supervisor: Raul Vicente, PhD

Tartu 2017

Machine Learning in VEP-based BCI

Abstract

In this thesis, a classification method for SSVEP-based BCI is proposed. The classification method is based on simple comparisons of extracted feature values and thresholds and it involves a way of optimising the thresholds. Optimising the thresholds is formalised as a maximisation task of the information transfer rate of BCI, but instead of using the standard formula for calculating ITR, more general formula is derived. This allows the thresholds to be automatically optimised and avoids calculating incorrect ITR estimate. The proposed method shows good performance in classifying targets of a BCI and achieves ITR as high as 60 bit/min. The proposed method also provides a way to reduce false classifications, which is important in real-world applications. BCIs have high potential to be used in the field of medicine as they provides a way for severely disabled people to control external devices.

Keywords

Brain-computer interface (BCI), electroencephalography (EEG), steady-state visual evoked potential (SSVEP), information transfer rate (ITR), maximisation, threshold optimisation, mutual information

CERCS: P170 Computer science, numerical analysis, systems, control

Masinõpe visuaalselt esilekutsutud potentsiaalidel põhinevas aju-arvuti liideses

Lühikokkuvõte

Antud töös esitatakse visuaalse stiimuliga esilekutsutud potentsiaalidel põhineva aju-arvuti liidese (AAL) jaoks klassifitseerimisreegel, mis põhineb tunnuste ja lävendväärtuste omavahelisel võrdlusel. Klassifitseerimise jaoks optimaalsete lävendväärtuste leidmine formaliseeritakse maksimeerimisülesandena, kus maksimeeritakse AALi informatsiooni edastamise kiirus, mille arvutamiseks tuletatakse eraldi valem, et vältida standardse valemi poolt vajalikke eeldusi. Esitatud reegel näitab AALi klassifitseerimisülesandes häid tulemusi, saavutades informatsiooni edastamise kiiruseks kuni 60 bitti minutis. Samuti võimaldab pakutud reegel vältida vale-ennustusi, mis on oluline AALi kasutamiseks igapäevaelus. AALid omavad suurt potentsiaali medistsiini valdkonnas, kuna võimaldavad raske puudega või halvatud isikutel seadmeid kontrollida.

Võtmesõnad

Elektroensefalograafia EEG, aju-arvuti liides (AAL), visuaalselt esilekutsutud potentsiaal, informatsiooni edastamise kiirus, maksimiseerimine, otsustuslävi, optimiseerimine

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Table of contents

Introduction

Direct communication channel between the brain and an external device that does not need to depend on any of the conventional input channels is called a brain-computer interface (BCI). BCI allows a user to control devices directly with their brain activity. It works by measuring the brain activity of the user and tries to find certain patterns from the recording.

One of the brain activity patterns that has turned out to be quite efficient in implementing BCIs is called steady-state visual evoked potential (SSVEP). It is a brain response to a visual stimulus. Over the recent years, many methods for detecting this pattern from brain activity have been proposed. These methods are based on detecting certain frequencies in the brain signal and different commands that can be sent to the external device correspond to different frequencies.

There are multiple ways to estimate the amount of SSVEP in the brain activity but often in the implemented BCIs only one of them is used. Common way to detect commands with the BCI is to estimate the amount of SSVEP in the brain activity and then use a simple classification rule to make a decision about the users chosen command.

These classification rules can be very simple, for example, value has to be larger than a threshold, or they can be complicated relationships between different frequencies present in the signal that are learned by machine learning. In this work, a simple classification rule that uses thresholds is proposed and a way of optimising these thresholds is derived. Optimising the thresholds will be formalised as a maximisation task of a performance measure of BCIs and a solution for it will be derived starting from basic assumptions.

The first chapter of this thesis gives background information about the biology and describes author's previous work. The previous work includes an implemented BCI which was improved as a practical part of this thesis.

The second chapter describes the feature extraction methods that are used to determine the amount of SSVEPs present in the user's brain signal. These feature extraction methods are all implemented in the author's BCI.

Third chapter gives overview of the related work. In particular, it describes the dataset of SSVEP data that is used to perform offline experiment in this thesis and discusses the articles where the same dataset was used to evaluate a BCI.

Fourth chapter is the contribution of the author. This is the chapter where the classification rule for SSVEP based BCI is presented and a way of optimising the thresholds for the classification rule is derived.

Finally, the fifth chapter gives overview of the obtained results. The performance of the proposed classification method is compared to standard machine learning method and to the articles described in Chapter 3.

1 Background information

1.1 Biological background

The aim of this section is to describe an event-related potential (ERP) of the brain called visual evoked potential (VEP) and to discuss how it can be measured and what important properties it has. How steady-state visual evoked potentials (SSVEPs) are used in SSVEP-based brain-computer interfaces (BCIs) is discussed in Section 1.2 and Chapter 2. This chapter provides a very brief overview of the topics, more detailed description can be found in the author's previous work [14].

1.1.1 Electrical activity in the brain

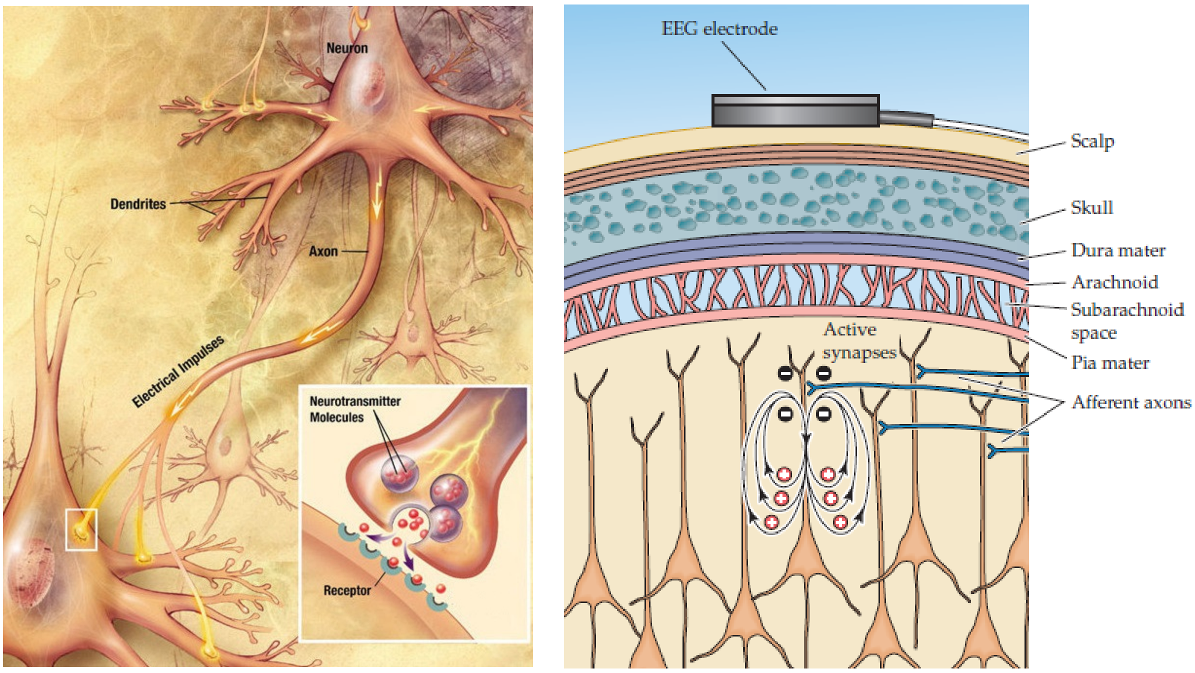
The brain is composed of two classes of cells—nerve cells or *neurons* and glial cells or glia. Neurons are the cells that are more important from the point of view of SSVEP-based BCIs, since they interact with each other by sending electro-chemical signals and these signals can be measured with high temporal resolution using electroencephalography (EEG) devices [6].

Typically a neuron is composed of a cell body, multiple branching nerve endings called *dendrites* that are used to receive signals from other neurons, and one nerve fibre called *axon* that is used to send signals to other neurons. The connections between axons and dendrites through which a signal is sent from one neuron to another are called *synapses*. The structure of a neuron can be seen in Figure 1.1a.

Neurons send signals by transporting ions across the cell membrane. The act of sending a signal is called *action potential* and it works by moving higher concentration of positive ions along an axon to a synapse through which the signal is sent to another neuron. The neuron that receives the signal is called *postsynaptic neuron*. The signal can make a postsynaptic neuron more prone to send a signal itself or it might inhibit it from sending a signal.

Based on an article by Buzsaki *et al.* [4], the signal causes ions to flow into the postsynaptic neuron. This in turn causes a balancing flow of ions from inside the neuron to flow out of it to achieve electroneutrality [4]. This produces different electric potentials near the locations where ions enter the cell and where ions exit the cell. These different electric potentials form a *current dipole*. See Figure 1.1b for illustration.

Although action potentials generate stronger currents than current dipoles, current dipoles are more likely to have synchronous activity which is required to produce large enough electric potential to be able to measure it from the scalp [4]. For the current dipoles to add up and produce large electric potential, the neurons have to be oriented the same way, be perpendicular to the surface of the brain as shown in Figure 1.1b and approximately 108 neurons have to have synchronous electrical activity to create a measurable field [22].



(a) Neurons and a chemical synapse [28, p. 17].

(b) Current dipole generated by a neuron [24, p. 669].

Figure 1.1: Neurons and a current dipole.

1.1.2 Electroencephalography

Neuroimaging method called EEG can be used to measure the electric potentials generated by current dipoles. EEG device consists of electrodes connected to a voltmeter, which measures the electric potential difference between two electrodes. Thus the EEG measures the difference between the electric potential of a fixed reference electrode and other electrodes. Commonly the electrodes are placed on the scalp according to 10/20, 10/10 or 10/5 electrode placement system that uses in most cases the first letter of the name of the brain lobe above which an electrode is located and a number to code the location of the electrode on the scalp [16].

In this thesis, a consumer-grade EEG device called Emotiv EPOC was used. The Emotiv EPOV device has 14 electrodes which are located at AF3, AF4, F3, F4, F7, F8, FC5, FC6, P7, P8, T7, T8, O1, O2 and two reference electrodes that are located either at P3 and P4 or behind the ears. The locations on the scalp are shown in Figure 1.2. Emotiv EPOC has a *sampling rate* of 128 Hz, which means that 128 voltage measurements are made in a second for every electrode.

The reason for using consumer-grade device was that these devices are relatively easy to use and affordable for a person in need to buy. Although research has shown that Emotiv EPOC performs significantly worse than a medical-grade devices [8], the performance of Emotiv EPOC is good enough to implement an SSVEP-based BCIs [21, 38, 19, 13].

[1] http://emotiv.wikia.com/wiki/Emotiv_EPOC

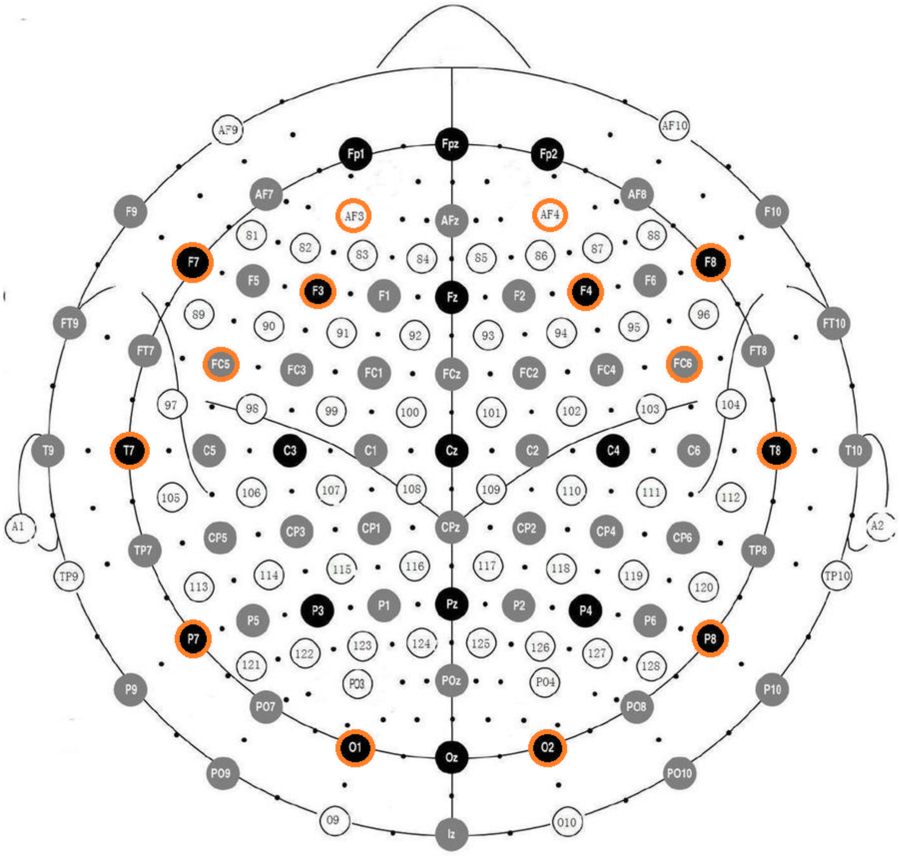


Figure 1.2: Electrode locations used by Emotiv EPOC[1]. Used locations are marked with orange circles.

1.1.3 Steady-state visual evoked potential

This section gives an overview of the steady-state visual evoked potential (SSVEP) which is a brain's response to a specific visual stimulus. In general, brain potential evoked by some event is called an event-related potential (ERP). ERPs are important in implementing BCIs, because ERPs are measurable with EEG devices using averaging techniques [18] and specific enough to be controlled by the user of the BCI.

VEP is a type of ERP, see Figure 1.3 for an example. VEPs are elicited by a visual stimuli, which in case of SSVEP-based BCIs is often a *flickering* square on a computer screen [37]. The visual stimuli of SSVEP-based BCIs are called *targets*. Since the *visual processing centre* is located posteriorly in the brain, when measuring VEPs, the EEG electrodes have to be placed on the back of the head. For the Emotiv EPOC device, the electrodes closest to the visual processing centre are O1 and O2, see Figure 1.2.

One blink of a visual stimulus produces a response as shown in the 2 Hz subplot in Figure 1.3. But as the speed of blinking increases, the consecutive VEPs merge together and form one continuous response called SSVEP [34]. If the stimulus is just a flickering square, then the elicited SSVEP's *fundamental frequency* is equal to the flickering frequency. In addition to flickering frequency of the target, also its *harmonics* might be elicited [10]. Detecting SSVEPs is preferred over detecting VEPs, because SSVEP is continuous and thus should always be present in the EEG signal.

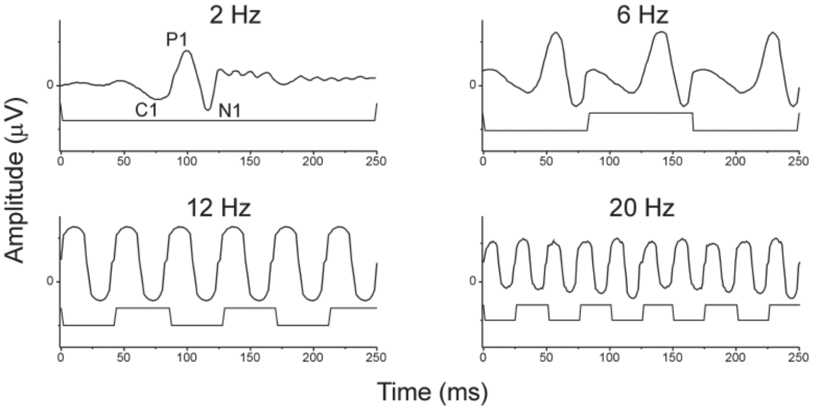


Figure 1.3: VEPs elicited by slow frequency stimulus and SSVEPs elicited by high frequency stimulus [34, p. 259]. The square waves denote the state switches or the flickering of the target and the smooth curve denotes the brain response.

As shown in Figure 1.3, targets with different flickering frequency produce responses with different frequencies. Another important property is that the stimulation of *central visual field* produces larger VEPs than stimulation of *peripheral vision* [11]. Using these two properties, it is possible to design visual stimuli that can be used in implementing a SSVEP-based BCI.

Common way of eliciting SSVEPs in a user of a BCI is displaying several targets on the computer screen that each flicker with different frequency and each of them corresponds to a different command that the BCI can process. Then, if the user is focusing on one of the targets, it elicits a brain response with known fundamental frequency and the rest of the targets do not elicit as large response. This response can be detected by recording the brain activity with EEG device and analysing the recording with feature extraction methods. In practice, detecting SSVEPs can be challenging due to the noisiness of the EEG data and differences between people.

1.1.4 Conclusion

When a user of a SSVEP-based BCI is presented with a visual stimulus that flickers with a fixed frequency, the user sees this flickering and his brain processes the signal in the visual processing centre. This processing of the information produces synchronised activity of neurons in the visual processing centre. This synchronised activity generates current dipoles that add up together so that it is possible to measure this activity from the scalp of the user using an EEG device. An important property of the brain's response to the stimulus is that it has the same fundamental frequency as the flickering frequency of the stimulus. Feature extraction methods described in Chapter 2 use this property to detect the elicited SSVEPs and the extracted features are used to predict the command chosen by the user.

1.2 Brain-computer interface

This chapter gives an overview of the BCI implemented as a practical part of the author's previous work [14]. This BCI was improved as a practical part of this thesis. Three of the most important improvements are the following. First, three additional *feature extraction* methods were added to the BCI that are described in sections 2.3–2.5. Second, previously the BCI did not use machine learning to classify the targets but as a practical part of this thesis, machine learning and some other improvements were added to the BCI. Third, the classification method proposed in Chapter 4 of this work was implemented.

1.2.1 Visual stimuli

The BCI used in this thesis uses a computer screen to present visual stimuli to the user, see Figure 1.4 for an example. Since a computer screen has a fixed *refresh rate*, there is a limit on what flickering frequencies can be used. Refresh rate shows how many frames per second the computer screen can display. For example, if the refresh rate is 60 Hz, then the maximum flickering frequency that can be used is 30 Hz. Another complication is that with the simplest method, only these flickering frequencies that exactly divide the refresh rate can be used. Other frequencies can still be used and detected, but designing the blinking sequence is more complicated [21].

As already mentioned in Section 1.1.3, visual stimulation with fixed frequency elicits a response in the brain that has the same fundamental frequency as the flickering, but in addition to this frequency, also its harmonics might be present in the response [10]. Research has also shown that these frequencies that are present in the *flickering waveform* are more likely to be also present in the SSVEP [26]. This might pose problems when target frequencies that are integer multiples of each other are used.

A user can send commands through the BCI by looking at one of the targets presented to him on a computer screen. Each target corresponds to a command that the user can choose and by focusing his gaze on a target, SSVEP that can be detected by the BCI is elicited. This way, the user can control devices by just focusing his gaze to different regions on a computer screen.

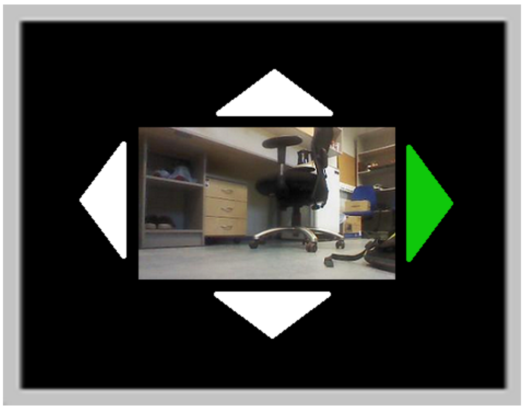


Figure 1.4: Example of visual stimuli on a computer screen [14]. The picture in the middle denotes video stream from the robot that the BCI is used to control. Arrows denote possible commands and the green arrow denotes user's current choice.

1.2.2 Signal pipeline

The signal pipeline implemented for the BCI as a part of the author's previous work [14] includes *detrending*, filtering, *windowing*, feature extraction, *interpolation* and finally classification. Previously implemented feature extraction methods were power spectral density analysis (PSDA) and canonical correlation analysis (CCA), which along with added feature extraction methods are discussed in Chapter 2.

Detrending is removing a constant or linear *trend* from the signal as a result of which the *mean* of the signal will be zero. This is important for the feature extraction methods, because some of these, like CCA, make the assumption that the input signal has zero mean. Detrending is also a necessary step before windowing the signal, which can be used with feature extraction methods based on *Fourier transform*. Detrending can be performed on segments of the signal separately instead of detrending the whole signal at once.

Windowing is used to smooth the ends of a signal. This is useful when using feature extraction methods based on Fourier transform, because Fourier transform assumes that the signal is periodic. If the signal under consideration is of finite length, it is thought of as repeating itself infinitely many times. This concatenation might, however, introduce steep jumps to the signal if the difference between the first value and the last value of the signal is too large. Windowing is essentially an element-wise multiplication of the signal with a function that has largest values in the middle of the range and near-zero values near the ends.

Filtering is used for removing noise from the signal. It is used to remove certain frequencies from the signal. Filters can be designed for example to remove all frequencies that are smaller than a given frequency, all frequencies that are larger than given frequency or all frequencies that are between two given frequencies. Due to the noisiness of the EEG recording, filtering can be a useful step before performing feature extraction to decrease the negative effect of noise on the feature extraction and classification.

Feature extraction methods are used to extract information about the frequencies that should be present in the EEG recording when the user is looking at the visual stimuli. The frequencies elicited by visual stimuli and other properties of SSVEP were discussed in Section 1.1.3. Usually feature extraction methods give feature values for each target frequency for each time step. Some methods, like PSDA, give separate results for the target frequency and its harmonics. These features can be used for classifying, which target is the user looking at.

Interpolation can be used if the feature extraction method gives results for other frequencies than the target frequencies of the BCI. Then the known values for other frequencies can be used to estimate values for target frequencies.

The classification method implemented in the author's previous work [14] does not use machine learning. It uses manually chosen weights and thresholds. Weights show how much each feature affects the final result and threshold is used for filtering out as much false positive results as possible. This classification method uses the assumption that higher feature values for a target mean that the corresponding target is more likely the one chosen by the user. But this might not always be the case. Machine learning can be used to find which features are better at classifying the targets and more complicated relationships between the feature values and targets can be found with machine learning.

2 Feature extraction methods for SSVEP-based BCI

This chapter gives an overview of the feature extraction methods used in the implemented BCI. The feature extraction methods are used to extract useful information from the EEG signal that can be used to classify the commands. The feature extraction methods called power spectral density analysis (PSDA) and canonical correlation analysis (CCA) were already discussed in author's previous work [14]. In addition to those method the improved version of the BCI has minimum energy combination (MEC), likelihood ratio test (LRT) and continuous wavelet transform (CWT) feature extraction methods.

2.1 Power spectral density analysis

Power spectral density analysis (PSDA) method was used already in 2002 by Cheng *et al.* [5]. This section provides insights to the theory behind the method and describes how it can be used to implement a SSVEP-based BCI. Parts of this section are taken from a report written by the author for a project of course Research Seminar in Data Mining in University of Tartu.

PSDA is a common method used to determine the amplitude of different frequencies present in a signal. It is based on discrete Fourier transform which decomposes a signal into a set of sine and cosine waves or equivalently to a set of complex exponentials. If $f(t)$ is the actual signal and $f_N(t)$, $t \in \{0, 1, \dots, N-1\}$ represents the recording of the signal, then

$$f_N(t) = \sum_{k=-N}^{N} c_k e^{2\pi i k t} \tag{2.1}$$

which approximates the actual signal $f(t)$ and the approximation improves as $N \to \infty$.

Discrete Fourier transform finds the coefficients c_k of the linear combination (2.1) by the following formula

$$c_k = \sum_{t=0}^{N-1} f_N(t) e^{-2\pi i k t/N}$$

which can be represented as a matrix multiplication

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W & W^2 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & \dots & W^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W^{N-1} & W^{2(N-1)} & \dots & W^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} f(0) \\ f(1) \\ f(2) \\ \vdots \\ f(N-1) \end{pmatrix}$$

where $W = e^{-2\pi i/N}$. An algorithm for performing this multiplication with $O(N \log N)$ complexity is called fast Fourier transform (FFT).

To be able to compare the amount of different frequencies present in the signal $f(t)$, one can find the amplitudes of the complex exponentials by taking the magnitude of the complex number c_k. Thus $|c_k|$ is the amplitude of the frequency

$$k \cdot \frac{f_s}{N} \tag{2.2}$$

where $k \in \{0, 1, \ldots, \lfloor N-1 \rfloor /2\}$ and f_s is the sampling rate in Hz. The frequencies obtained by formula (2.2) are called *frequency bins*. By squaring the magnitudes one gets a *periodogram*, an estimation for the power spectral density (PSD) of the signal $f(t)$, and hence the name power spectral density analysis.

The final aspect to consider in this method is that the targets with higher flickering frequency produce SSVEPs with smaller amplitude and that there is a lot of noise present in the EEG recording, especially when using a consumer-grade device. Thus it might be more beneficial to compare signal-to-noise ratios (SNRs) instead of the amplitudes or the powers themselves, for example by using formula [3]

$$SNR(f_t) = \frac{2P(f_t)}{P(f_{t-1}) + P(f_{t+1})} \tag{2.3}$$

or more generally [33]

$$SNR(f_t) = \frac{nP(f_t)}{\sum_{i=1}^{n/2} \left(P(f_{t-i}) + P(f_{t+i})\right)} \tag{2.4}$$

where P is the periodogram, $f_1, \ldots, f_t, \ldots, f_N$ represent the frequency bins in increasing order, f_t is the target flickering frequency and n is the number of adjacent frequency bins used in the calculation of SNR. Either SNRs or powers for each target and its harmonic frequencies at each time step can be used as features for classification.

More recent feature extraction method called minimum energy combination (MEC) uses spatial filtering to try to overcome some of the weaknesses of PSDA method [9, 30].

2.2 Canonical correlation analysis

Canonical correlation analysis (CCA) feature extraction method has been one of the most applied methods in SSVEP-based BCIs and it has shown very good results even with a consumer-grade EEG device [13, 21]. The method was introduced by Lin *et al.* [20] and this section mainly follows their description of the method with some additional insights to the theory behind the method. Parts of this section are taken from a report written by the author for a project of course Research Seminar in Data Mining in University of Tartu.

CCA provides a way to calculate the correlation between two sets of random variables. It was introduced by Harold Hotelling in 1936 [12].

In SSVEP-based BCI setting, one set of random variables is the multichannel EEG recording and the other is a set of reference signals [20]. Each target has its own set of reference signals based on the flickering frequency of the target. The reference signals are sine and cosine waves with the same frequency as the target flickering frequency and also some of its harmonics. Usually two or three harmonics are used. For a target m with frequency f_m the set of reference signals is

$$Y_m = \begin{pmatrix} \sin(2\pi \cdot f_m \cdot t) \\ \cos(2\pi \cdot f_m \cdot t) \\ \vdots \\ \sin(2\pi \cdot N_h \cdot f_m \cdot t) \\ \cos(2\pi \cdot N_h \cdot f_m \cdot t) \end{pmatrix}, \quad t = \frac{1}{f_s}, \frac{2}{f_s}, \ldots, \frac{N}{f_s} \tag{2.5}$$

where f_s is the sampling rate, N_h is the number of harmonics used and N is the length of the recorded signal [20].

Similarly the multichannel EEG recording can be organised a into matrix X where each row contains signal from one channel.

CCA finds two new sets of features whereas the features of one set are linear combinations of the rows of X and the features of the other set are linear combinations of the rows of Y_m. Thus given a matrices of weights W_X and W_{Y_m} the new features are $W_X X$ and $W_{Y_m} Y_m$.

The new features are chosen so that the corresponding rows of $W_X X$ and $W_{Y_m} Y_m$ are maximally correlated, meaning that the first row of $W_X X$ is maximally correlated with the first row of $W_{Y_m} Y_m$ and similarly with other rows. The first rows are called the first pair of canonical variates. The subsequent rows have additional constraint that they have to be uncorrelated with the previous canonical variates. The rows of $W_X X$ themselves have maximal possible variance and similarly the rows of $W_{Y_m} Y_m$.

The problem of finding the first pair of canonical variates can be formulated as a maximisation task

$$\max_{\vec{a},\vec{b}} \mathbf{Cor}(\vec{a}^T X, \vec{b}^T Y_m) = \frac{\vec{a}^T X Y_m^T \vec{b}}{\sqrt{\vec{a}^T X X^T \vec{a} \vec{b}^T Y_m Y_m^T \vec{b}}} \tag{2.6}$$

where $\vec{a}$ is a column vector representing the first row of W_X and $\vec{b}$ is a column vector representing the first row of W_{Y_m}.

The solution to this problem can be obtained by finding the singular value decomposition (SVD) of the matrix

$$K = \Sigma_{XX}^{-\frac{1}{2}} \Sigma_{XY} \Sigma_{YY}^{-\frac{1}{2}}$$

where

$$\Sigma_{XY} = \mathbf{Cov}(X, Y) = XY^T.$$

SVD gives matrices U with columns $\vec{u}_i$, V with columns $\vec{v}_i$ where $i \in \{1, \ldots, \text{rank}(K)\}$ and D such that

$$K = UDV^T$$

and the vectors $\vec{a}$ and $\vec{b}$ for the problem (2.6) can be calculated as

$$\vec{a} = \Sigma_{XX}^{-\frac{1}{2}} \vec{u}_1$$
$$\vec{b} = \Sigma_{YY}^{-\frac{1}{2}} \vec{v}_1$$

which can be used to calculate the first canonical variates. By replacing $\vec{u}_1$ and $\vec{v}_1$ with other columns of matrices U and V, other canonical variates can be calculated. To the best of the author's knowledge, however, only the first canonical variates have been used in SSVEP-based BCI feature extraction methods so far.

To identify the target that the user is looking at, one can compare the correlations obtained by formula (2.6) for different targets and the target whose set of reference signals has the highest canonical correlation with corresponding $\vec{a}^T X$ can be considered to be the target the user is looking at. Therefore, the canonical correlations between the EEG and each target's reference signals at each time step can be used as features for classification.

The advantage of this method over PSDA is that it is inherently multidimensional which means that CCA method can analyse signals from multiple EEG channels at once. In case of PSDA the features have to be extracted separately for each channel.

2.3 Likelihood ratio test

Likelihood ratio test (LRT) method was introduced by Zhang *et al.* [35] and this section mainly follows his description of the method with some additional insights to the theory behind the method.

LRT feature extraction method is based on a statistical test. It compares a dataset obtained by sampling against a synthetic data from idealised model. In the case of SSVEP-based BCI, the idealised data is the same that was used in CCA method as reference signals (2.5) and the data obtained by sampling is the EEG signal. The test is used separately for each target as each target has separate set of reference signals.

Since the goal of a feature extraction methods for SSVEP-based BCIs is to detect SSVEPs in the EEG signal, the statistical test in LRT method is used to find how unlikely it is that the recorded EEG data is uncorrelated with the reference signal. Therefore the null hypothesis for the test is that the signals are uncorrelated and the alternative hypothesis is that they are correlated.

Assumptions on the distribution are that both sets of signals are normally distributed. Symbolically, by representing the multichannel EEG signal as a random vector $\vec{X}$ and the reference signals as a random vector $\vec{Y}_m$ the assumption is that [35]

$$\vec{X} \sim N(\mu_X, \Sigma_{XX}) \qquad \vec{Y}_m \sim N(\mu_{Y_m}, \Sigma_{Y_mY_m})$$

where Σ_{XX} and $\Sigma_{Y_mY_m}$ denote the covariance matrices of $\vec{X}$ and $\vec{Y}_m$.

For convenience, the two random vectors will be combined into one vector [35]

$$Z = \begin{pmatrix} \vec{X} \\ \vec{Y}_m \end{pmatrix}, \qquad \vec{\mu} = \begin{pmatrix} \vec{\mu}_X \\ \vec{\mu}_{Y_m} \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} \Sigma_{XX} & \Sigma_{XY_m} \\ \Sigma_{Y_mX} & \Sigma_{Y_mY_m} \end{pmatrix}.$$

Using this notation, the null hypothesis H_0 is that $\Sigma_{XY_m} = \mathbf{0}$ and the alternative hypothesis H_1 that $\Sigma_{XY_m} \neq \mathbf{0}$ where $\mathbf{0}$ denotes a null matrix [35]. A null matrix is a matrix consisting of only zeroes.

Now given a set of observations $\vec{z}_1, \dots, \vec{z}_n$, the LRT statistic can be calculated as [35]

$$\lambda = \frac{\max_{H_0} L\left(\vec{\mu}, \Sigma \mid \vec{z}_1, \dots, \vec{z}_n\right)}{\max_{H_0 \cup H_1} L\left(\vec{\mu}, \Sigma \mid \vec{z}_1, \dots, \vec{z}_n\right)} \tag{2.7}$$

where L denotes the likelihood and it can be calculated using probability density function f which for multidimensional normal distribution is as follows

$$L\left(\vec{\mu}, \Sigma \mid \vec{z}_1, \dots, \vec{z}_n\right) = f\left(\vec{z}_1, \dots, \vec{z}_n \mid \vec{\mu}, \Sigma\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^m |\Sigma|}} \exp\left(-\frac{(\vec{z}_i - \vec{\mu})^T \Sigma^{-1} (\vec{z}_i - \vec{\mu})}{2}\right).$$

The maximisations in (2.7) mean that the parameters $\vec{\mu}$ and Σ are sought so that the corresponding quantities are maximised while corresponding conditions are satisfied—the condition is $\Sigma_{XY_m} = 0$ for the numerator and $\Sigma_{XY_m} = 0 \vee \Sigma_{XY_m} \neq 0$ for the denominator.

This means that the maximum likelihood estimates $\hat{\vec{\mu}}$ and $\hat{\Sigma}$ are sought for the parameters $\vec{\mu}$ and Σ. For given observations $\vec{z}_1, \dots, \vec{z}_n$ these estimates can be calculated as

$$\hat{\vec{\mu}} = \frac{1}{n} \sum_{i=1}^{n} \vec{z}_i, \qquad \hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (\vec{z}_i - \hat{\vec{\mu}})(\vec{z}_i - \hat{\vec{\mu}})^T \tag{2.8}$$

These estimates and matrix properties can be used to rewrite the equation (2.7) as [35]

$$\lambda = \frac{|\hat{\Sigma}|^{n/2}}{|\hat{\Sigma}_{XX}|^{n/2} \cdot |\hat{\Sigma}_{Y_m Y_m}|^{n/2}}.$$

Since λ shows how different $\vec{X}$ and $\vec{Y}_m$ are, $1 - \lambda$ shows how similar these are [35]. To take into account the dimensions, p-th root of λ can be taken before subtraction [35]

$$C_m = 1 - \left(\frac{|\hat{\Sigma}|^{n/2}}{|\hat{\Sigma}_{XX}|^{n/2} \cdot |\hat{\Sigma}_{Y_m Y_m}|^{n/2}}\right)^{1/p}$$

where p is the dimension of Y_m. If $\vec{X}$ and $\vec{Y}_m$ are identical, then $C_m = 1$; if $\vec{X}$ and $\vec{Y}_m$ are independent, then $C_m = 0$ [35]. C_m can be calculated for each target at each time step and used as features in classification.

This method is similar to the CCA feature extraction method since both of these methods try to find similarities between the EEG signal and the same set of reference signals.

2.4 Minimum energy combination

Minimum energy combination (MEC) feature extraction method was introduced by Friman *et al.* [9] and it has been shown to achieve very high information transfer rate (ITR) [30]. This section follows the description of the method given by Volosyak [30].

MEC method makes the assumption that the EEG data $y_i(t)$ from channel $i \in \{1, 2, \dots, N_y\}$ can be represented as a sum of sine and cosine waves and noise signal $E_{i,t}$

$$y_i(t) = \sum_{k=1}^{N_h} \left(a_{i,k} \sin 2\pi k f t + b_{i,k} \cos \pi k f t\right) + E_{i,t} \tag{2.9}$$

where N_h is the number of harmonics used [30]. By inspecting the equation (2.9), it can be seen that it can be represented in a matrix form

$$Y = XG + E$$

where X is the matrix defined as the matrix of reference signals in (2.5), G is a matrix containing all the amplitudes $a_{i,k}$ and $b_{i,k}$ of the sine and cosine waves in X, E is the noise and Y contains the EEG signal for all the channels.

Next, the MEC method finds *linear combinations* $s_j = Yw_j$ with weights w_j of the signals y_i such that w_j minimises

$$\min_{w_j} w_j^T \tilde{Y}^T \tilde{Y} w_j \tag{2.10}$$

where

$$\tilde{Y} = Y - X(X^T X)^{-1} X^T Y$$

is the orthogonal projection of the EEG data onto the sine and cosine waves [30]. By definition, $\tilde{Y}$ should not contain SSVEP activity. Therefore, by minimising (2.10), the noise and nuisance component of the signal are minimised, or in other words, the energy of $\tilde{Y}$ is minimised and hence the name minimum energy combination.

The problem (2.10) is solved by finding the eigenvector v_1 corresponding to the smallest eigenvalue λ_1 of the matrix $\tilde{Y}^T \tilde{Y}$. Additional uncorrelated channels can be added by choosing the next eigenvectors $v_2, v_3, \dots$ corresponding to next smallest eigenvalues $\lambda_2, \lambda_3, \dots$.

The weight vectors w_j can therefore be organised into a matrix

$$W = \left(\frac{v_1}{\sqrt{\lambda_1}}, \dots, \frac{v_{N_s}}{\sqrt{\lambda_{N_s}}} \right)$$

where N_s can be chosen as the smallest value for which

$$\frac{\sum_{i=1}^{N_s} \lambda_i}{\sum_{i=1}^{N_y} \lambda_i} > 0.1$$

where N_y is the number of EEG channels [30].

Therefore the new signals s_j, $j \in \{1, 2, \dots, N_s\}$ can be also organised into matrix

$$S = YW.$$

The features used for classification can be calculated for a target by using its corresponding reference signals X_k [30]

$$\hat{P} = \frac{1}{N_s N_h} \sum_{l=1}^{N_s} \sum_{k=1}^{N_h} s_l^T X_k X_k^T s_l.$$

Features can be normalised over all targets

$$p_i = \frac{\hat{P}_i}{\sum_{j=1}^{N_f} \hat{P}_j}$$

where N_f is the number of targets. Volosyak [30] used instead a softmax function to enhance the gap between the features

$$p_i' = \frac{e^{\alpha p_i}}{\sum_{j=1}^{N_f} e^{\alpha p_i}}$$

where α was set to 0.25. Therefore, for each target, $\hat{P}$ can be calculated and transformed to p_i or p_i' which can be used as features for classification.

2.5 Continuous wavelet transform

Zhang *et al.* [36] proposed a BCI system based on continuous wavelet transform (CWT). This method is similar to PSDA method, but instead of Fourier transform, CWT is used.

While PSDA method decomposes the EEG signal into complex exponentials, the CWT method decomposes the signal into time-shifted and scaled *wavelets*. The shifted and scaled wavelets are generated from a chosen *mother wavelet*. Fourier transform can be viewed as a special case of CWT with mother wavelet ψ set to

$$\phi(t) = e^{-2\pi i t}.$$

In the work by Zhang *et al.* [36] different mother wavelets were compared and complex Morlet wavelet was chosen as mother wavelet

$$\phi(t) = \frac{1}{\sqrt{\pi f_b}} e^{2i\pi f_c x} e^{-\frac{x^2}{f_b}}$$

where f_b is the bandwidth parameter and f_c the wavelet centre frequency.

The formula of CWT for signal $x(t)$ and mother wavelet $\phi(t)$ is

$$W(a,b) = \langle x(t), \phi_{a,b}(t) \rangle = \int_{-\infty}^{\infty} x(t)\phi_{a,b}^*(t) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\phi^* \left(\frac{t-b}{a} \right) \tag{2.11}$$

where a is the scale factor and b the shift factor [36]. The discrete form of the transpose is

$$W(a,b) = \frac{\sqrt{a}}{N} \sum_{k=0}^{N-1} X(k)\Phi(ak)e^{\frac{i2\pi bk}{N}}$$

where $b \in \{0, 1, \ldots, N-1\}$, N is the number of samples of EEG data and $X(k)$ and $\Phi(k)$ denote the discrete Fourier transform of the sampled signal $x(n)$ and discrete wavelet $\phi(n)$ [36].

To interpret the wavelet coefficients, scale factor a can to be converted into frequency f

$$f = \frac{f_s \cdot f_0}{a}$$

where f_s is the sampling frequency and f_0 the centre frequency of the wavelet in Hz [36].

The average energy of wavelet coefficients $W(f,b)$ for every EEG channel can be computed as [36]

$$P_f = \frac{1}{N}\sum_{i=1}^{N}|W(f,i)|^2.$$

Therefore P_f for each EEG channel for each target frequency f can be calculated and used as features for classification in SSVEP-based BCI.

The advantage of this method over PSDA method is that PSDA method makes the assumption that the signal is periodic—it oscillates the same way from $t = -\infty$ to $t = \infty$ during every period. In CWT such assumption is not needed as the amplitude of the wavelets decreases over time and reaches zero.

3 Related work

This chapter describes a publicly available SSVEP dataset [2] that was used to perform offline experiments in this work and gives detailed overview of 6 articles where the same dataset was used to evaluate performances of BCIs. In Chapter 5, the results of these articles are compared to the results of this work. This chapter also describes the performance measures used to evaluate BCIs.

3.1 SSVEP dataset for offline experiments

The offline experiments of this work were mostly performed on the publicly available EEG dataset by Bakardjian *et al.* [2]. The dataset contains EEG recordings of four subjects. The flickering frequencies of the targets were 8 Hz, 14 Hz and 28 Hz and the visual stimuli were displayed on a computer monitor with 170 Hz refresh rate approximately 90 cm away from the subject's eyes. With each subject, five trials of recording the EEG for each flickering frequency were performed. Each trial consists of about 25 seconds of EEG data and 15 seconds of that is with visual stimulation of the target. EEG data was recorded using BIOSEMI EEG system with 128 channels and 256 Hz sampling rate. To make the results comparable to Emotiv EPOC, only two of the 128 channels were used—O1 and O2.

The advantages of using a publicly available dataset is that the obtained results can be compared to other results published in the literature. Overview of other works that use the same dataset and implement a classification algorithm for BCI can be seen in Section 3.3.

3.2 Performance measures

There are multiple ways to measure the performance of a BCI. One way to evaluate a BCI is to calculate its classification accuracy, that is the ratio of correctly classified targets to the number of classifications made.

Accuracy, however, does not reflect how long it takes for the BCI to make the decision. Therefore, mean detection time (MDT) is also used that shows how much time it takes on average to classify a target. For some BCIs, the detection time is fixed while for others it varies, allowing the BCI to keep collecting additional data, if it cannot make confident enough prediction with the data at hand. Avoiding making uncertain classifications is beneficial, because having a BCI make as few false positive predictions as possible is a desirable property in a real-world application.

A performance measure that combines the accuracy P and the number of targets N is called information transfer rate (ITR) and the formula for calculating it in units of bits per classification of command is [31]

$$\mathrm{ITR}_c = \log_2 N + P \log_2 P + (1-P)\log_2\left(\frac{1-P}{N-1}\right). \tag{3.1}$$

ITR_c shows how many bits of information is transferred through the BCI with one prediction. The ITR can also be calculated in units of bits/min that is commonly used to measure the performances of BCIs as it also takes into account the MDT

$$\mathrm{ITR} = \mathrm{ITR}_c \cdot \frac{60}{\mathrm{MDT}} \tag{3.2}$$

and it shows the amount of information transferred in one minute. The assumptions that should be satisfied in order to use given formulas to calculate ITR are: all predictions are equally likely, if choosing a wrong target, all wrong targets are equally likely to be chosen and the accuracy of successfully classifying a target has to be the same for every target [31].

For evaluating a specific classifier, it can be useful to use performance measures like true positive rate (TPR) or recall, positive predictivity value (PPV) or precision and false positive rate (FPR). The formulas for calculating these measures are as follows

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \qquad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}, \qquad \mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}},$$

where TP denotes the number of true positives, TN true negatives, FP false positives and FN false negatives. With the same notation, accuracy can be calculated as

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}.$$

TPR, PPV and FPR can be useful when choosing a threshold value for a feature of a binary classifier. FPR and TPR are used to plot receiver operating characteristic (ROC) curves. In ROC curves, for each threshold value FPR and TPR of the classifier are calculated and the result can be plotted as TPR versus FPR for visual evaluation of the classifier. Area under the generated curve can be used as performance measure and suitable threshold can be chosen from the curve manually or automatically. Similar to ROC curve is the precision-recall curve that plots instead of TPR versus FPR, PPV versus TPR.

3.3 Articles using BIOSEMI dataset for BCI evaluation

This section gives an overview of all the articles that the author could find at the time of writing this thesis that use the dataset described in Section 3.1 and report performances of their implemented BCI. The articles are described to be able to compare them to the results of this work.

3.3.1 Features extraction based on subspace methods with application to SSVEP BCI

In the article by Velchev *et al.* [29], a feature extraction method based on PSD analysis called multiple signal classification (MUSIC) is implemented. Similarly to PSDA method, this method also estimates PSD, but instead of using the periodogram method discussed in Section 2.1, it uses a method called MUSIC. This method uses the covariance matrix of a signal to calculate the estimate.

Theory behind MUSIC method

This section follows the description of the MUSIC method given in the book by Stoica and Moses [25]. MUSIC method makes the assumption, that the signal $x(t)$ can be represented by a series of complex exponentials $\alpha_j e^{i(\omega_j t+\phi_j)}$, $j \in \{1, \dots, n\}$ with additive white noise $e(t)$

$$x(t) = \sum_{j=1}^{n} \alpha_j e^{i(\omega_j t+\phi_j)} + e(t) \tag{3.3}$$

where β_j is the amplitude, ω_j the frequency in units of radians per sample and ϕ_j the initial phase of the corresponding sinusoid [25].

This means that MUSIC method is parametric technique for estimating PSD unlike the periodogram method, which is non-parametric. Therefore, if the assumptions of MUSIC are met, it can outperform the periodogram method [25]. Likewise, if the assumptions are not met, periodogram method is likely to outperform MUSIC method [25].

Let's represent the equation (3.3) with matrices as

$$X(t) = AY(t) + E(t)$$

where

$$A = \begin{pmatrix} a(\omega_1) & a(\omega_2) & \dots & a(\omega_n) \end{pmatrix}$$

$$a(\omega) = \begin{pmatrix} 1 & e^{-i\omega} & e^{-i\omega 2} & \dots & e^{-i\omega(m-1)} \end{pmatrix}^T \tag{3.4}$$

$$Y(t) = \begin{pmatrix} \alpha_1 e^{i(\omega_1 t+\phi_1)} \\ \alpha_2 e^{i(\omega_2 t+\phi_2)} \\ \vdots \\ \alpha_n e^{i(\omega_n t+\phi_n)} \end{pmatrix} \qquad X(t) = \begin{pmatrix} x(t) \\ x(t-1) \\ \vdots \\ x(t-m+1) \end{pmatrix} \qquad E(t) = \begin{pmatrix} e(t) \\ e(t-1) \\ \vdots \\ e(t-m+1) \end{pmatrix}.$$

The covariance matrix of $X(t)$ is

$$R = \mathbf{E}(X(t)X^*(t)) = APA^* + \sigma^2 I$$

where

$$P = \begin{pmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha_n \end{pmatrix}$$

By organising the eigenvectors corresponding to $m-n$ smallest eigenvalues of the matrix R into matrix $G = (\vec{g}_1, \dots, \vec{g}_{m-n})$, it can be shown that frequency components of the signal $x(t)$ are ω for which [25]

$$a^*(\omega)GG^*a(\omega) = 0.$$

First step of the MUSIC method is estimating the covariance matrix of $X(t)$ using the formula

$$\hat{R} = \frac{1}{N}\sum_{t=m}^{N} X(t)X^*(t). \tag{3.5}$$

The next step of MUSIC method is finding the $m-n$ eigenvalues and corresponding eigenvectors $\hat{\vec{g}}_1, \dots, \hat{\vec{g}}_{m-n}$ of the estimated covariance matrix $\hat{R}$.

Finally, the frequencies giving n highest peaks of

$$\frac{1}{a^*(\omega)GG^*a(\omega)}, \qquad \omega \in [-\pi, \pi] \tag{3.6}$$

are considered to be the frequencies of the frequency components [25].

SSVEP-based BCI feature extraction

The first step in the SSVEP-based BCI by Velchev *et al.* [29] is estimating the covariance matrix $\hat{R}$ for each of the used EEG channels as shown in (3.5). Then the eigenvalues and eigenvectors of the matrix $\hat{R}$ have to be calculated.

The size of the noise subspace $m-n$ must be preliminarily specified. Larger noise subspace gives a smooth PSD that may miss some peak frequencies; smaller noise subspace gives detailed PSD that may include spurious frequency peaks [29].

Then the pseudo-PSD in equation 3.6 can be estimated using formula

$$PSD_{pseudo}(f) = \frac{1}{\sum_{i=1}^{m-n} \left|a^T(f) \cdot g_k\right|^2}. \tag{3.7}$$

where $a(t)$ is defined as in equation (3.4) and $g_1, \dots, g_{m-n}$ are the $m-n$ eigenvectors corresponding of $\hat{R}$ to $m-n$ smallest eigenvalues $\lambda_1, \dots, \lambda_{m-n}$.

To get an estimation for PSD, the sum in (3.7) has to be weighed by corresponding eigenvalues [29]

$$PSD(f) = \frac{1}{\sum_{i=1}^{m-n} \left|\frac{1}{\lambda_i} \cdot a^T(f) \cdot g_k\right|^2}.$$

Then to get the features to be used in SSVEP-based BCI, the relative powers p_i for the peaks in PSD of frequencies $\hat{f}_1, \dots, \hat{f}_k$ that are closest to the target frequencies $f_1, \dots, f_k$ have to be calculated [29]

$$p_i = \frac{PSD(\hat{f}_i)}{\sum_{i=1}^{k} PSD(f)}.$$

Another feature that is used in classification is spectrum intensity ratio that is similar to SNR from equations (2.3) and (2.4) that can be used in PSDA method

$$SI_i = \frac{PSD(\hat{f}_i)}{\sum_{f=\hat{f}_i-\frac{B}{2}}^{\hat{f}_i-\Delta f} PSD(f) + \sum_{f=\hat{f}_i+\Delta f}^{\hat{f}_i+\frac{B}{2}} PSD(f)},$$

where B is the bandwidth that was set to 0.5 Hz in the article and Δf is the chosen frequency resolution [29].

The final feature vector to be classified with support vector machine (SVM) with Gaussian kernel using one-vs-one strategy for classification is

$$\left(p_1, \ldots, p_k, SI_1, \ldots, SI_k, \hat{f}_1, \ldots, \hat{f}_k\right).$$

Results

In the work by Velchev *et al.* [29], only two of the three available classes were used. The confusion matrix seen in Table 3.1a is obtained by k-fold cross validation. Velchev *et al.* also experimented with adding a class for absence of stimulus. This approach reduced the accuracy considerably and the confusion matrix can be seen in Table 3.1b.

They also provide a comparison on how the window size affects the accuracy which can be seen in Table 3.2. The results in tables 3.1a and 3.1b were obtained by using window size of 512 samples with an overlap of consecutive segments of 256 samples.

Table 3.1: Confusion matrices for the proposed method [29].

(a) Confusion matrix for two classes.

	Predicted class			
True class	275 46%	24 4%	**TPR** 92%	8 Hz
	29 4.8%	270 45.2%	**TPR** 45.2%	14 Hz
	PPV 90.5%	**PPV** 91.8%	**ACC** 91.1%	

(b) Confusion matrix with absence class.

	Predicted class				
True class	321 32.4%	49 4.9%	22 2.2%	**TPR** 81.9%	None
	97 9.8%	189 19.1%	13 1.3%	**TPR** 63.2%	8 Hz
	35 3.5%	11 1.1%	253 25.6%	**TPR** 84.6%	14 Hz
	PPV 70.9%	**PPV** 75.9%	**PPV** 87.8%	**ACC** 77.1%	

3.3.2 A prototype of SSVEP-based BCI for home appliances control

In the work by Anindya *et al* [1], an SSVEP-based BCI was implemented based on PSDA method that was already discussed in Section 2.1. The PSD was estimated using periodogram method. The feature used for classification was the frequency with the highest amplitude and the results show that the feature extraction method is more accurate with the lower frequencies 8 Hz and 14 Hz.

Table 3.2: Comparison of window size and accuracy [29].

Window size	**Accuracy**
128	63.2%
256	75%
512	91.1%
1024	94.3%

As a preprocessing step, the EEG data was filtered using windowed-sinc digital filter with Blackman window. Blackman window is also used before performing FFT. Only electrodes O1, O2, POz and Oz were used in this work.

The classification was done using SVM and two different kernels were compared. With linear kernel the accuracy was 65% and for radial-base function (RBF) kernel it was 71.67%. More detailed results are shown in Table 3.3.

Table 3.3: Results of the proposed method [1].

Kernel	**Number of trials**	**Correct results**	**Accuracy**
Linear	60	39	65%
RBF	60	43	71.67%

3.3.3 Bio-inspired filter banks for SSVEP-based brain-computer interfaces

In the work by Demir *et al.* [7], a feature extraction method called bio-inspired filter bank (BIFB) is introduced. The method is in essence very similar to PSDA method. Only Oz electrode as used in this work.

In BIFB method, PSD of the EEG signal is estimated using periodogram method and triangular filters are used on the result to make the powers of different frequencies more comparable. This is useful, because higher frequency stimulation produces smaller VEP responses [17].

The triangular filter for target k that has flickering frequency of f_k Hz is defined as

$$H_k(f) = \begin{cases} \frac{f-\left(f_k-\frac{B_k}{2}\right)}{B} \cdot g_k, & \left(f_k - \frac{B_k}{2}\right) \leq f \leq f_k \\ \frac{\left(f_k+\frac{B_k}{2}\right)-f}{B} \cdot g_k, & f_k \leq f \leq \left(f_k + \frac{B_k}{2}\right) \\ 0, & \text{otherwise} \end{cases}$$

where B_k is the bandwidth and g_k the gain of corresponding target k [7]. The filter is shown in Figure 3.1. The peak of the triangle is at the corresponding target frequency, bandwidth determines the length of the base of the triangle and gain determines the height of the triangle. As can be seen in the Figure 3.1, filters can also be designed for the harmonics of the target frequencies. In the figure, the red triangular filter corresponds to the second harmonic of the 8 Hz target.

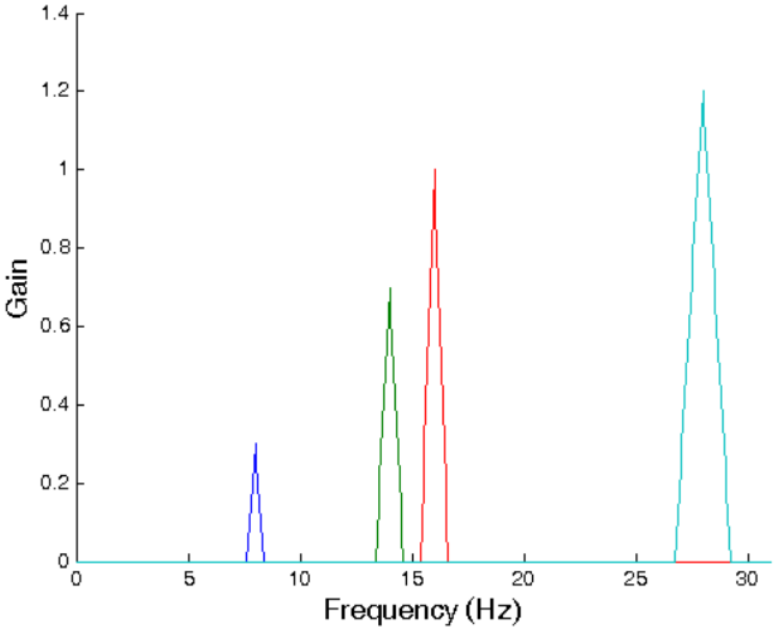


Figure 3.1: Sample triangular filters for 8 Hz, 14 Hz and 28 Hz targets [7].

The filters are multiplied with the periodogram of the EEG signal to obtain the feature c_k for each target k as follows

$$c_k = \sum_f H_k(f) \cdot PSD(f) + \sum_f H_{2k}(f) \cdot PSD(f) \cdot w_k$$

where w_k is the harmonic weight—this allows to change how large impact the harmonic frequency has on the feature value [7]. This method requires optimisation of bandwidth, gain and harmonic weight for each subject.

Demir *et al.* [7] also reported the performances of PSDA and CCA method for comparison. In the PSDA method, a bandpass filter was applied to the EEG signal, and Hamming window was applied before performing PSDA. Finally peak finding algorithm was used on the obtained periodogram. The detection is classified as successful if the target frequency or its second harmonic is detected as peak frequency. The results can be seen in Table 3.4.

Table 3.4: Results of the proposed method compared to PSDA and CCA [7].

Subject	**Number of trials**	**PSDA**			**CCA**			**BIFB**		
		MDT (sec)	**ACC (%)**	**ITR (bits/min)**	**MDT (sec)**	**ACC (%)**	**ITR (bits/min)**	**MDT (sec)**	**ACC (%)**	**ITR (bits/min)**
1	15	10	66.7	2.00	4	73.3	7.2	7.5	100	12.7
2	15	9	66.7	2.22	4	60	3.2	7.8	100	12.2
3	15	15	60	0.9	5	66.7	4	10	86.7	5.3
4	15	15	6.7	-	3	66.7	6.67	8.33	66.7	2.4

3.3.4 Implementation of bilinear separation algorithm as a classification method for SSVEP-based brain-computer interface

In the work by Jukiewicz and Cysewska-Sobusiak [15] PSDA method was used for feature extraction and two classification methods were compared—bilinear separation and SVM. SVM was used with linear kernel as this produced better results than quadratic, polynomial or Gaussian RBF kernel.

They only used electrodes O1, O2 and Oz of the 128 available electrodes and of the three classes only two as in [29], which is described in Section 3.3.1. The features were the amplitudes of the 8 Hz and 14 Hz *frequency components* normalised between 0 and 1 of the EEG signal.

Jukiewicz and Cysewska-Sobusiak [15] used bilinear separation to classify the two classes using the two extracted features. For each sample, two features are extracted and thus the samples can be presented in two-dimensional space. The bilinear separation algorithm finds a value t_k for each feature k that best separates the data to the two classes. An example of classification thresholds and samples are depicted in Figure 3.2.

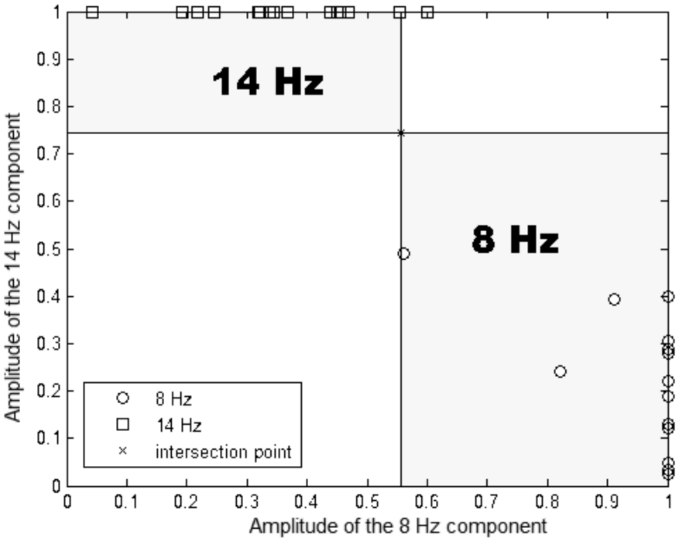


Figure 3.2: Classification using bilinear separation [15].

The classification is done for a sample point (x_1, x_2) according to criterion

$$\text{class} = \begin{cases} \text{14Hz}, & \text{if}(t_1 < x_1) \wedge (x_2 < t_2) \\ \text{8Hz}, & \text{if}(x_1 < t_1) \wedge (t_2 < x_2) \end{cases}. \tag{3.8}$$

The samples that do not satisfy neither of the conditions presented in (3.8) were not classified and therefore the algorithm is able to reject data that it is uncertain about.

The results of the work by Jukiewicz and Cysewska-Sobusiak [15] are shown in figures 3.3, 3.4 and 3.5. The best results are for subject 1 with 93% average accuracy of five-fold cross validation and 33.1 bits/min ITR for the two classes using bilinear separation. Best results for SVM were 90% of average accuracy and 32.8 bit/min ITR.

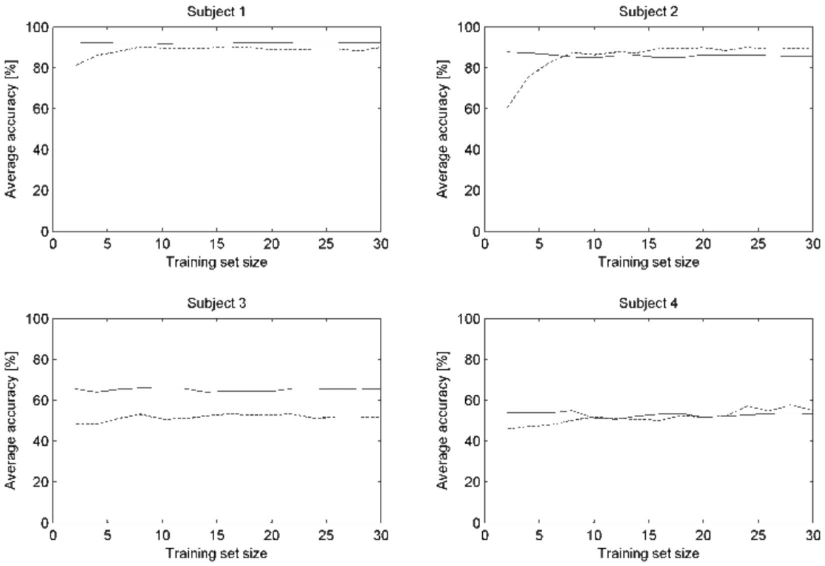


Figure 3.3: Accuracy of the proposed method (continuous line) compared to SVM classification (dashed line) [15].

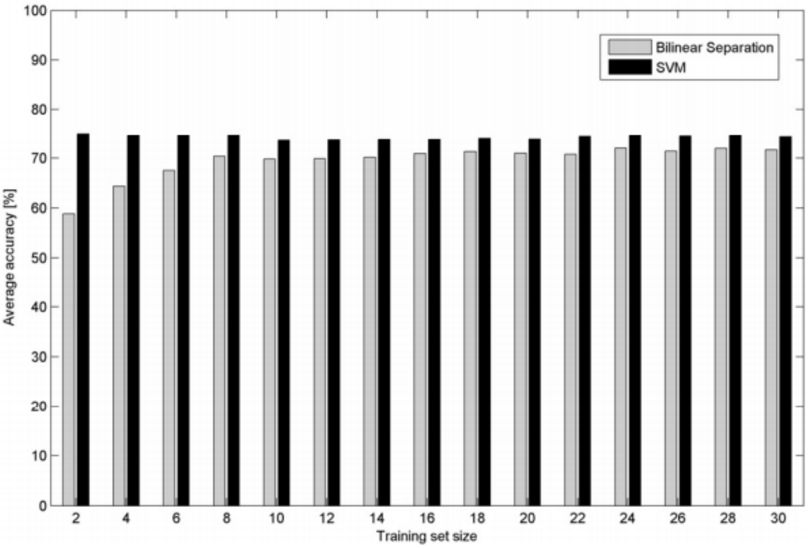


Figure 3.4: Accuracy of the proposed method compared to SVM classification [15].

3.3.5 Frequency detection in medium and high frequency SSVEP based brain computer interface systems by scaling of sine-curve fit amplitudes

In the work by Karnati [17] *et al.* blind source separation (BSS) algorithm called algorithm for multiple unknown signals extraction (AMUSE) is used for preprocessing the signal. During the preprocessing, artefacts caused by eye blinks and muscle activity is removed. Then sine curves are fit to the signal by non-linear least square method. This way the amplitudes for the target frequencies are obtained which are first used to select 8 channels of the available 128 channels. These amplitudes are also calculated online to use these in the classification process. Only two of the available four subject's data was used

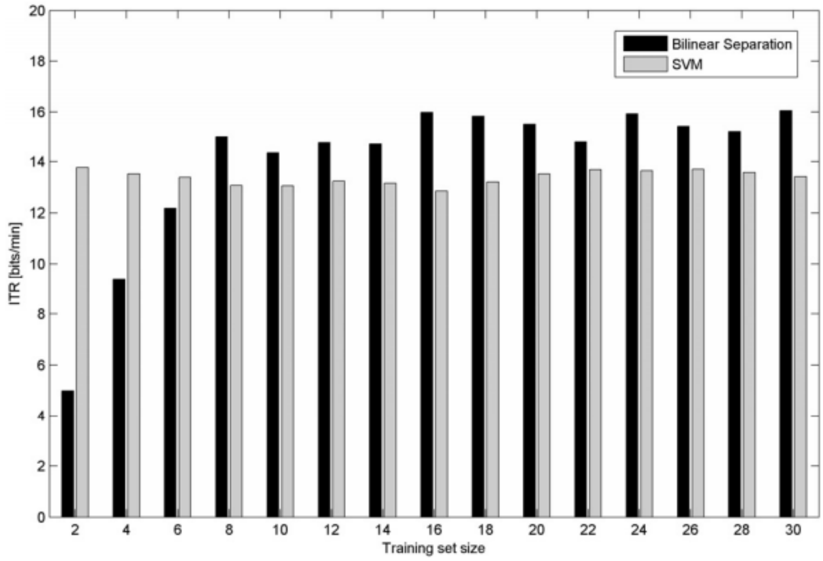


Figure 3.5: ITR of the proposed method compared to SVM classification [15].

and the window length was 0.5 seconds. Machine learning was not used for classification in this work.

Theory behind AMUSE algorithm

This section follows the AMUSE algorithm description given by Tong *et al.* [27]. In BSS it is assumed that the recorded signal $\vec{x}(t) \in \mathbb{R}^n$ can be represented as a sum of source signals $\vec{s}(t) \in \mathbb{R}^m$ scaled with parameters $A \in \mathbb{R}^{n\times m}$ and noise $\vec{n}(t) \in \mathbb{R}^m$

$$\vec{x}(t) = A\vec{s}(t) + \vec{n}(t), \qquad t \in \{1, 2, \ldots, N\}.$$

The AMUSE algorithm estimates the sources $\vec{s}(t)$ and the matrix A with the following steps [27]. First the covariance matrix of the input data is estimated

$$R_x = \mathbf{E}\left(x(t)x^T(t)\right)$$

and then its SVD is calculated

$$R_x = U_x D_x V_x^T.$$

Since R_x is symmetric, then $U_x = V_x$.

Next the number of sources m can be estimated and an orthogonalisation transformation for the matrix A is calculated [27]

$$T = \begin{pmatrix} (D_{11}-\sigma^2)^{-\frac{1}{2}} & 0 & \ldots & 0 \\ 0 & (D_{22}-\sigma^2)^{-\frac{1}{2}} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & (D_{mm}-\sigma^2)^{-\frac{1}{2}} \end{pmatrix} U_x^T.$$

Orthogonalisation of the parameter matrix is performed to reduce the complexity of the BSS problem.

Next, the orthogonalisation is performed

$$\vec{y}(t) = T\vec{x}(t) = TA\vec{s}(t) + T\vec{n}(t)$$

and time-shifted covariance matrix of $\vec{y}(t)$ is calculated for a τ

$$R_y(\tau) = \mathbf{E}\left(\vec{y}(t)\vec{y}(t-\tau)\right)$$

The time-shifted covariance matrix can be symmetrised and by using SVD the matrix U_y can be obtained

$$\frac{R_y(\tau) + R_y(\tau)}{2} = U_y D_y U_y^T. \tag{3.9}$$

From equation (3.9) it can be shown that the matrix A can be estimated as

$$\hat{A} = T^{-1}U_y$$

where T^{-1} can be the pseudo-inverse [27]. Therefore the sources $\vec{s}(t)$ can be estimated as

$$\begin{aligned} T\hat{A}\hat{\vec{s}}(t) &= \vec{y}(t) - T\vec{n}(t) \\ \hat{\vec{s}}(t) &= U_y^T\vec{y}(t) - U_y^T T\vec{n}(t) \end{aligned}$$

and in SSVEP-based BCI the low frequency components corresponding to eye blinks and high frequency components corresponding to muscle activity can be removed from the signal [17].

SSVEP-based BCI feature extraction

Before feature extraction Karnati *et al.* [17] choose 8 channels according to the amplitudes of fit sine curves. The channels are chosen based on the sum of the amplitudes of 14 Hz and 28 Hz fit sine waves. Four channels with highest 14 Hz amplitude and four channels with highest 28 Hz amplitude are chosen as the channels to be used, as they considered 8 Hz target easier to classify. This channel selection is done using data from all trials.

The feature extraction is done in the following steps. First, let $S_{f,s}$ denote the amplitude of fit sine wave where f is the frequency of the sine wave, s the stimulating frequency. Features used in the classification are [17]

$$F_f = S_{f,f} + E_f$$

where E_f is error term calculated as

$$E_f = -(\mathrm{zs}_f \cdot \sigma)$$

where zs_f is z-score of the summed sine wave amplitudes Sum_f for corresponding target f

$$\mathrm{Sum}_f = \sum_{s\in\{8,14,28\}} S_{f,s}$$

and σ is the standard deviation of the amplitudes of fit sine waves with target frequencies. Z-score of a sample Sum_f is calculated as the distance between the value and the mean in terms of standard deviation

$$\mathrm{zs}_f = \frac{\mathrm{Sum}_f - \hat{\mu}}{\hat{\sigma}}.$$

Finally, the target that has the highest feature value F_f is predicted to be the chosen target.

Results

The results of classifying the targets are reported separately for each target. As already mentioned, the window length was 0.5 seconds and the data of two subjects was used. The accuracy of detecting 8 Hz target was 86%, for 14 Hz 83% and for 28 Hz 92%.

3.3.6 Principal component analysis-based spectral recognition for SSVEP-based brain-computer interfaces

In the work by Yehia *et al.* [32], the PSDA method is improved using principal component analysis (PCA) and linear discriminant analysis (LDA) is used for classification. The proposed method is compared to CCA and its improvements: multi-set CCA and multi-way CCA. Different filters were used as a preprocessing step [32]. In this method also subject-specific thresholds are calculated that are used in the classification process. This makes the method adaptable to per-subject variabilities.

As a preprocessing step, the EEG signal is filtered with bandpass filter and then processed with common average reference (CAR) filter to remove common noise artefacts. Finally moving average filter (MAF) is applied to the signal.

SSVEP-based BCI feature extraction

The feature extraction method proposed by Yehia *et al.* [32] estimates periodogram to find the powers of target frequencies. The target frequency that has the highest power is considered to be the detected frequency.

For a detected frequency f, confidence is calculated as

$$\mathrm{conf}_i(f) = \frac{P_i(f) - P_i(f_n)}{P_i(f)} \tag{3.10}$$

where i denotes the channel, P_i denotes the periodogram for the signal of i-th channel and f_n denotes the target frequency that had the lowest power among all the target frequencies.

Eight channels of the available 128 were used in this work—P7, P3, Pz, P4, P8, O1, Oz and O2. But for each subject, only these channels i were used that had weight w_i above subject-specific threshold w_{th} that was determined using grid search. The weight w_i was determined on training set. Every time that the detected frequency f matched the expected frequency, the weight w_i was increased by the confidence $\mathrm{conf}_i(f)$ calculated using (3.10).

Finally PCA is applied to the periodogram of each channel. During training, the number of principal components m to be used was decided by comparing the classification accuracy and choosing m for which the accuracy was highest. This is also subject-specific.

Results

The results of the proposed method are compared to multi-set CCA and multi-way CCA. Multi-set CCA uses subject-specific reference signals that are obtained through training instead of the set of signals presented in equation (2.5). Multi-set CCA maximises the correlation between SSVEP signals across the training trials during training phase. Multi-way CCA also calculates subject specific reference signals, but does so by maximising correlation between SSVEP signals and the reference signals (2.5) during training phase. The comparison of the results can be seen in Table 3.5.

Table 3.5: Performance of the proposed method compared to variations of CCA method [32]

Subject	**Accuracy (%)**			
	PCA-SR	**Multi-way CCA**	**Multi-set CCA**	**CCA**
1	92.43	80.50	69.98	62.69
2	87.39	71.16	58.35	59.52
3	94.56	76.76	79.80	65.51
4	76.61	60.53	47.17	56.04

3.3.7 Conclusion

In two of the articles [29, 15] only two of the three classes were used, the rest of the articles [1, 7, 17, 32] used all three available classes to measure the performance of their implemented method. The articles which used two classes achieved similar performances with 1 second window length, the accuracy for both of these were around 75%. In the article by Velchev *et al.* [29], it was demonstrated, however, that increasing the time window increases the accuracy to over 90%.

Of the articles where three classes were used, the article by Yehia *et al.* [32] and the article by Demir *et al.* [7] are more detailed than the other two and achieve superior performance. Demir *et al.* [7] even achieve 100% accuracy for two of the subjects, but this comes at the cost of having long MDT. Yehia *et al.* [32] unfortunately do not report results separately for different window lengths, but only average performance over different window lengths or average over different datasets. The detailed results can be seen in Table 3.6.

This concludes the overview of the related work where the dataset [2] was used to evaluate BCI classification algorithms. Now, by having overview of the state of the art result, in the next chapter a method of classifying the targets of a BCI is proposed and in Chapter 5 the results of the proposed method are compared to results of the related work.

Table 3.6: Results of the related articles.

Article	**Section**	**Feature extraction**	**Classification**	**Preprocessing**	**Classes**	**Window length/ MDT (s)**	**Subjects**	**Channels**	**Accuracy (%)**
[29]	3.3.1	MUSIC	SVM, Gaussian kernel, one-vs-one	N/A	8 Hz, 14 Hz	0.5	All	N/A	63.2
						1			75
						2			91.1
						4			94.3
[1]	3.3.2	PSDA	SVM linear	Windowed sinc filter, Blackman window	8 Hz, 14 Hz, 28 Hz	N/A	All	O1, O2, POz, Oz	65
			SVM RBF						71.67
[7]	3.3.3	BIFB	No machine learning (but requires finding subject specific parameters)	Bandpass filter, Hamming window	8 Hz, 14 Hz, 28 Hz	7.5	1	Oz	100
						7.8	2		100
						10	3		86.7
						8.33	4		66.7
		PSDA				10	1		66.7
						9	2		66.7
						15	3		60
						15	4		6.7
		CCA		N/A		4	1		73.3
						4	2		60
						5	3		66.7
						3	4		66.7
[15]	3.3.4	PSDA	Bilinear separation	N/A	8 Hz, 14 Hz	1	1	O1, O2, Oz	93
						1	All		∼74
			SVM			1	1		90
						1	All		∼71
[17]	3.3.5	Least square sine fitting	No machine learning	AMUSE	8 Hz	0.5	Two subjects	8 subject specific channels	86
					14 Hz				83
					28 Hz				92
[32]	3.3.6	PSDA + PCA	LDA (and grid search for subject specific parameters)	Bandbass filter, CAR, MAF	8 Hz, 14 Hz, 28 Hz	Average over time windows up to 4 s	1	Subject specific from the selection: P7, P3, Pz, P4, P8, O1, Oz, O2	92.43
							2		87.39
							3		94.56
							4		76.61
		Multi-way CCA					1		80.50
							2		71.16
							3		76.76
							4		60.53
		Multi-set CCA					1		69.98
							2		58.35
							3		79.80
							4		47.17
		CCA					1		62.69
							2		59.52
							3		65.51
							4		56.04

4 Proposed classification method

In this chapter, a method of classifying targets of a BCI with thresholds and a method for finding these thresholds is proposed. Finding the thresholds will be formalised as maximisation task, where the value to be maximised is a performance measure for the BCI. Therefore, first the performance measure is derived, a method of calculating it is proposed and finally the gradient of the performance measure is calculated. By using the gradient, thresholds that maximise the performance measure can be found.

4.1 Classifying with cut-off thresholds

In this section, the proposed method of classifying targets of a BCI from features using cut-off thresholds is described. The features can be extracted by one of the methods described in Section 2. For all the described feature extraction methods, a higher feature value for a target means that the corresponding target is more likely to be the user's choice. Therefore, using a threshold that indicates how large a feature value has to be in order to predict that target is a reasonable approach.

The proposed rule for classifying with thresholds is defined as follows. Assuming that there are n targets in a BCI, let us denote a class corresponding to a target as $k \in [n] = \{1, 2, \dots, n\}$, a cut-off threshold for a class k as t_k and feature extracted by a feature extraction method for the class k as f_k. Cut-off threshold is a value t_k such that if a sample has feature f_k for which $f_k \geq t_k$ and for all the other classes $k' \in [n] \setminus k$ it holds that $f_{k'} < t_{k'}$ then the sample is classified as class k.

It can be seen that classifying by the described rule can lead to a situation where a sample does not belong to any of the classes. The samples that have all the class features f_k smaller than corresponding thresholds t_k will not be classified and the samples that have multiple feature values over the thresholds will not be classified. In simple words, if the BCI is uncertain about a sample, it will not make a prediction. Having a possibility to not make classification for a sample is good for filtering out false classifications which in case of a BCI are much more costly than not making a prediction. This is so under the assumption that having the BCI do nothing is more beneficial than executing wrong commands.

Not making a prediction will be depicted on confusion matrices with additional "nothing" class. In case of three classes ($n = 3$), the confusion matrix could look as shown in Table 4.1a. It can be seen from the last column of the confusion matrix that for 306 samples that were actually from class 1 classification was not made. For class 2 and 3, 289 and 554 samples were not classified respectively. The last row will always consists of only zeroes as there are no samples for which "nothing" is the correct class.

In the calculation of accuracy for the confusion matrices, the last row and column will be discarded, therefore using only the part of the matrix that is coloured green. This captures the idea that not making a prediction is better than making a wrong prediction. The last column is thus provided just to give an overview of how many samples were not

Table 4.1: Confusion matrix examples.

(a) Confusion matrix with many predictions.

	Predicted class				
True class	259	0	0	306	1
	3	270	3	289	2
	1	0	10	554	3
	0	0	0	0	Nothing

(b) Confusion matrix with few predictions.

	Predicted class				
True class	1	0	0	564	1
	0	3	0	562	2
	0	0	0	565	3
	0	0	0	0	Nothing

classified. The accuracy for matrix 4.1a is therefore

$$ACC = \frac{\text{Trace of matrix}}{\text{Sum of elements exluding last column}} = \frac{259 + 270 + 10}{259 + 270 + 10 + 3 + 3 + 1} \approx 0.987$$

while for matrix 4.1b is is

$$\text{ACC} = \frac{1+3}{1+3} = 1.$$

As can be seen, it might not always be the best idea to choose classifier with best accuracy, as with the added “nothing” class the classifier can make only very few predictions, thus the user has to wait very long before getting a response from the BCI. Thus, there is a trade-off between accuracy and the number of predictions made. Fortunately, in the BCI context, this problem has been already addressed by introducing ITR as performance measure and it also works in the case of added “nothing” class.

Recall from Section 3.2 that ITR depends on the accuracy and MDT of the classifier. One way to calculate the MDT in units of second per command for offline experiments is

$$\text{MDT}_{\text{offline}} = \frac{\text{Length of input data (s)}}{\text{Number of predictions made}} \tag{4.1}$$

which for matrices 4.1a and 4.1b would give

$$\text{MDT}_{\text{offline}} = \frac{225}{259 + 270 + 10 + 3 + 3 + 1} \approx 0.41$$

and

$$\text{MDT}_{\text{offline}} = \frac{225}{1+3} = 56.25$$

respectively, assuming the length of the input EEG data was 225 seconds in both cases. Therefore, for matrix 4.1b the classifications were made with 100% accuracy but on average the classifications took 56.25 seconds to make, while for matrix 4.1a the accuracy was also very high, but classification only took on average approximately 0.41 seconds.

Note that MDT of 0.41 seconds is quite unrealistic for online BCI as it takes time for the user to shift gaze between targets, it takes time to elicit SSVEP in brain and to obtain enough EEG data for feature extraction methods to extract reliable features.

The reason why so low MDT is achievable in offline case, is that it is desirable in offline case to have as many samples as possible to obtain reliable results and therefore all the data that should contain SSVEP data is used in the analysis. For example, when using a BCI online, it is reasonable to discard all the previous EEG data after a classification is made as the previous data does not contain information about which target the user wishes to choose next. In offline case it is known what is the expected target and if after a classification the expected target stays the same, discarding the data is not required. Not discarding the data gives more data for analysis.

Therefore, to not discard data from analysis and to obtain better online ITR estimate, a different method for calculating MDT is needed. The method for estimating MDT that would be obtainable in online case when using the same classifier that was trained offline is derived in the Section 4.2.

4.2 Online ITR estimation

Estimating online MDT and online ITR is required in the proposed method of thresholds optimisation described in the following sections. The threshold optimisation will be formalised as a maximisation task of the online ITR estimate. Since the actual online ITR is different for different thresholds, the online ITR cannot just be measured online, but it has to be estimated in the threshold optimisation process using the information obtained in offline analysis.

First, let us introduce the required notation. Let s denote the time step length between consecutive feature extractions and let w denote the window length which shows how much EEG data is used by feature extraction method at each time step, both in units of seconds. Therefore, when starting an online BCI, first w seconds of EEG data is obtained and then the first feature extraction method will be made. Then, additional s seconds of data will be recorded and again the last w seconds of recorded data will be used in feature extraction. Thus the overlap of consecutive windows is $w - s$ seconds.

Following the example given before, in the online case after a classification is made the last w seconds of data should be discarded and not used in the next feature extraction as this data mostly contains the information about which command the user wanted to choose previously and not the information about which command the user wishes to choose next. In offline testing, the w seconds of data have to be discarded not after every classification, but every time after the expected target changes.

Therefore, to estimate online MDT, it is needed to know how often the classifier makes a prediction, that is, does not predict “nothing” class. This is needed because the assumption was made that with every prediction $w - s$ seconds of data is discarded in online case that is not discarded in offline case.

Making a prediction at a time step can be modelled as Bernoulli random variable $X \sim B(p)$ as it has two possible outcomes—with probability p the prediction is made and with probability $1 - p$ the prediction is not made. The probability p can be estimated as the proportion of predictions made to all samples

$$p = \frac{\text{Number of samples not classified as “nothing” class}}{\text{Number of samples}}. \tag{4.2}$$

Therefore, assuming that the probability of making a prediction is the same at each time step and that at least $w - s$ seconds of data is already collected, the number of time steps needed to take before making a prediction can be modelled as geometric random variable $Y \sim G(p)$ as it is the number of prediction trials needed to get one successful prediction. Therefore, the expected number of time steps needed before a prediction is made, that is the expected value of $Y \sim G(p)$, is

$$\mathbf{E}(Y) = \frac{1}{p}.$$

Now, by making the assumption that the probability of making a prediction at a time step will be the same for online case as it was for offline, the online MDT can be estimated as

$$\widehat{\mathrm{MDT}} = w - s + \frac{1}{p} \cdot s = w + \left(\frac{1}{p} - 1\right) \cdot s. \tag{4.3}$$

For example, if $p = 1$, then at every time step a prediction is expected to be made and MDT is the lowest $\widehat{\mathrm{MDT}} = w$. If $p = \frac{1}{2}$ then prediction is expected to be made after every two time steps and $\widehat{\mathrm{MDT}} = w + s$ etc.

A relationship between the $\mathrm{MDT}_{\text{offline}}$ calculated as equation (4.1) and $\widehat{\mathrm{MDT}}$ is that they are equal if the same amount of data segments of length $w - s$ are discarded in both of them or in other words, the number of expected target changes is equal to the number of prediction predictions made.

Finally, to estimate the online ITR of the classifier from the results obtained offline, an additional assumption has to be made that the accuracy will be the same in online case as it was offline. If the given assumptions are satisfied, ITR can be estimated as

$$\widehat{\mathrm{ITR}} = \mathrm{ITR}_c \cdot \frac{60}{\widehat{\mathrm{MDT}}}$$

where ITR_c is calculated as in (3.1) and $\widehat{\mathrm{MDT}}$ as in (4.3).

Therefore, in this section a method for estimating online performance of a BCI based on offline performance has been proposed to evaluate the classifiers of BCIs and to find optimal thresholds for classification. These ideas are used in the following sections to describe the proposed algorithm for finding the optimal thresholds.

4.3 ITR for unbalanced classes, predictions and accuracies

As discussed in section 3.2, commonly used performance measure for a BCI is ITR presented in formula (3.1). But the ITR as presented in (3.1) makes many assumptions about the obtained results. In particular, the classification accuracy has to be the same for each target, all the predictions have to be equally likely and, in the case of making a wrong prediction, all the wrong targets have to be equally likely to be chosen [31]. An example of a confusion matrix that satisfies all these assumptions is given in table 4.2a.

Table 4.2: Confusion matrix examples.

(a) Balanced matrix.

	Predicted		
True class	20	4	4
	4	20	4
	4	4	20

(b) Predicted to 2 classes.

	Predicted			
True class	51	0	0	49
	0	48	0	52
	0	0	1	99
	0	0	0	0

(c) Predicted to 3 classes.

	Predicted			
True class	21	0	1	78
	0	22	2	76
	0	0	10	90
	0	0	0	0

As can be seen, a confusion matrix satisfying all the assumptions for formula (3.1) looks artificial. In real applications, the classifier is likely to be able to classify certain classes better than others and is more likely to make false classifications to certain classes. Furthermore, when using added “nothing” class as described in section 4.1, it is possible for the confusion matrix to become even more unbalanced.

Consider the example confusion matrices given in tables 4.2b and 4.2c where the “nothing” class is added. In these cases, even if there is equal amount of samples from each class, for some thresholds it might happen that there are no samples classified as one of the classes. This raises the question of how to evaluate the result and which confusion matrix is better.

In more detail, results from matrix 4.2b show that the classifier was practically not able to classify samples as class 3. Using formula (3.1) would give an ITRs of 1.58 and 1.23 bits per prediction for matrices 4.2b and 4.2c respectively. But the actual amount of transferred information is different, because the formula assumes that there are three classes and each of the classes is equally well classified. For matrix 4.2b, however, the classifier practically ignores to class 3. Therefore it would be more fair, if in the calculation of ITR for matrix 4.2b the number of classes was considered 2 instead of 3. In this case the ITR would be 1 bit per prediction instead of 1.58 which shows that actually matrix 4.2c is better than 4.2b.

Next, an ITR calculating method is derived that does not make any of the assumptions that the formula (3.1) makes and thus gives more accurate ITR when the prediction probabilities, accuracies or classes are unbalanced. Being able to calculate accurate ITR is essential in the threshold optimisation process as it will be formalised as ITR maximisation task.

Let P and C denote the random variables that model the predicted class and correct class respectively. Let P_i and C_j denote the events that $P = i$ and $C = j$ for classes i and j respectively. ITR can be calculated as the *mutual information* of these random variables. Mutual information shows how much information is obtained about one random variable through the other. In the case of a BCI, it shows how much information the predicted class gives about the correct class.

In case of n classes $[n] = \{1, 2, \ldots, n\}$, the mutual information $I(P, C)$ between P and C

can be calculated as

$$I(P,C) = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{P}\big(P_i \cap C_j\big) \log_2 \left(\frac{\mathbf{P}(P_i \cap C_j)}{\mathbf{P}(P_i) \cdot \mathbf{P}(C_j)} \right).$$

If the previously discussed assumptions are true, meaning that

$$\begin{aligned} \mathbf{P}(P_i) &= \mathbf{P}(P_j), && i,j \in [n] \\ \mathbf{P}(C_i \mid P_i) &= \mathbf{P}(C_j \mid P_j), && i,j \in [n] \\ \mathbf{P}(C_i \mid P_k) &= \mathbf{P}(C_j \mid P_k), && i,j \in [n],\ \text{and } k \in [n] \setminus \{i,j\} \end{aligned}$$

which gives $\mathbf{P}(P_i) = \frac{1}{n}$ and by denoting $\mathbf{P}(C_i \mid P_i) = p$

$$\mathbf{P}(C_i \mid P_j) = \frac{1-p}{n-1} \qquad\qquad i,j \in [n], \quad i \neq j,$$

then

$$\begin{aligned} I(P,C) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{P}\big(C_i \mid P_j\big)\mathbf{P}(P_j) \log_2 \left(\frac{\mathbf{P}(C_i \mid P_j)\mathbf{P}(P_j)}{\mathbf{P}(C_i) \cdot \mathbf{P}(P_j)} \right) \\ &= np \cdot \frac{1}{n} \log_2(np) + n(n-1) \cdot \frac{1-p}{n-1} \cdot \frac{1}{n} \log_2 \left(\frac{1-p}{n-1} \cdot n \right) \\ &= p\log_2 p + p\log_2 n + (1-p)\log_2 \left(\frac{1-p}{n-1} \right) + \log_2 n - p\log_2 n \\ &= \log_2 n + p\log_2 p + (1-p)\log_2 \left(\frac{1-p}{n-1} \right) \end{aligned}$$

which is the formula in (3.1). Therefore the traditional ITR (3.1) is a special case of the mutual information of P and C and thus the mutual information can be used to calculate the amount of information transferred in one minute

$$\text{ITR}_{MI} = I(P,C) \cdot \frac{60}{\text{MDT}} \tag{4.4}$$

similarly as for the traditional ITR.

Mutual information between the predicted class P and correct class C is more powerful as a performance measure than the traditional ITR (3.1) as it can be used in the case when classes are unbalanced or prediction accuracies are very different. Having a good performance measure is essential in the proposed threshold finding method, as it will be formalised as maximisation of the performance measure.

4.4 Calculating mutual information for different thresholds

To formalise the threshold finding as a maximisation task, first assumptions on the distribution of the features are made to be able to calculate the performance measure for different threshold values t_i.

Let F_i denote a random variable which models the feature value for target $i \in [n] = \{1, 2, \ldots, n\}$ where, as before, n is the number of targets. F_i is a continuous random variable. Let C denote a random variable that models the correct class, thus taking values from $[n]$. And finally, let P denote a random variable that denotes the predicted class.

It was observed empirically, that features extracted by the described feature extraction methods can be modelled well by skew normal distribution. Skew normal distribution was fit to the features given a class histograms depicted in Figure 4.1. The histograms of features given a class are shown for subject 1 from dataset [2] and features were extracted with CCA method with three harmonics. Recall that the features extracted by CCA method are the canonical correlation between EEG signal and the corresponding set of reference signals for each target at each time step. Thus, in this case $F_i \mid C_j$ models the canonical correlation between EEG signal and the reference signals of class i while the correct class is j. As can be seen, the fit skew normal probability density functions (PDFs) closely resemble the histograms. The parameters for skew normal distribution were found using least squares method for curve fitting.

The advantage of using the estimated PDFs in formalising the maximisation task is that the PDFs are continuous, unlike the histograms, and their definite integrals can be calculated using the corresponding cumulative distribution functions (CDFs). The relationship between a PDF $f_X(x)$ and a CDF $F_X(x)$ of a continuous random variable X is

$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt.$$

Therefore, the value of a CDF $F_X(x)$ at x is equal to the probability that $X \leq x$

$$F_X(x) = \mathbf{P}(X \leq x) = \mathbf{P}(X < x).$$

This means that for a random variable F_i, which models the feature value for class i, its CDF at t_i gives the probability that the feature value is smaller than the given threshold t_i. Similarly, the probability that feature value is larger than threshold t_i is equal to the *complementary CDF* at t_i

$$\bar{F}_{F_i}(t_i) = 1 - \mathbf{P}(F_i > t_i) = 1 - \mathbf{P}(F_i \geq t_i).$$

since the event $F_i \geq t_i$ is complementary to the event of the feature being smaller than t_i. The complementary CDFs of the corresponding PDFs from Figure 4.1 can be seen in Figure 4.2.

Therefore, according to classification rule presented in 4.1, the probability of a sample being classified as class i is

$$\mathbf{P}(P_i) = \mathbf{P}\left(F_i \geq t_i \cap \left(\bigcap_{j \in [n] \setminus \{i\}} F_j < t_j\right)\right). \tag{4.5}$$

Being able to calculate this probability is essential in the proposed classification method.

To use the previously fit skew normal distribution CDFs in the calculation of (4.5), the event P_i has to be conditioned on a class C_k, as the fit skew normal distributions were

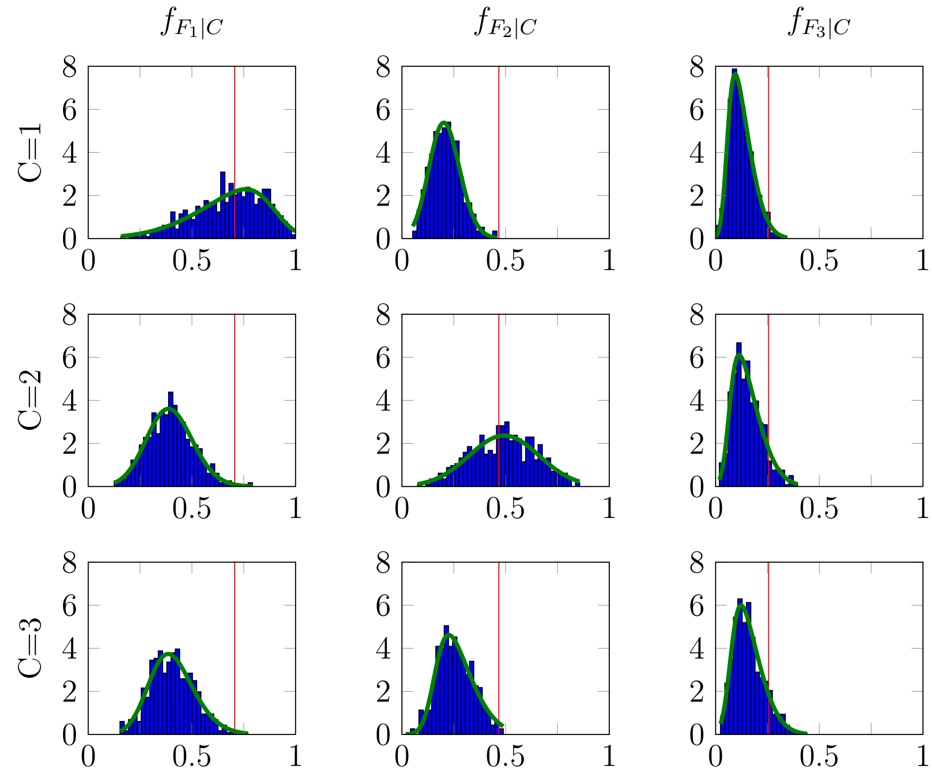


Figure 4.1: Histograms of features given class (blue). Green line denotes skew normal distribution fit to the data and red lines indicate possible cut-off thresholds.

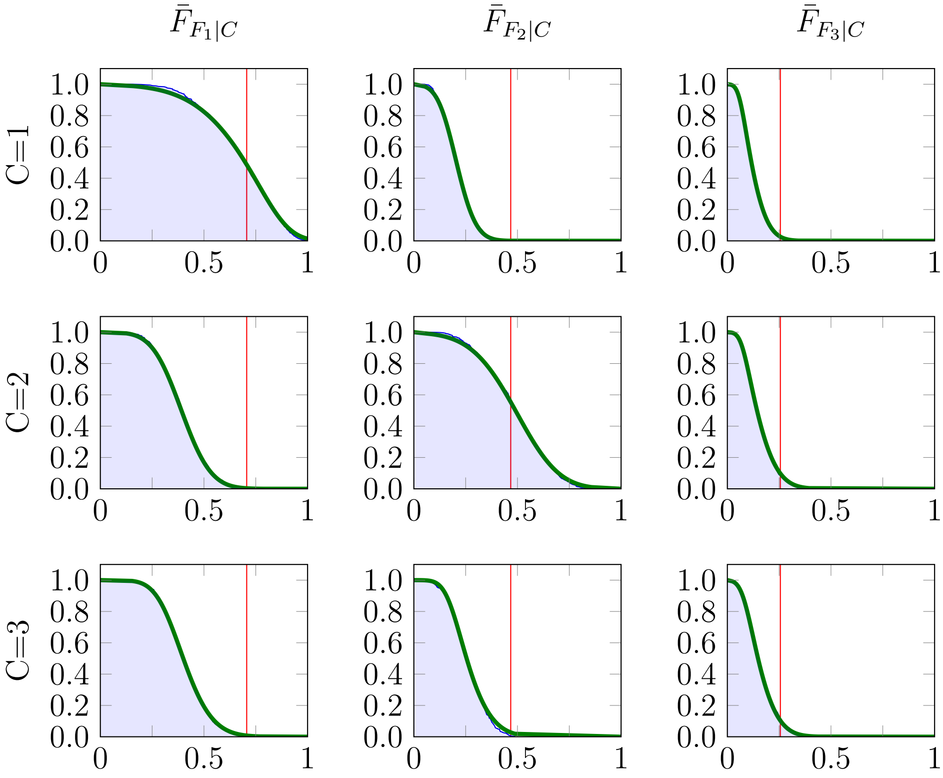


Figure 4.2: Complementary CDFs of the corresponding PDFs in Figure 4.1.

fit to $P_i \mid C_k$. However, to use the CDFs for calculating the probability, the features, conditioned on C_k, have to be mutually independent. This means that for the CDFs $F_{F_1|C_k}, \ldots, F_{F_n|C_k}$ of random variables $F_1 \mid C_k, \ldots, F_n \mid C_k$ the following holds

$$\prod_{i=1}^{n} F_{F_i|C_k}(f_i) = F_{F_1,\ldots,F_n|C_k}(f_1, \ldots, f_n)$$

where $F_{F_1,\ldots,F_n|C_k}$ is the joint CDF.

Whether the conditional independence of features is a reasonable assumptions can be inspected from Figure 4.3. In this figure, only the data for class 1 of subject 1 was used and features were extracted using CCA with three harmonics. Only dependence that can be seen from the figure is between feature 2 and feature 3, which occurs probably due to the overlap of two reference signals for 14 Hz and 28 Hz frequency targets as three harmonics were used in CCA. Otherwise no clear dependence between features can be seen from the figure, which of course does not verify that the features are conditionally independent, put provides evidence that the assumption is reasonable. Another thing to note is that the figures only provide pairwise scatterplots, while the assumption was made that the features are mutually independent. Scatterplots for all three classes can be seen in Figure 4.4.

Therefore, the probability of predicting class i when the correct class is k under the assumption that features conditioned on a class are mutually independent is

$$\begin{aligned}
\mathbf{P}(P_i \mid C_k) &= \mathbf{P}\left(F_i \geq t_i \cap \left(\bigcap_{j\in[n]\setminus\{i\}} F_j < t_j\right) \middle| C_k\right) \\
&= \mathbf{P}\left(\bigcap_{j\in[n]\setminus\{i\}} F_j < t_j \middle| C_k\right) - \mathbf{P}\left(\bigcap_{j\in[n]} F_j < t_j \middle| C_k\right) \\
&= \prod_{j\in[n]\setminus\{i\}} \mathbf{P}(F_j < t_j \mid C_k) - \prod_{j\in[n]} \mathbf{P}(F_j < t_j \mid C_k) \\
&= (1 - \mathbf{P}(F_i < t_i \mid C_k)) \cdot \prod_{j\in[n]\setminus\{i\}} \mathbf{P}(F_j < t_j \mid C_k) \\
&= \mathbf{P}(F_i \geq t_i \mid C_k) \cdot \prod_{j\in[n]\setminus\{i\}} \mathbf{P}(F_j < t_j \mid C_k) \\
&= \bar{F}_{F_i|C_k}(t_i) \prod_{j\in[n]\setminus\{i\}} F_{F_j|C_k}(t_j)
\end{aligned} \tag{4.6}$$

which means that the probability under consideration can be calculated using the CDFs and a complementary CDF of the previously fit skew normal distributions.

Now, since the events C_k partition the sample space, the probability for predicting a class (4.5) can be calculated using the formula of total probability

$$\mathbf{P}(P_i) = \sum_{j=1}^{n} \mathbf{P}(P_i \mid C_j)\mathbf{P}(C_j)$$

where $\mathbf{P}(C_j)$ is simply the proportion of samples from class j to all samples and $\mathbf{P}(P_i \mid C_j)$ can be calculated as in 4.6.

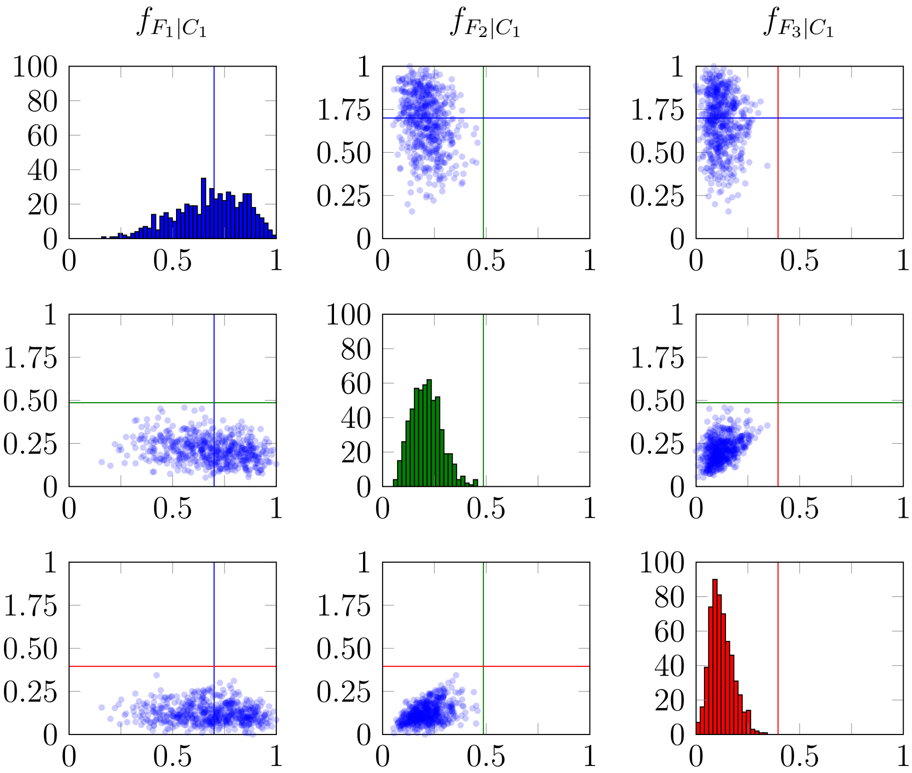


Figure 4.3: Histograms of features and pairwise scatterplots of features, both of samples from class 1. Coloured lines denote possible cut-off thresholds for features.

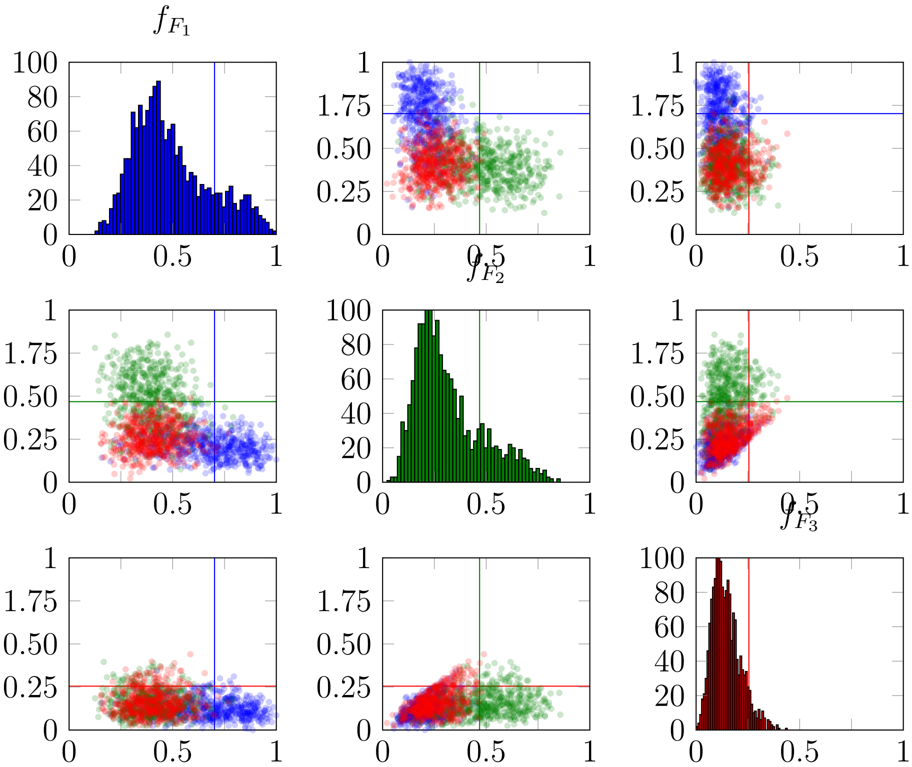


Figure 4.4: Histograms of features and pairwise scatterplots of features for samples from class 1 (blue), class 2 (green) and class 3 (red).

Therefore, under the independence assumption all the probabilities required for calculating the mutual information between P and C can be calculated using the CDFs. But since the presented classification rule can sometimes not make a prediction, that is predict a sample as "nothing" class, the calculation of these probabilities should take this into account. This is similar to the calculation of accuracy presented in Section 4.1 that only uses part of the confusion matrix to calculate the accuracy. This approach is reasonable, because it allows the classifier to not make a prediction when it is not confident enough, while not making enough predictions is penalised by the MDT.

Therefore, let M denote the event that prediction was made, that is "nothing" class was not predicted. The probability of M can be calculated as

$$\mathbf{P}(M) = \mathbf{P}\left(\bigcup_{i=1}^{n} P_i\right) = \sum_{i=1}^{n} \mathbf{P}(P = i)$$

because the events P_i are mutually exclusive, meaning that none of the samples are predicted as multiple classes. Note that $\mathbf{P}(M)$ is equal to p from (4.2). Now by using the probability of M, the conditional probability of predicting a class i and the conditional probability of a class j conditioned on M can be calculated as

$$\mathbf{P}(P_i \mid M) = \frac{\mathbf{P}(P_i \cap M)}{\mathbf{P}(M)} = \frac{\mathbf{P}(P_i)}{\mathbf{P}(M)}$$

and

$$\begin{aligned}\mathbf{P}(C_j \mid M) &= \frac{\mathbf{P}\left(C_j \cap \left(\bigcup_{i=1}^{n} P_i\right)\right)}{\mathbf{P}(M)} \\ &= \frac{\mathbf{P}\left(\bigcup_{i=1}^{n} (C_j \cap P_i)\right)}{\mathbf{P}(M)} \\ &= \frac{1}{\mathbf{P}(M)} \sum_{i=1}^{n} \mathbf{P}(C_j \cap P_i) \\ &= \frac{1}{\mathbf{P}(M)} \sum_{i=1}^{n} \mathbf{P}(P_i \mid C_j)\mathbf{P}(C_j)\end{aligned}$$

where again the disjointness of events P_i for $i \in [n]$ was used. As can be seen, the probabilities of $P_i \mid M$ and $C_j \mid M$ can also be calculated using the CDFs.

Finally, the probability of predicting i given that class is j conditioned on M, or symbolically $\mathbf{P}(P_i \mid C_j, M)$, has to be calculated. For this the following probability is needed

$$\mathbf{P}(M \mid C_j) = \sum_{i=1}^{n} \mathbf{P}(P = i \mid M).$$

Thus the last probability needed to calculate mutual information in case of added "nothing" class is

$$\mathbf{P}(P_i \mid C_j, M) = \frac{\mathbf{P}(P_i \cap C_j \cap M)}{\mathbf{P}(C_j \cap M)} = \frac{\mathbf{P}(P_i \cap M \mid C_j)\mathbf{P}(C_j)}{\mathbf{P}(C_j \cap M)} = \frac{\mathbf{P}(P_i \mid C_j)}{\mathbf{P}(M \mid C_j)}$$

Therefore, by making the assumptions that features are mutually independent given a class and features given a class have a certain distribution for which PDF can be fit to the data and its CDF can be calculated, then the mutual information between the predicted classes $P_i \mid M$ and correct classes $C_j \mid M$ can be calculated using just the CDFs fit to data and the proportions of classes. To make the maximisation of the ITR based on mutual information easier, in the next section the gradient of the mutual information is calculated.

4.5 The rate of change of ITR

In Section 4.3 a method for calculating the ITR based on mutual information ITR_{MI} from thresholds was introduced 4.4 and in the previous section it was shown that it can be calculated using just the CDFs of the distributions of features given class and proportions of different classes. In this section it is shown that the gradient of ITR_{MI} can be calculated using in addition to CDFs and class proportions, the corresponding PDFs. The gradient will be used in finding the optimal thresholds for classification.

There are some optimising algorithms that can be used to find local maxima of a function which take only the function itself as an input, but other algorithms also use the information about the rate of change of the function—its gradient. Some algorithms even take the Hessian as input, which is a matrix consisting of second order derivatives of the objective function. In this work, a method based on gradient descent algorithm is used for optimisation, which requires the gradient of the objective function.

First note that due to the definition of CDF, the PDF $f_{F_i|C_k}$ of a random variable $F_i \mid C_k$ is the derivative of the corresponding CDF $F_{F_i|C_k}$

$$\frac{dF_{F_i|C_k}}{dt_i} = f_{F_i|C_k}.$$

For the complementary CDF $\bar{F}_{F_i|C_k}$ the relationship with PDF is

$$\frac{d\bar{F}_{F_i|C_k}}{dt_i} = \frac{d}{dt_i}(1 - F_{F_i|C_k}) = -f_{F_i|C_k}.$$

These relationships allow the gradient of ITR_{MI} to be easily calculated.

The partial derivatives of the functions derived in the Section 4.4 needed for the calculation of the gradient of ITR_{MI} for $i, k, \ell \in [n]$ are

$$\begin{aligned}
\frac{\partial}{\partial t_\ell}\mathbf{P}(P_i \mid C_k) &= \begin{cases} -f_{F_i|C_k}(t_i) \cdot \prod_{j \in [n](t_i) \setminus \{i\}} F_{F_j|C_k}(t_j) & \text{if } i = \ell \\ \bar{F}_{F_i|C_k}(t_i) \cdot f_{F_\ell|C_k}(t_\ell) \cdot \prod_{j \in [n] \setminus \{i,\ell\}} F_{F_j|C_k}(t_j) & \text{if } i \neq \ell \end{cases} \\
\frac{\partial}{\partial t_\ell}\mathbf{P}(P_i) &= \sum_{j=1}^{n} \mathbf{P}(C_j) \frac{\partial}{\partial t_\ell}\mathbf{P}(P_i \mid C_j) \\
\frac{\partial}{\partial t_\ell}\mathbf{P}(M) &= \sum_{j=1}^{n} \frac{\partial}{\partial t_\ell}\mathbf{P}(P_j) \\
\frac{\partial}{\partial t_\ell}\mathbf{P}(P_i \mid M) &= \frac{\mathbf{P}(M)\frac{\partial}{\partial t_\ell}\mathbf{P}(P_i) - \mathbf{P}(P_i)\frac{\partial}{\partial t_\ell}\mathbf{P}(M)}{(\mathbf{P}(M))^2}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial t_\ell}\mathbf{P}(C_k \mid M) &= \frac{\left(\mathbf{P}(M)\sum_{j=1}^{n}\frac{\partial}{\partial t_\ell}\mathbf{P}(P_j \cap C_k)\right) - \left(\sum_{j=1}^{n}\mathbf{P}(P_j \cap C_k)\frac{\partial}{\partial t_\ell}\mathbf{P}(M)\right)}{(\mathbf{P}(M))^2}\\
\frac{\partial}{\partial t_\ell}\mathbf{P}(P_i \mid C_j, M) &= \frac{\mathbf{P}(M \mid C_j)\frac{\partial}{\partial t_\ell}\mathbf{P}(P_i \mid C_j) - \mathbf{P}(P_i \mid C_j)\frac{\partial}{\partial t_\ell}\mathbf{P}(M \mid C_j)}{(\mathbf{P}(M \mid C_j))^2}\\
\frac{\partial}{\partial t_\ell}\mathbf{P}(M \mid C_j) &= \sum_{i=1}^{n}\frac{\partial}{\partial t_\ell}\mathbf{P}(P_i \mid C_j)
\end{aligned}$$

where the partial derivatives are taken with respect to the threshold t_ℓ of the feature for class ℓ. Now only the partial derivatives of ITR_{MI} remain to be found to be able to calculate the gradient of ITR_{MI}.

To present the partial derivatives of mutual information more compactly, note that mutual information of random variables P and C can be calculated as

$$I(P, C) = H(C) - H(C \mid P) = H(P) - H(P \mid C)$$

where $H(C)$ is the *entropy* of C and $H(C \mid P)$ is the conditional entropy. For C, Entropy shows the average amount of information in the correct classes and the conditional entropy $H(C \mid P)$ shows how much additional information is needed to know the correct class, given that the predicted class is known. Formulas for calculating entropy and conditional entropy are

$$\begin{aligned}
H(P) &= -\sum_{i=1}^{n}\mathbf{P}(P_i)\log_2\mathbf{P}(P_i)\\
H(P \mid C) &= \sum_{j=1}^{n}\mathbf{P}(C_j)H(P \mid C = j).
\end{aligned}$$

and the corresponding partial derivatives are

$$\begin{aligned}
\frac{\partial}{\partial t_\ell}H(P) &= -\sum_{i=1}^{n}\frac{\ln(\mathbf{P}(P_i)) + 1}{\ln 2}\frac{\partial}{\partial t_\ell}\mathbf{P}(P_i)\\
\frac{\partial}{\partial t_\ell}H(P \mid C) &= \sum_{j=1}^{n}\left(H(P \mid C_j)\frac{\partial}{\partial t_\ell}\mathbf{P}(C_k) + \mathbf{P}(C_k)\frac{\partial}{\partial t_\ell}H(P \mid C_j)\right).
\end{aligned}$$

Finally, the partial derivatives of mutual information, $\widehat{\mathrm{MDT}}$ and $\widehat{\mathrm{ITR}}_{MI}$ are

$$\frac{\partial}{\partial t_\ell}I(P \mid M, C \mid M) = \frac{\partial}{\partial t_\ell}H(P \mid M) - \frac{\partial}{\partial t_\ell}H(P \mid C, M),$$

$$\frac{\partial}{\partial t_\ell}\widehat{\mathrm{MDT}} = -\frac{s}{(\mathbf{P}(M))^2}\cdot\frac{\partial}{\partial t_\ell}\mathbf{P}(M)$$

and

$$\frac{\partial}{\partial t_\ell}\mathrm{ITR}_{MI} = \left(\widehat{\mathrm{MDT}}\frac{\partial}{\partial t_\ell}I(P \mid M, C \mid M) - I(P \mid M, C \mid M)\frac{\partial}{\partial t_\ell}\widehat{\mathrm{MDT}}\right)\frac{60}{\widehat{\mathrm{MDT}}^2}$$

where

$$I(P \mid M, C \mid M) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{P}(P_i \mid C_j, M)\mathbf{P}(C_j \mid M) \log_2 \left(\frac{\mathbf{P}(P_i \mid C_j, M)}{\mathbf{P}(C_j \mid M)} \right).$$

Thus the gradient of $\widehat{\mathrm{ITR}}_{MI}$ is

$$\nabla \widehat{\mathrm{ITR}}_{MI} = \left(\frac{\partial}{\partial t_1} \widehat{\mathrm{ITR}}_{MI}, \ldots, \frac{\partial}{\partial t_n} \widehat{\mathrm{ITR}}_{MI} \right)^T.$$

Therefore, it has been shown that the gradient of the proposed performance measure can be calculated using just the CDFs, PDFs described in Section 4.4 and class probabilities. As the gradient shows the direction of the greatest rate of increase, it can be used to find the thresholds that maximise ITR.

4.6 Method of finding the thresholds

In this section, the proposed method of finding the cut-off thresholds for classification described in sections 4.1–4.5 is summarised. First, assumption was made that the features are conditionally independent conditioned on correct class and that the features are from known conditional distribution, conditioned on correct class, for which PDF and CDF can be calculated. It was observed in Section 4.4 that skew normal distribution fits the data well. The parameters of the distribution can be calculated using least squares method for curve fitting, or if the closed form solution for calculating the maximum likelihood estimates of the parameters is known, this can be used instead.

Then, a way of estimating online ITR of a classifier given offline results was proposed in Section 4.2 and in Section 4.3 an ITR calculation method was derived that does not make the strong assumptions that the standard ITR calculation method makes. Results from these sections were used to define the performance measure that is used in finding the optimal thresholds.

In Section 4.4 it was shown that the proposed performance measure can be calculated directly from CDFs of the conditional feature distribution and the proportion of classes and in section 4.5 the gradient of the performance measure was calculated. This gradient can be used to find thresholds that maximise the proposed performance measure. In this work, gradient descent, or rather gradient ascent algorithm is used to solve the maximisation task.

The performance measure to be maximised is the function $\widehat{\mathrm{ITR}}_{MI}(\vec{t})$ where $\vec{t} = (t_1, \ldots, t_n)$ denotes the vector of thresholds. The idea of the gradient descent algorithm is simple. It starts at a point $\vec{t_0}$ which might be randomly chosen or chosen by some heuristic and calculates the gradient at this point. Depending on how large the gradient is, the input $\vec{t_0}$ is updated by the rule

$$\vec{t}_{m+1} = \vec{t}_m + \mu \nabla \mathrm{ITR}(\vec{t}_m)$$

where μ is input parameter for the algorithm called step size. As gradient gets smaller, that is the function value approaches a local maximum, the updating step gets also smaller, until it converges, that is meets stopping criteria. The output of the algorithm is the vector $\vec{t}$ which gave the largest value of ITR.

In this work different values of μ were used for different feature extraction methods and different BCI parameters. The stopping criteria was that if the ITR improves less than 10^{-6} in two consecutive steps, then the algorithm stops and returns the thresholds. Twenty different initialisation points were used and the final result was the threshold vector that corresponded to the largest ITR of the obtained results.

Thus by using an optimisation algorithm the thresholds that maximise the performance measure can be found and therefore, the problem of finding optimal thresholds for classifying the targets of a BCI as described in Section 4.1 is solved.

4.7 Empirically choosing parameters

In this section the effect of normalising features and using moving average filter (MAF) filter on features before classification is investigated. Normalising features provides better overview of how large the feature for one class is compared to the features of other classes at the same time step. Using MAF filter can be beneficial to average out sudden changes in feature values that most likely can be attributed to noise. MAF of length m works by replacing each m consecutive values with their average.

For testing how MAF and normalising affect the classification, all the data from dataset [2] was used. The classifier was evaluated using 5-fold cross validation, thus in each fold there are 1 fold of data for testing and 4 folds for training. The thresholds were chosen as the mean thresholds from 4-fold cross validation of the training data. In each fold of the 4-fold cross validation, the gradient descent algorithm was ran as described in Section 4.5.

The results can be seen in tables 4.3 and 4.4. CCA and PSDA feature extraction methods were used, both with 3 harmonics for each class. Both methods were used with window length of $w = 1$ second and time step of $s = 0.125$ seconds. Feature "PSDA" denotes the sum of feature values for all three harmonics; "PSDA_1" and "PSDA_2" correspond to the first and second harmonic respectively. "Pred" denotes the number of predictions made.

It can be seen from the tables that as hypothesised in the beginning of the section, using MAF filter and normalised features indeed slightly improves the performance of the classifier.

4.8 Combining features from different extraction methods

So far, the classification has only been discussed in the case where there are n classes, each class has a corresponding feature and for each feature a threshold is found. But it would be desirable to combine features from different feature extraction methods in which case each class would have multiple features. Using multiple feature extraction methods has the benefit of providing more information to the classifier and thus potentially making the BCI more accurate.

Table 4.3: Effect of normalising features.

(a) Not normalised features.

		S1	S2	S3	S4	Avg
CCA	**ITR$_{\text{MI}}$**	**58.90**	**42.01**	**9.56**	**9.26**	**29.93**
	ITR	66.53	44.78	11.10	5.22	31.91
	ACC	0.97	0.88	0.66	0.53	0.76
	MDT	1.20	1.23	1.76	1.38	1.39
	Pred	643	595	240	417	473.75
PSDA	**ITR$_{\text{MI}}$**	**33.06**	**24.34**	**0.58**	**0.00**	**14.49**
	ITR	33.22	21.03	0.20	0.00	13.61
	ACC	0.81	0.73	0.30	0.33	0.54
	MDT	1.24	1.32	1.36	71.50	18.85
	Pred	582	477	439	3	375.25
PSDA_1	**ITR$_{\text{MI}}$**	**29.47**	**21.31**	**1.93**	**0.00**	**13.18**
	ITR	30.20	17.07	3.77	0.03	12.77
	ACC	0.77	0.66	0.18	0.25	0.47
	MDT	1.16	1.16	1.30	53.84	14.36
	Pred	743	746	503	4	499.00
PSDA_2	**ITR$_{\text{MI}}$**	**8.99**	**5.00**	**2.03**	**0.00**	**4.00**
	ITR	4.20	3.07	1.01	0.16	2.11
	ACC	0.50	0.48	0.25	0.00	0.31
	MDT	1.20	1.22	1.35	212.75	54.13
	Pred	649	616	445	1	427.75
Average	**ITR$_{\text{MI}}$**	**32.60**	**23.16**	**3.52**	**2.32**	**15.40**
	ITR	33.54	21.49	4.02	1.35	15.10
	ACC	0.76	0.69	0.35	0.28	0.52
	MDT	1.20	1.23	1.44	84.87	22.19
	Pred	654.25	608.50	406.75	106.25	443.94

(b) Normalised features.

			S2	S3	S4	Avg
CCA	**ITR$_{\text{MI}}$**	**62.03**	**47.43**	**11.17**	**12.25**	**33.22**
	ITR	60.49	44.90	11.97	8.91	31.56
	ACC	0.92	0.85	0.68	0.60	0.77
	MDT	1.11	1.12	1.83	1.47	1.38
	Pred	910	857	223	355	586.25
PSDA	**ITR$_{\text{MI}}$**	**38.98**	**24.82**	**14.57**	**2.24**	**20.15**
	ITR	36.42	20.62	10.14	1.06	17.06
	ACC	0.80	0.70	0.58	0.42	0.62
	MDT	1.08	1.17	1.04	1.38	1.17
	Pred	1016	727	1250	420	853.25
PSDA_1	**ITR$_{\text{MI}}$**	**32.25**	**24.33**	**7.37**	**3.81**	**16.94**
	ITR	30.48	19.48	4.98	0.29	13.81
	ACC	0.76	0.69	0.50	0.40	0.59
	MDT	1.07	1.17	1.03	2.89	1.54
	Pred	1078	723	1338	105	811.00
PSDA_2	**ITR$_{\text{MI}}$**	**10.28**	**5.49**	**6.90**	**0.76**	**5.86**
	ITR	4.14	2.04	4.40	0.45	2.76
	ACC	0.51	0.45	0.49	0.41	0.47
	MDT	1.37	1.34	1.03	2.37	1.53
	Pred	426	453	1350	142	592.75
Average	**ITR$_{\text{MI}}$**	**35.89**	**25.52**	**10.00**	**4.76**	**19.04**
	ITR	32.88	21.76	7.87	2.68	16.30
	ACC	0.75	0.67	0.56	0.46	0.61
	MDT	1.16	1.20	1.23	2.03	1.41
	Pred	857.50	690.00	1040.3	255.50	710.81

To combine multiple feature extraction methods linear discriminant analysis (LDA) was used in this work. LDA is a dimensionality reduction and classification method that finds a projection of the input data to a lower dimensional space so that the variance between the classes is maximised with respect to the variance within the classes. To classify a sample, LDA finds which class mean is closest to the sample in the lower dimensional space and classifies the sample as this class. Therefore, decision borders can be found in the lower dimensional space that divide the samples into different classes.

LDA can be used to reduce the number of features when multiple feature extraction methods are used by finding the described lower dimensional space and calculating the signed distances of the samples to the decision borders. If there are $n > 2$ classes, there will be n decision borders and therefore n distances for each sample. These distances can be interpreted as confidence scores for corresponding classes. Therefore, no matter how many features there are, LDA always projects the data into $n-1$ dimensional space and finds n hyperplanes as decision borders, thus giving one feature for each class.

By taking again the data of subject 1 from [2] as an example and using the same features as shown in Table 4.4, the samples are projected to lower dimensional space (see Figure 4.6). In Figure 4.5 it can be seen that skew normal distribution fits well the new features too.

To summarise, LDA can take each sample's m features extracted by the feature extraction

Table 4.4: Effect of normalising features when using MAF filter on features.

(a) Not normalised features.

		S1	S2	S3	S4	Avg
CCA	**ITR$_{MI}$**	**52.15**	**43.86**	**12.23**	**14.31**	**30.64**
	ITR	57.59	47.97	13.47	8.11	31.79
	ACC	0.97	0.94	0.73	0.62	0.81
	MDT	1.42	1.48	2.10	1.79	1.70
	Pred	694	579	214	311	449.50
PSDA	**ITR$_{MI}$**	**35.30**	**25.12**	**0.29**	**0.00**	**15.18**
	ITR	37.12	20.27	0.13	0.09	14.40
	ACC	0.87	0.75	0.30	0.20	0.53
	MDT	1.47	1.55	1.75	42.75	11.88
	Pred	606	485	334	5	357.50
PSDA_1	**ITR$_{MI}$**	**34.22**	**25.35**	**1.77**	**0.00**	**15.34**
	ITR	35.16	18.42	0.00	0.00	13.40
	ACC	0.85	0.72	0.33	0.33	0.56
	MDT	1.43	1.47	1.61	35.81	10.08
	Pred	673	608	427	6	428.50
PSDA_2	**ITR$_{MI}$**	**10.56**	**6.51**	**1.81**	**0.00**	**4.72**
	ITR	5.64	4.66	0.37	0.50	2.79
	ACC	0.55	0.54	0.28	0.00	0.34
	MDT	1.49	1.62	1.59	70.50	18.80
	Pred	567	423	444	3	359.25
Average	**ITR$_{MI}$**	**33.06**	**25.21**	**4.03**	**3.58**	**16.47**
	ITR	33.88	22.83	3.49	2.17	15.59
	ACC	0.81	0.74	0.41	0.29	0.56
	MDT	1.45	1.53	1.76	37.71	10.62
	Pred	635.00	523.75	354.75	81.25	398.69

(b) Normalised features.

		S1	S2	S3	S4	Avg
CCA	**ITR$_{MI}$**	**55.60**	**47.33**	**13.71**	**14.47**	**32.78**
	ITR	54.52	44.08	16.12	9.91	31.16
	ACC	0.94	0.90	0.74	0.63	0.80
	MDT	1.33	1.37	1.90	1.61	1.55
	Pred	1008	857	270	433	642.00
PSDA	**ITR$_{MI}$**	**43.41**	**28.20**	**16.68**	**5.06**	**23.34**
	ITR	40.93	22.90	11.18	2.84	19.46
	ACC	0.87	0.76	0.62	0.49	0.68
	MDT	1.33	1.41	1.29	1.56	1.40
	Pred	995	718	1231	479	855.75
PSDA_1	**ITR$_{MI}$**	**37.49**	**23.87**	**8.44**	**1.09**	**17.72**
	ITR	36.36	17.18	6.15	0.10	14.95
	ACC	0.85	0.69	0.54	0.36	0.61
	MDT	1.34	1.35	1.30	1.51	1.37
	Pred	974	905	1222	542	910.75
PSDA_2	**ITR$_{MI}$**	**15.96**	**7.52**	**8.01**	**1.22**	**8.17**
	ITR	7.52	3.29	5.08	0.19	4.02
	ACC	0.59	0.51	0.52	0.37	0.50
	MDT	1.57	1.74	1.28	1.41	1.50
	Pred	467	337	1349	729	720.50
Average	**ITR$_{MI}$**	**38.11**	**26.73**	**11.71**	**5.46**	**20.50**
	ITR	34.83	21.86	9.63	3.26	17.40
	ACC	0.81	0.71	0.61	0.46	0.65
	MDT	1.39	1.47	1.44	1.52	1.46
	Pred	861.00	704.25	1018	545.75	782.25

Table 4.5: CCA and PSDA features combined with LDA.

	S1	S2	S3	S4	Avg
ITR$_{MI}$	**57.85**	**47.66**	**36.83**	**8.95**	**37.82**
ITR	56.13	42.25	33.55	8.99	35.23
ACC	0.95	0.89	0.86	0.63	0.83
MDT	1.34	1.39	1.51	1.72	1.49
Pred	974	794	535	348	662.75

methods as input, where m can be larger than the number of classes n, and produces n new features that correspond to the confidence scores of the sample belonging to corresponding classes. Thus LDA can be used to combine information from different feature extraction methods or from the features of different classes.

To see how well combining features with LDA works compared to using just one feature extraction method, consider the Table 4.5. In this table, the results of using the described threshold finding method on the new features can be seen. As can be seen from tables 4.3 and 4.4, on average combining features from different feature extraction methods indeed outperforms each of the methods separately.

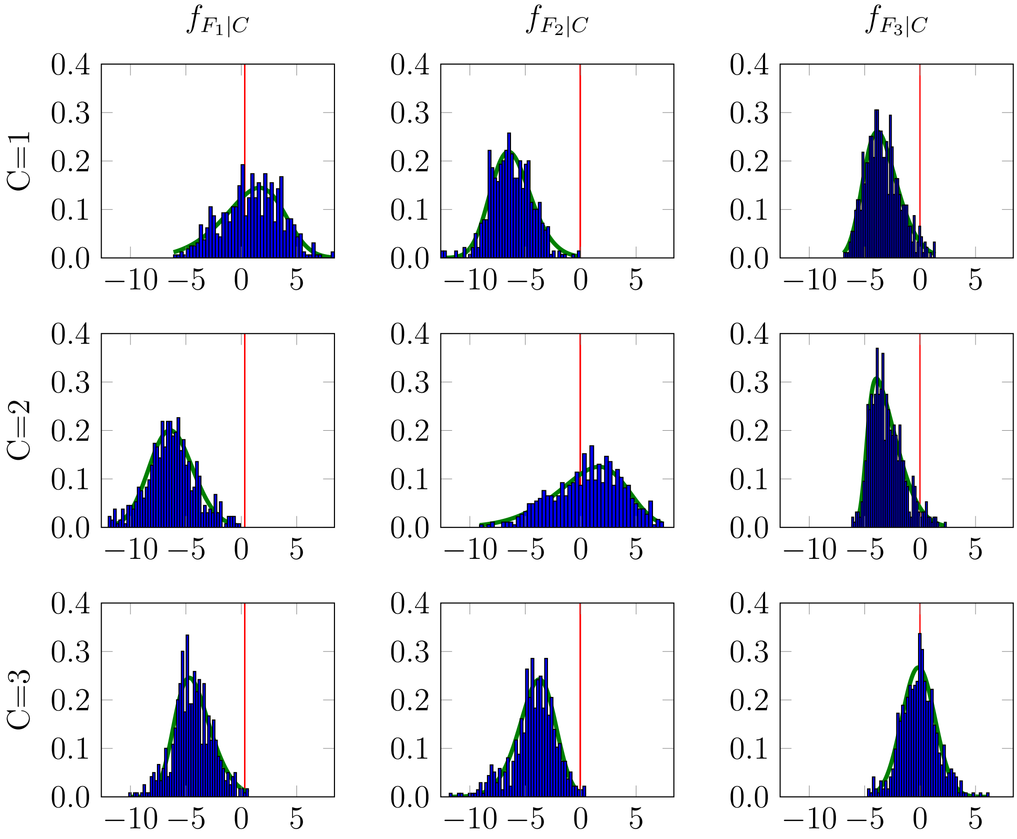


Figure 4.5: Skew normal fits projected samples.

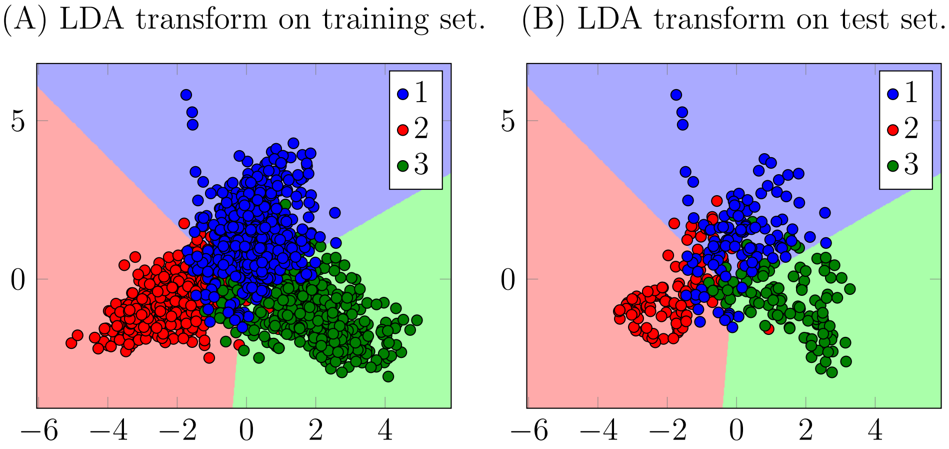


Figure 4.6: Samples in lower dimensional space.

5 Results

5.1 Feature value change over time

In the results given in previous sections, it was not addressed how the feature values change over time. To know how well the classifier would work in online case, it is important to know whether a feature value is below its threshold for very long periods of time or it is able to make predictions in roughly equal time steps. This is especially important in the case with added “nothing” class as too many consecutive predictions predictions to “nothing” class would make the BCI unresponsive.

In figures 5.1 and 5.2, the changes of feature values of subject 1 from dataset [2] are depicted. In this case, CCA and PSDA feature extraction methods were used, both with three harmonics, and their features were combined with LDA as described in Section 4.8.

As can be seen from the figures, the predictions are not accumulated around certain time interval, but happen every with similar frequency over the whole training and test set. Therefore, it can be expected that for subject 1 the BCI would not have long unresponsive periods in online case and the classifier is thus suitable for online usage.

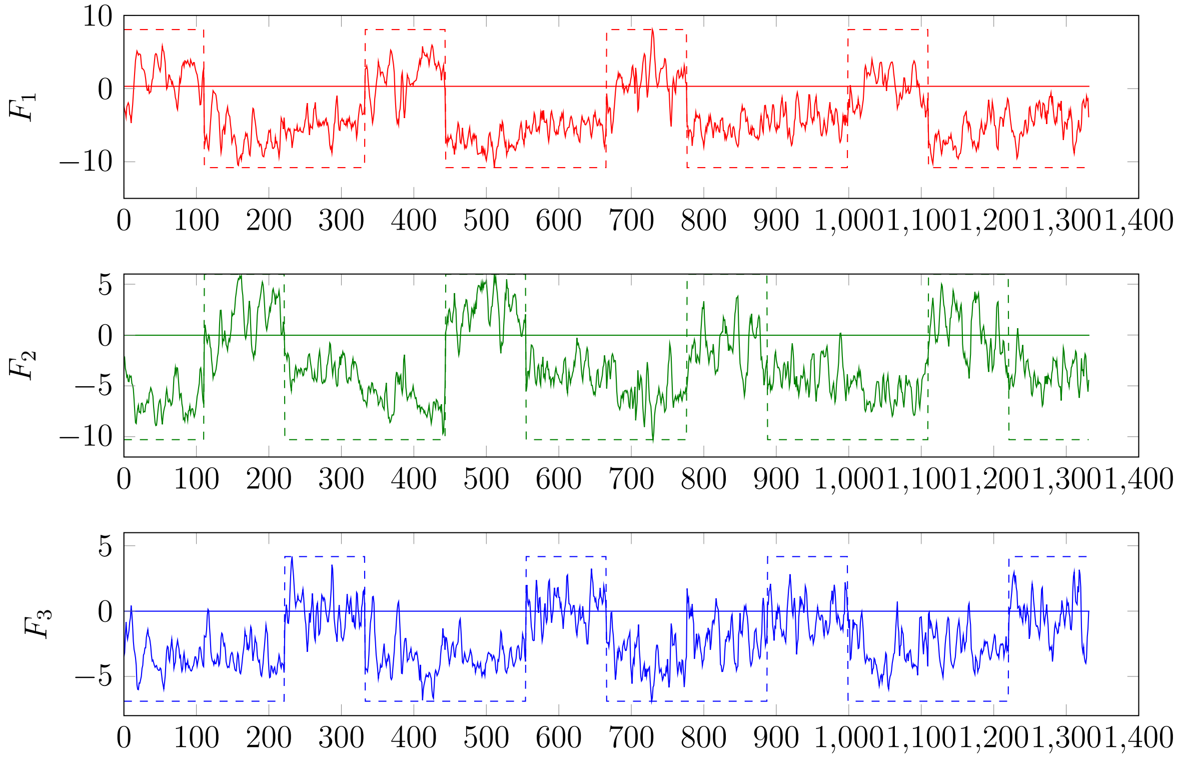


Figure 5.1: Feature value over time in training set. When dashed line is higher for a feature, this means that the corresponding target is currently the expected target. Straight line denotes cut-off threshold.

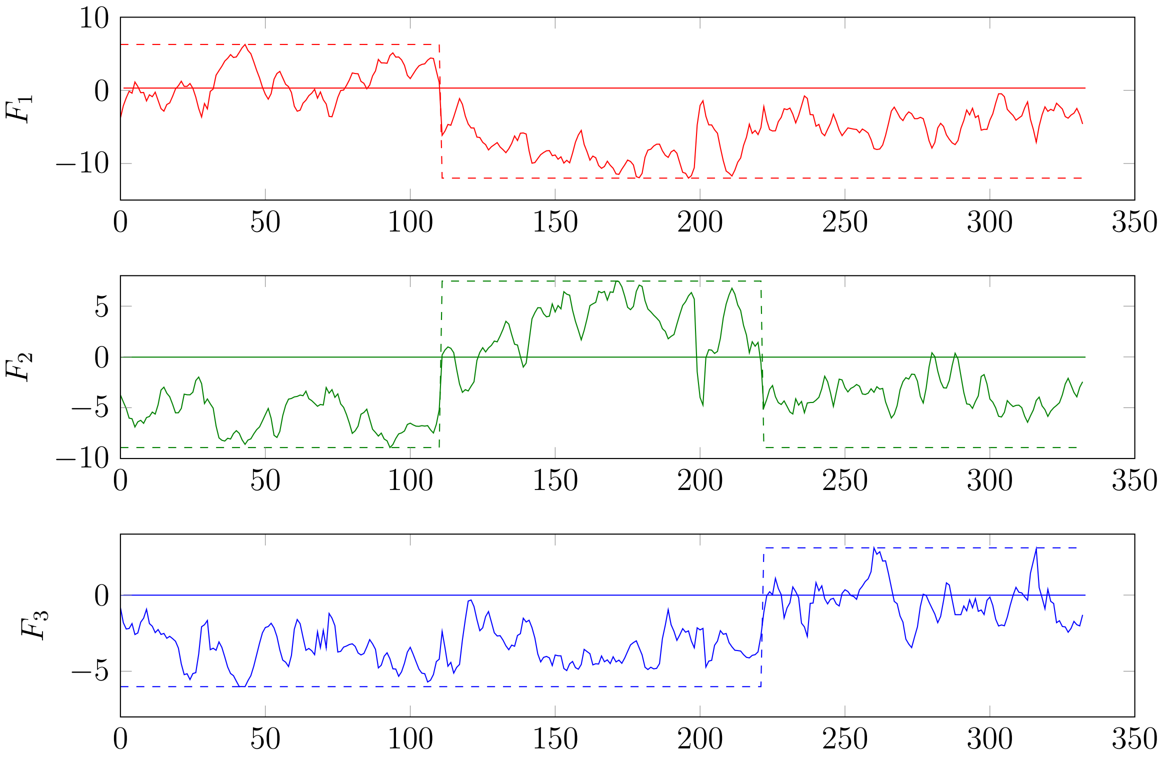


Figure 5.2: Feature value over time in test corresponding to the training set used in Figure 5.1.

5.2 Comparison to Random Forest

In this section, the proposed classification method is compared to the widely used Random Forest. It should be noted, however, that in this comparison Random Forest is at a disadvantage, because the proposed method can not make a prediction if it is uncertain about a sample while Random Forest has to classify all the samples. Random Forest was used with 50 trees, the rest of the parameters were the default parameters provided by scikit-learn version 0.18.1 [23].

To make this comparison more fair, the proposed classifier will also be forced to classify all the samples. The rule for classifying the samples for which either more than one feature value was larger than its threshold or all the feature values were lower than their thresholds will be the following: these samples will be classified to the class k whose feature value's f_k signed distance to its threshold t_k is the largest

$$\arg\max_k f_k - t_k. \tag{5.1}$$

The results will be shown for both, the original proposed classifier and the classifier which is not allowed to predict "nothing". The results can be seen in Table 5.1. As hypothesised before, the proposed classifier outperforms Random Forest as it is designed to maximise ITR and can not classify a sample. The current test shows that Random Forest is slightly better than the classifier that does not predict "nothing". The higher ITR was achieved thanks to having lower MDT and lower MDT was achieved because the proposed classifier used MAF filter. Few additional tests suggest that without MAF filter the proposed classifier without “nothing” class is practically as good as Random Forest.

Table 5.1: Comparison of the proposed classifier to Random Forest when without using MAF filter.

		S2	**S2**	**S1**	**S3**	**S1**	**Avg**
		PSDA	**PSDA** **CCA**	**MEC** **LRT**	**PSDA** **CCA**	**PSDA** **CCA**	
Random Forest	**ITR$_{\mathrm{MI}}$**	**15.60**	**31.01**	**24.71**	**19.08**	**40.40**	**26.16**
	ITR	10.95	26.03	22.70	15.62	38.63	22.79
	ACC	0.58	0.71	0.83	0.63	0.79	0.71
	MDT	1.00	1.00	2.00	1.00	1.00	1.20
	Pred	1695	1695	1575	1695	1695	1671
Proposed classifier without “nothing”	**ITR$_{\mathrm{MI}}$**	**16.47**	**31.62**	**25.39**	**19.32**	**41.91**	**26.94**
	ITR	12.18	26.94	22.83	13.12	38.31	22.68
	ACC	0.59	0.72	0.83	0.60	0.79	0.71
	MDT	1.00	1.00	2.00	1.00	1.00	1.20
	Pred	1695	1695	1575	1695	1695	1671
Proposed classifier	**ITR$_{\mathrm{MI}}$**	**32.37**	**53.73**	**36.31**	**33.19**	**64.62**	**44.04**
	ITR	30.14	48.31	34.84	27.61	62.33	40.65
	ACC	0.80	0.89	0.94	0.82	0.94	0.88
	MDT	1.32	1.22	2.08	1.56	1.14	1.47
	Pred	479	606	940	308	790	624.6

Table 5.2: Comparison of the proposed classifier to Random Forest when using MAF filter.

		S1	**S1**	**S1**	**S3**	**S4**	**S2**	**Avg**
		CCA **PSDA**	**CCA**	**CCA** **PSDA**	**LRT**	**CCA** **PSDA**	**CWT** **LRT**	
	Window	**1**	**1**	**2**	**2**	**4**	**2**	
Random Forest	**ITR$_{\mathrm{MI}}$**	**47.65**	**39.27**	**33.39**	**5.85**	**2.46**	**24.43**	**25.51**
	ITR	45.91	36.11	32.61	5.76	0.97	23.54	24.15
	ACC	0.83	0.77	0.91	0.59	0.48	0.84	0.74
	MDT	1	1	2	2	4	2	2
	Pred	1665	1665	1545	1545	1305	1545	1545
Proposed classifier without “nothing”	**ITR$_{\mathrm{MI}}$**	**38.48**	**36.45**	**29.33**	**9.05**	**1.80**	**21.26**	**22.73**
	ITR	36.26	33.59	27.93	8.89	0.02	19.89	21.10
	ACC	0.83	0.81	0.91	0.67	0.35	0.83	0.73
	MDT	1.25	1.25	2.25	2.25	4.25	2.25	2.25
	Pred	1665	1665	1545	1545	1305	1545	1545
Proposed classifier	**ITR$_{\mathrm{MI}}$**	**58.18**	**56.46**	**34.47**	**20.99**	**5.65**	**31.49**	**34.54**
	ITR	56.42	55.26	32.98	20.87	1.74	31.34	33.10
	ACC	0.95	0.95	0.95	0.87	0.55	0.94	0.87
	MDT	1.34	1.34	2.29	2.55	5.12	2.36	2.5
	Pred	976	989	1151	458	164	814	758

Therefore, due to the possibility of predicting “nothing” class, the proposed classifier outperforms Random Forest. If the proposed classifier is not allowed to predict “nothing” and is forced to use rule (5.1) instead, then the performance is similar to Random Forest.

5.3 Comparison to related work

As was the case with Random Forest in the previous section, it is complicated to compare the proposed classifier to the related articles. In most of the articles, it is not mentioned that the classifier can not make a prediction. However, in two of the articles the it is used, namely the article by Demir *et al.* [7] and the article by Jukiewicz and Cysewska-Sobusiak [15].

In the article by Demir *et al.* [7] much longer MDT was used than in this thesis and their achieved ITR was therefore very low. For example, highest accuracy with 4 second MDT is reported to be 73% while in our work, over 80% average accuracy was achieved with MDT of ∼1.5 seconds.

In the work by Jukiewicz and Cysewska-Sobusiak [15] only two classes 8 Hz and 14 Hz were used and the highest reported accuracy is 93% with average over the subjects around 74%. These results were also outperformed in this work. For example, classifying with combination of CCA and PSDA features gave highest accuracy of 95% with average over subjects of 83% as shown in Table 4.5.

The results of the rest of the articles can be seen in Table 3.6. Since in these articles possibility of not predicting was not used, the results cannot be directly compared. However, it is clear that the proposed classifier performs very well and if it does not outperform the classifiers from related work, it very likely performs at similar level.

Conclusion

The aim of this thesis has been to derive a method for optimising thresholds of a classification rule for SSVEP-based BCI. In the first chapter, the biological background and author's previous work was discussed to understand the basics of SSVEP-based BCIs.

The second chapter discussed the feature extraction methods that were implemented in a BCI that is a practical part of this thesis. Widely used PSDA and CCA feature extraction methods were described and in addition to these, more recent methods of LRT, MEC and CWT were presented.

In the third chapter a literature overview was provided to understand what has been done recently in the field and to be able to compare the obtained results to the state of the art.

Fourth chapter is the main contribution of the author. In this chapter, a simple classification rule based on thresholds was presented and in the following sections a method of finding optimal thresholds was derived. Along the way, online estimates of MDT and ITR were derived and a formula for calculating transferred information through mutual information was presented. Finally, after calculating the performance measure and its gradient, a gradient descent algorithm was used to optimise the thresholds.

In the fifth chapter the performance of the proposed classifier was compared to Random Forest and also to the related articles about which an overview was given in Chapter 3.

References

[1] S. F. Anindya, H. H. Rachmat, and E. Sutjiredjeki. A prototype of SSVEP-based BCI for home appliances control. In *2016 1st International Conference on Biomedical Engineering (IBIOMED)*, pages 1–6, 2016.

[2] H. Bakardjiana, T. Tanakaa, and A. Cichocki. Optimization of SSVEP brain responses with application to eight-command Brain-Computer Interface. *Neuroscience Letters*, 469(1):34–38, 2010. http://www.bakardjian.com/work/ssvep_data_Bakardjian.html. (24.04.2017).

[3] G. Bin, X. Gao, Y. Wang, B. Hong, and S. Gao. VEP-based brain-computer interfaces: Time, frequency, and code modulations. *Computational Intelligence Magazine, IEEE*, 4(4):22–26, 2009.

[4] G. Buzsáki, C. A. Anastassiou, and C. Koch. The origin of extracellular fields and currents–EEG, ECoG, LFP and spikes. *Nature Reviews Neuroscience*, 13(6):407–420, 2012.

[5] M. Cheng, X. Gao, S. Gao, and D. Xu. Design and implementation of a brain-computer interface with high transfer rates. *Biomedical Engineering, IEEE Transactions on*, 49(10):1181–1186, 2002.

[6] A. M. Dale and E. Halgren. Spatiotemporal mapping of brain activity by integration of multiple imaging modalities. *Current Opinion in Neurobiology*, 11(2):202—208, 2001.

[7] A. F. Demir, H. Arslan, and I. Uysal. Bio-inspired Filter Banks for SSVEP-based Brain-computer Interfaces. In *2016 IEEE International Conference on Biomedical and Health Informatics (BHI)*, Las Vegas, NV, USA, 2016.

[8] M. Duvinage, T. Castermans, M. Petieau, T. Hoellinger, G. Cheron, and T. Dutoit. Performance of the Emotiv EPOC headset for P300-based applications. *BioMedical Engineering OnLine*, 12(56), 2013.

[9] O. Friman, I. Volosyak, and A. Graser. Multiple channel detection of steady-state visual evoked potentials for brain-computer interfaces. *Biomedical Engineering, IEEE Transactions on*, 54:742–750, 2007.

[10] C. S. Herrmann. Human EEG responses to 1–100 Hz flicker: resonance phenomena in visual cortex and their potential correlation to cognitive phenomena. *Experimental Brain Research*, 137(3-4):346–353, 2001.

[11] M. Hoshiyama and R. Kakigi. Effects of attention on pattern-reversal visual evoked potentials: Foveal field stimulation versus peripheral field stimulation. *Brain Topography*, 13(4):293–298, 2001.

[12] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3-4):321—377, 1936.

[13] F. T. Hvaring and A. H. Ulltveit-Moe. A comparison of visual evoked potential (VEP)-based methods for the low-cost Emotiv EPOC neuroheadset. Master's thesis, Norwegian University of Science and Technology, 2014.

[14] A. Ingel. Control a Robot via VEP Using Emotiv EPOC. Bachelor's Thesis, University of Tartu, 2015.

[15] M. JUKIEWICZ and A. CYSEWSKA-SOBUSIAK. Implementation of Bilinear Separation algorithm as a classification method for SSVEP-based brain-computer interface. *Measurement Automation Monitoring*, 61(2):51–53, 2015.

[16] V. Jurcak, D. Tsuzuki, and I. Dani. 10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems. *NeuroImage*, 34(4):1600–1611, 2007.

[17] V. B. R. Karnati, U. S. G. Verma, J. S. Amerineni, and V. H. Shah. Frequency detection in Medium and High frequency SSVEP based Brain Computer Interface systems by scaling of sine-curve fit amplitudes. In *2014 First International Conference on Networks Soft Computing (ICNSC2014)*, pages 300–303, 2014.

[18] J. D. Kropotov. *Quantitative EEG, Event-Related Potentials and Neurotherapy.* Academic Press, San Diego, 2009.

[19] Y.-P. Lin, Y. Wang, and T.-P. Jung. Assessing the feasibility of online SSVEP decoding in human walking using a consumer EEG headset. *NeuroEngineering and Rehabilitation*, 11(119), 2014.

[20] Z. Lin, C. Zhang, W. Wu, and X. Gao. Frequency Recognition Based on Canonical Correlation Analysis for SSVEP-based BCIs. *Biomedical Engineering*, 54(6), 2007.

[21] Y. Liu, X. Jiang, T. Cao, F. Wan, P. U. Mak, P.-I. Mak, and M. I. Vai. Implementation of SSVEP based BCI with Emotiv EPOC. *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2012 IEEE International Conference*, pages 34–37, 2012.

[22] P. Olejniczak. Neurophysiologic basis of EEG. *Journal of Clinical Neurophysiology*, 23(3):186–189, 2006.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] D. Purves, G. J. Augustine, D. Fitzpatrick, W. C. Hall, A.-S. Lamantia, J. O. McNamara, and S. M. Williams, editors. *Neuroscience.* Sinauer Associates Inc, 3 edition, 2004.

[25] P. Stoica and R. L. Moses. *Spectral Analysis of Signals.* Prentice Hall, 1 edition, 2005.

[26] F. Teng, Y. Chen, A. M. Choong, S. Gustafson, C. Reichley, P. Lawhead, and D. Waddell1. Square or sine: Finding a waveform with high success rate of eliciting SSVEP. *Computational Intelligence and Neuroscience*, 2011, 2011.

[27] L. Tong, R. w. Liu, V. C. Soon, and Y. F. Huang. Indeterminacy and identifiability of blind identification. *IEEE Transactions on Circuits and Systems*, 38(5):499–509, 1991.

[28] *Alzheimer's Disease: Unraveling the Mystery.* U. S. Department of Health and Human Services, National Institutes of Health, 2002. (NIH Publication No. 02-3782).

[29] Y. Velchev, D. Radev, and S. Radeva. Features Extraction Based on Subspace Methods with Application to SSVEP BCI. *International Journal of Emerging Engineering Research and Technology*, 4(1):52–58, 2016.

[30] I. Volosyak. SSVEP-based Bremen-BCI interface–boosting information transfer rates. *Journal of Neural Engineering*, 8(3), 2011.

[31] J. R. Wolpaw, H. Ramoser, D. J. McFarland, and G. Pfurtscheller. EEG-Based Communication: Improved Accuracy by Response Verification. *IEEE Transactions on Rehabilitation Engineering*, 6(3):326–333, 1998.

[32] A. G. Yehia, S. Eldawlatly, and M. Taher. Principal component analysis-based spectral recognition for SSVEP-based Brain-Computer Interfaces. In *2015 Tenth International Conference on Computer Engineering Systems (ICCES)*, pages 410–415, 2015.

[33] W. Yijun, W. Ruiping, G. Xiaorong, and G. Shangkai. Brain-computer interface based on the high-frequency steady-state visual evoked potential. *First International Conference on Neural Interface and Control Proceedings*, 2005:37–39, 2005.

[34] A. Zani, A. M. Proverbio, and M. I. Posner, editors. *The Cognitive Electrophysiology of Mind and Brain.* Academic Press, San Diego, 2003.

[35] Y. Zhang, L. Dong, R. Zhang, D. Yao, Y. Zhang, and P. Xu. An Efficient Frequency Recognition Method Based on Likelihood Ratio Test for SSVEP-Based BCI. *Computational and Mathematical Methods in Medicine*, 2014, 2014.

[36] Z. Zhang, X. Li, and Z. Deng. A CWT-based SSVEP classification method for brain-computer interface system. In *2010 International Conference on Intelligent Control and Information Processing*, pages 43–48, 2010.

[37] D. Zhu, J. Bieger, G. G. Molina, and R. M. Aarts. A survey of stimulation methods used in SSVEP-based BCIs. *Computational Intelligence and Neuroscience*, 2010:1–13, 2010.

[38] B. J. Zier. SSVEP-based brain computer interface using the Emotiv EPOC. Master's thesis, Eastern Washington University, 2012.

Internet URLs were valid on 18.05.2017

Appendices

I Code of the application

The application written as a practical part of this thesis is open-source and the code is accessible from Github repository[1].

[1] https://github.com/kahvel/VEP-BCI

II Acronyms

SSVEP steady-state visual evoked potential 2, 7–14, 16, 18, 20, 21, 24, 25, 31, 33, 36, 37, 57

SVD singular value decomposition 15, 30, 31

SVM support vector machine 25, 26, 28–30

TPR true positive rate 22, 25

VEP visual evoked potential 7, 9, 10, 26, 65

III Glossary

action potential – event of sending a signal by a neuron. 7

axon – nerve fibre through which a neuron sends signals. 7

central visual field – very centre of gaze that is clearly seen. 10

complementary CDF – . 41–43, 46

current dipole – source of the electrical activity that can be measured from the scalp using EEG. 7, 8, 10

dendrite – nerve ending through which a neuron receives signals. 7

detrend – removing the trend from a signal. 12

entropy – . 47

feature extraction – extracting information that can be used for classification from the signal 11–14, 16, 17, 23, 25, 26, 28, 31, 32, 35–37, 41, 49–51

flickering – the blinking or the *state* switches of a target. 9–11, 14, 21, 26

flickering waveform – waveform obtained when plotting the state switches of a target. 11

Fourier transform – transforming a signal from time domain to frequency domain. 12, 13, 19

frequency bin – a value at which the estimated PSD function is defined. 14

frequency component – a *pure tone* that a signal contains. 28, 64

fundamental – frequency component of a signal that has the lowest frequency among all the frequency components. 64

fundamental frequency – frequency of the component of a signal that has the lowest frequency among all the frequency components. 9, 10

harmonic – frequency component of a signal that is an integer multiple of the *fundamental* frequency. 9, 11, 12, 14, 15, 18, 27, 43, 49

interpolation – approximating unknown values between known values. 12

linear combination – for a sequence of values, each value is multiplied by a constant and the result is added up. 18

mean – the measure of central tendency. 12

mother wavelet – function from which the shifted and scaled wavelets are generated from. 19

mutual information – . 39, 40, 45, 57

neuron – nerve cell, main building block of the brain. 7, 10, 64, 65

periodogram – estimate of PSD. 14, 23, 25–27, 32

peripheral vision – not the centre of gaze but the area that surrounds the centre and is not as clearly seen. 10

postsynaptic neuron – a neuron that received a signal through a synapse. 7

pure tone – waveform that contains only one frequency. 64

refresh rate – the number of consecutive images shown on screen in one second. 11, 21

sampling rate – rate at which samples are obtained for example from an EEG device. 8, 14, 15, 21

state – state of a target. 64

synapse – a connection between neurons. 7, 65

target – a visual stimulus of a VEP-based BCI 9–12, 14–16, 18–22, 24, 26, 27, 29, 31–33, 35–38, 41, 43, 49, 64, 65

trend – steady increase or decrease of values in a signal. 12, 64

visual processing centre – part of the brain that is responsible for visual perception. 9, 10

wavelet – oscillating wave-like function with an amplitude that begins and ends at zero. 19, 20

window – a function that is used to smooth the ends of a signal. 12, 26, 27

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Anti Ingel** (16.02.1993)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis
   Machine Learning in VEP-based BCI
   supervised by Ilya Kuzovkin and Raul Vicente

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu 18.05.2017