NTNU – Trondheim
Norwegian University of
Science and Technology

Brain-Computer Interface In Control Systems

Michael Soukup

Master of Science in Cybernetics
Submission date: July 2014
Supervisor: Jan Tommy Gravdahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

A Brain-Computer Interface (BCI) is a system that allows for direct communication between the brain and an external device. Originally, the motivation for developing BCIs has been to provide severely disabled individuals with a basic communication system. Recent years, BCIs directed at regular consumers in practical control applications have gained popularity as well, for which the ultimate goal is to provide a more natural way of communicating with machines.

However, BCIs intended at use in control systems face several challenges and are still inferior to conventional controllers in terms of usability, reliability and practical value. In this thesis, we explore two novel concepts that can enforce BCIs. The first concept relies on detection of so-called Error-Related Potentials (ErrPs), which are the response in brainwaves to an erroneous event. We argue for that these potentials can serve as reward-based signals that give feedback to the system, which enables the BCI to adapt to the user. The second concept is to use sequence labeling frameworks based on Conditional Random Fields (CRFs) to translate brainwaves into control signals with greater accuracy. We also suggest how these two concepts can be combined.

Our experiments to detect ErrPs in BCI control applications using a consumer grade headset to obtain EEG measurements indicate no presence of ErrPs, however, the reliability of the EEG recordings is questionable. Furthermore, we have developed a new implementation of the so-called Sparse Hidden-Dynamic CRF (SHDCRF) and measure its performance on a common BCI classification task. In our experiment, the model outperforms similar classifiers that represent the state-of-the-art, and the results suggest that the proposed model is superior in terms of accuracy and modeling capacity.

Preface

This thesis concludes my Master of Science education at the Norwegian University of Science and Technology (NTNU) carried out at the Department of Engineering Cybernetics. The work presented here is a continuation of the semester project completed fall 2013.

I would like to thank my supervisors Professor Jan Tommy Gravdahl and PhD candidate Serge Gale for valuable support, discussions, ideas and motivation throughout the past year. I am also grateful to the guys at GG48 for providing a great work environment.

Trondheim, 23 July 2014

Michael Soukup

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Background

While traditional communication and control relies on peripheral nerves and efferent motor pathways, a brain-computer interface (BCI) provides an alternate communication channel directly between the brain and an external device. A BCI measures brain activity associated with the user's intent in real-time and translates this activity into corresponding control signals.

Recent years, the motivation for developing BCIs has not only been to provide an alternate communication channel for severely disabled people, but also to use BCIs for communication and control in industrial environments and consumer applications. The ultimate goal is a more natural way of communicating with computers and machines. However, there are major challenges that remain unsolved before BCIs can compete with conventional controllers.

Problem formulation

BCIs aimed at use in control systems needs substantial improvements in terms of signal processing of EEG and utilization of efficient mental protocols. The goal is to enforce the usability of practical BCIs by exploring two novel paradigms, and verify their roles as catalysts for better performance.

Literature survey

A thoroughly literature survey of BCIs in general was completed in our previous work (Soukup (2013)). The purpose of this literature survey was to gain an overview over BCIs in general; how do they work, who use them, which methods is used, and what is state-of-the-art. Graimann et al. (2010) provides an excellent introduction to BCIs and Berger (2008) reviews the current state of different aspects of BCI research. The highly cited paper by Wolpaw et al. (2002) reviews standard BCI approaches and Guger et al. (2013) presents several state-of-the-art BCI publications. The bibliometric study by Hamadicharef (2010) has helped to compile a credible, reliable and trustworthy list of literature.

The literature study concerning the topics in this thesis can be divided in two: The study of error-related EEG potentials and the study of discriminative graphical models for classification.

The main contribution for the study of error-related potentials is the doctoral thesis Dornhege and Millán (2007a), where the phenomenon is described in-depth. The objective of this thesis is also somehow similar to ours as they seek to improve BCI reliability by utilizing these potentials. In addition, the experiment in Iturrate et al. (2010) was the original inspiration for studying this concept.

Sutton and Mccallum (2006) and Sutton (2012) provides an excellent introduction to discriminative graphical models, better known as conditional random fields. The original publications by Lafferty et al. (2001), Quattoni and Wang (2007) and Morency et al. (2007) have played a key role in understanding the different variations and extensions that exist for this type of model. Finally, the work in Shen et al. (2011) have been the main inspiration for our final model.

1.2 Objectives

The main objectives of this thesis are:

1. Perform a study of error-related EEG potentials and conditional random fields.

2. Verify whether these concepts can enhance BCI performance through experiments and realization.

1.3 Limitations

First, BCI research is a young and interdisciplinary field, which requires knowledge about a wide variety of disciplines. There is also a lack of textbooks and study programs which incorporate all these disciplines. Second, the available EEG equipment is not in mint condition.

1.4 Structure of the report

Chapter 2 gives an introduction to BCIs, a description of the necessary components of a BCI, commonly used techniques for processing EEG signals, and typical BCI applications. Note that this chapter is taken directly from our previous work in Soukup (2013), with the exception of some small additions and modifications. In chapter 3, we present our motivation for the two concepts of interest and give the necessary background for understanding them. Finally, in chapter 4 we review the design, implementation, tests and results for the experiments and analyses we have conducted.

Chapter 2

Brain-computer interfaces

(from Soukup (2013), with small modifications)

2.1 Introduction

A brain-computer interface (BCI) is a direct communication pathway between brain and machine, and allows for interaction with external devices using thoughts alone. BCIs rely on direct electrophysiological measures to analyze brain activity in real-time and translate the user's intent into commands. This translation involves signal processing and pattern recognition, and is typically done by a computer. However, BCIs does not decipher brain activity, but relies on neurophysological phenomena to recognize commands. Figure 2.1 shows a typical schematic representation of a BCI, and incorporates signal detection, processing and deployment.

Since BCIs bypasses normal output pathways of peripheral nerves and muscles, they have originally been directed at people with severe neuromuscular disorders, such as amyotrophic lateral sclerosis (ALS), brainstem stroke, and spinal cord injury (Wolpaw et al. (2002)). A BCI can provide such individuals, who may be completely paralyzed, with basic communication capabilities and enable them to use assistive technologies. Common application are spelling devices, neuroprosthetics and control of wheelchairs. Recent years there has been significant effort to develop BCIs aimed at able-bodies as well. BCI-based controllers directed at practical use has already been adapted by computer gaming (Marshall et al. (2013)) and have been used experimentally in many other real-world applications, for instance robotic control. However,

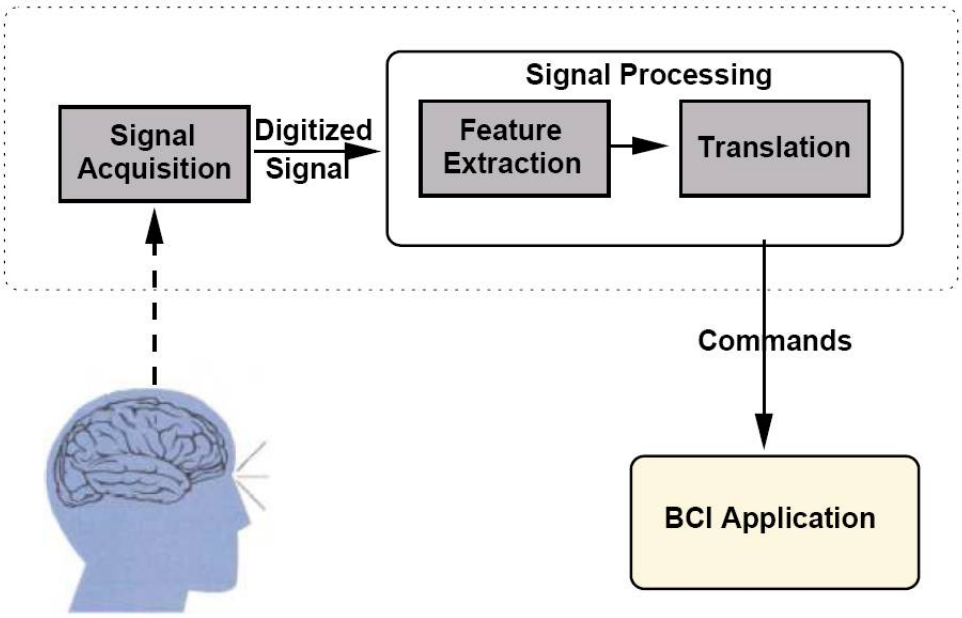


Figure 2.1: Block diagram of a basic BCI system.

current BCIs can only provide unreliable basic communication with low information transfer rates, and are still inferior to conventional controllers in terms of speed, usability and reliability. Additionally, BCI performance varies considerably acoss users; there are even those who cannot use a BCI at all, who are so-called BCI illiterate (Tan and Nijholt, 2010, Chapter 3). This overall lack of capability pose the main challenge of BCIs today.

BCI research is a relatively young and inherently interdisciplinary field, requiring contributions from neuroscience, psychology, medicine, human-computer interaction (HCI), many facets of engineering, and other disciplines (Dunne et al. (2008)). The field has grown substantially the last decade. Guger et al. (2013) reports that "up to the early 2000s, no more than 5 groups where active in BCI research. Now, over 300 laboratories are focused on this work". Indeed, a bibliometric study by Hamadicharef (2010) shows that BCI literature follows a power law growth (Figure 2.2). BCI competition events, where BCI researchers compete to solve BCI related problems, has also been a key factor to draw more attention to the field, and has motivated to new approaches and development of new techniques for signal processing of brain activity (BCI Competitions). It is widely assumed that this field still is to reach its full potential, and as Berger (2008) states: "The ultimate limit to this technology can only be ascertained by

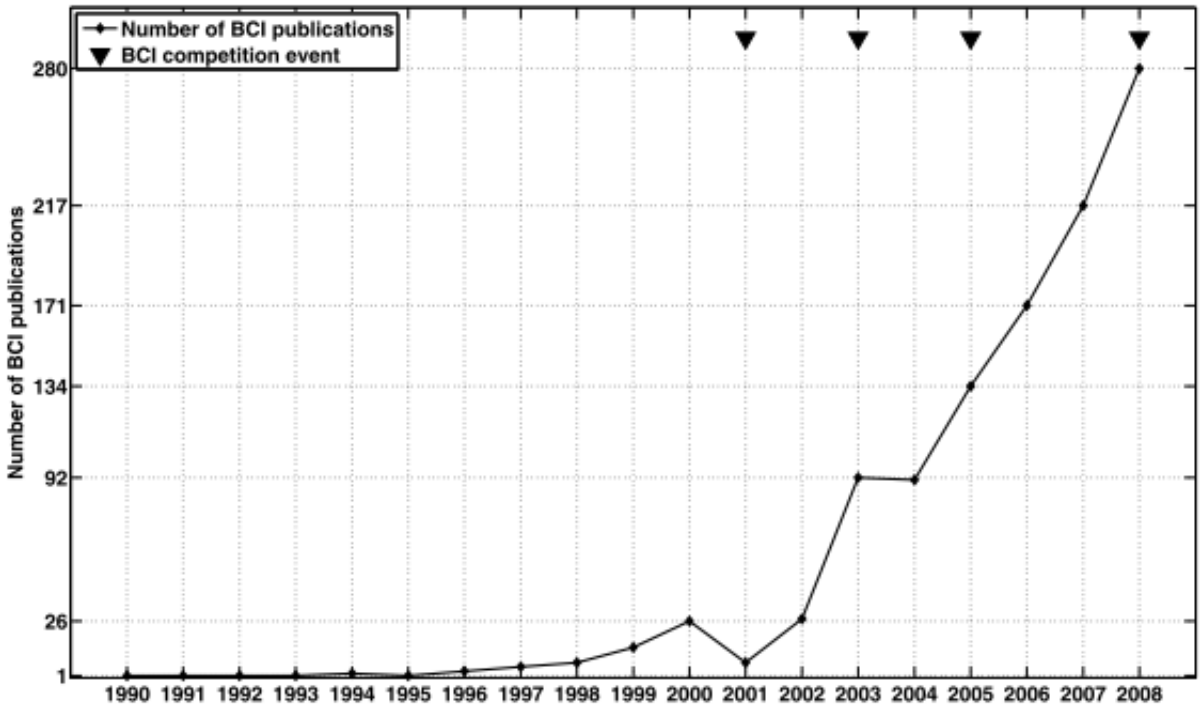


Figure 2.2: Number of published BCI literature per year in the period 1990-2008. (From Hamadicharef (2010)).

research that explores these possibilities."

Many of the earliest BCIs operated in a synchronous, or cue-paced, manner. That is, the user could only initiate a command during a given time frame, often indicated by visual or acoustic cues. While this mode of operation simplifies the detection and classification process, cue-paced BCIs are impractical in many real-world applications. Today, most BCIs operate asynchronously, or self-paced, by interpreting the signals in real-time. This is more convenient since the user can issue commands whenever they want. However, this mode of operation is technically more demanding and puts strict computational requirements on the system. Furthermore, BCIs can be broadly classified into those that rely on either invasive or non-invasive recordings of brain activity. Invasive BCIs use implanted sensors to measure brain activity and requires surgical procedures.

There are several ethical issues with respect to the use of BCIs. Microelectrodes has been implanted in both animals and humans to obtain high-quality recordings of brain activity. Like other invasive medical technology, these procedures are indeed not free of risk. Additionally, BCIs are still in an early stage of development, and invasive BCIs may not even guarantee a

sufficient communication alternative to severe disabled individuals. A common misconception is that BCIs are able to *decipher* brain activity, which have given concerns regarding wire-tapping of thoughts, but current BCIs can only detect and classify specific patterns of activity that are associated with specific tasks or events (Graimann et al. (2010)). However, no one can predict what BCIs will be capable of in the future, and certain panels have already considered future BCI scenarios. For instance, Berger (2008) reports that the Advanced Technology Research Institute in Kyoto already has addressed ethical issues concerned with using BCIs to enhancing normal cognitive function, and the ethical consequences of broadly distributed development and use of BCIs.

2.2 Measuring brain activity

A critical first step of any BCI is to obtain real-time recordings of brain activity. Sensors intended to capture brain activity has been used for clinical purposes for decades, and some are perfectly adaptable to BCIs. To better understand how these sensors work, and exactly what is measured, we must take a closer look at the anatomy and functions of the human brain.

2.2.1 The human brain

The main part of the human brain, the cerebral cortex, is a thick layer of neural tissue that inhabit billions of neurons. Neurons are the basic processing unit in the central nervous system and exist in many different shapes and sizes. Each neuron is a electrically excitable and generates an electrochemical pulse, an action potential, when exchanging information. The action potential is generated by a difference in concentration of ions, including K^+, Na^+, Cl^- and Ca^{2+}, in the membranes' inferior and exterior. The resting potential of

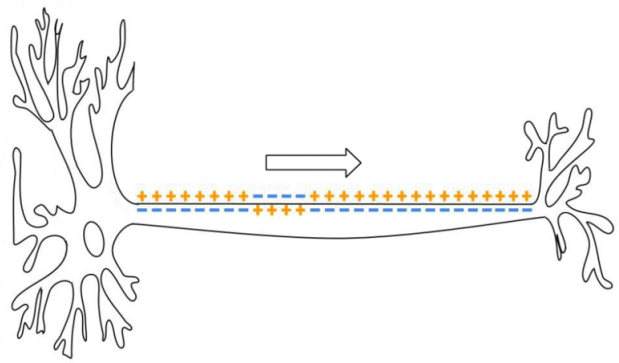


Figure 2.3: Action potential in neurons. (Illustration: Wikipedia)

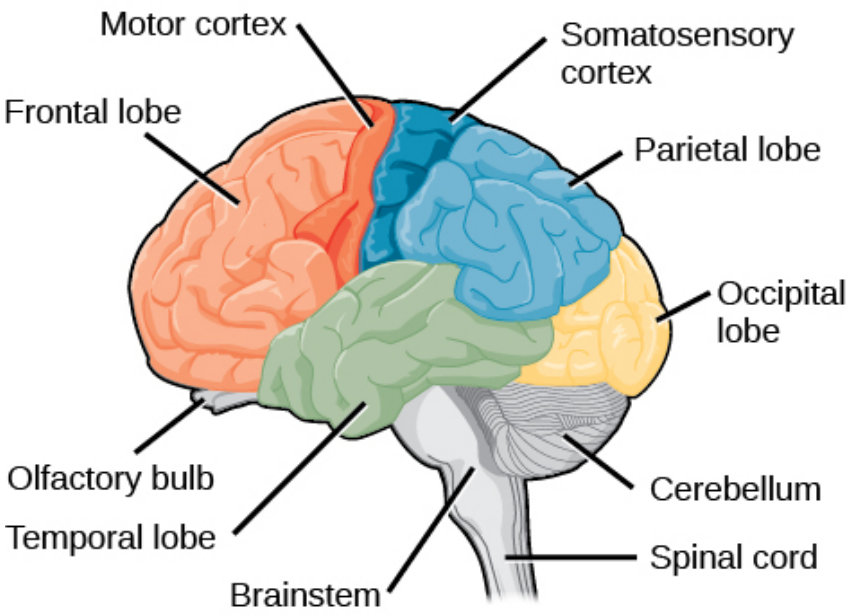


Figure 2.4: Anatomy of the human brain.

a typical neuron is -70 mV, and if the adjacent potential exceeds a threshold potential (around -55 mV), a chain reaction is started, and the potential start to propagate across the membrane, reaching vales up to +100 mV. Figure 2.3 illustrates how the potential propagates through the axon and how the polarity of the membranes' inferior and exterior change.

Neurons group together in so-called neural networks to perform specific human functions, and these networks is typically separated by location. Figure 2.4 illustrates how the cerebral cortex is divided into four main lobes. Each lobe are associated with specific functions and cognitive behavior: The frontal lobe are related to high-level human cognitive behavior like interpretation of language, emotion and personality, planning, problem solving, and voluntary movement of muscles. The occipital lobe is where most visual processing is done, while the temporal lobe is associated with perception and recognition of auditory stimuli, memory, and speech. The parietal lobe handles tasks related to balance, touch and orientation.

2.2.2 Neuroimaging

Recordings of brain activity can be obtained by direct measurements of the electrophysiological signals caused by action potentials, or by monitoring changes in blood flow. The most commonly used are magnetoencephalography (MEG), functional magnetic resonance imaging

(fMRI), near-infrared spectroscopy (NIRS) and electroencephalography (EEG). A short description of each, with focus on how they apply to BCIs follows:

MEG records the magnetic fields induced by electrical activities in the brain. It has high temporal and spatial resolution, but are very sensitive to external magnetic fields. For instance, the earth's natural magnetic field is several magnitudes stronger than MEG. MEG are also very expensive and big devices, which makes them unsuitable for BCIs.

fMRI and NIRS are hemodynamic based techniques, and depends on changes in the blood oxygenation levels. Because of the slow nature of blood flow they have poor temporal resolution, which introduce undesirable delay and poor response time in BCI applications. However, Kanoh et al. (2009) presents a prototype of a NIRS-based BCI.

EEG records fluctuations in the local electric potentials on the surface of the scalp. EEG is very sensitive to disturbances in the electric field and contains a lot of noise, consequently it has low signal-to-noise ratio (SNR) and low spatial resolution. However, EEG has very high temporal resolution. In addition to low cost, high portability and low set-up time, they are well suited for BCIs, and account for signal acquisition in the vast majority of BCI applications.

2.2.3 Electroencephalogram (EEG)

EEG measures the summation of synchronous activity from thousands or even millions of neurons that have the same spatial orientation. It is a very well established method that have been used for clinical purposes for decades. Because scalp-recorded EEG equipment is lightweight, inexpensive and easy to apply, it provides the most practical non-invasive access to brain activity.

However, non-invasive EEG is not without disadvantages: Since the signal is damped by layers of bone and skin tissue, scalp-recorded EEG is very weak, and ranges from 10 μV to 100 μV for adults. This makes it susceptible to so-called artifacts, which are contaminations caused by other electrical activities, and results in a low signal-to-noise ratio (SNR = $\frac{P_{signal}}{P_{noise}}$). Examples are electromyographic (EMG) activity from muscle movements or electrooculographic (EOG) activ-

ity caused by eye movements. External electromagnetic sources like power lines and electrical equipment may also distort the signal.

Furthermore, most EEG electrodes requires that a conductive solution is applied to achieve adequate signal quality. This procedure can be time-consuming and cumbersome. So-called dry electrodes for EEG are gaining attention (Dunne et al. (2008)), but can still not compete with wet electrodes in terms of signal quality. Dry electrodes has nonetheless recently been applied in BCIs, for instance in the work done by Mladenov (2012).

Intracranial EEG, also known as electrocorticography (ECoG), use electrodes placed directly on the exposed surface of the brain to record cerebral activity. Consequently, this requires an invasive procedure. Since the electric field is not damped by the scalp it provides considerable better signal quality, with amplitudes ranging from 10-20 mV, and is protected by external contaminations which results in a higher SNR.

There is still room for improving EEG electrodes. Berger (2008) points out that "...this sensor technology has experienced limited growth and needs substantial improvement".

International 10-20 system

It is crucial that EEG electrodes have the same spatial orientation across usage, because a BCI system depends on recognition of repeatable mental activity. Therefore, EEG-based BCIs rely on the international 10-20 system, which are an international standard for placing electrodes. Figure 2.5 shows how this system divides the scalp into multiple sections and apply labels that corresponds to different lobes of the cerebral cortex. Commonly used electrodes are positioned 10, 20, 20, 20, 20 and 10% of the total nasion-inion distance, which makes the system more tolerant for across-subject usage.

2.2.4 Mental activity observable with EEG

Certain cognitive actions and responses to external and internal stimuli can be observed with EEG. These phenomena has particular characteristics in the time- and frequency domain, and are used clinically to diagnose patients with brain related issues. To review all of them is outside the scope of this report, we will restrict ourself to the most commonly utilized in BCIs.

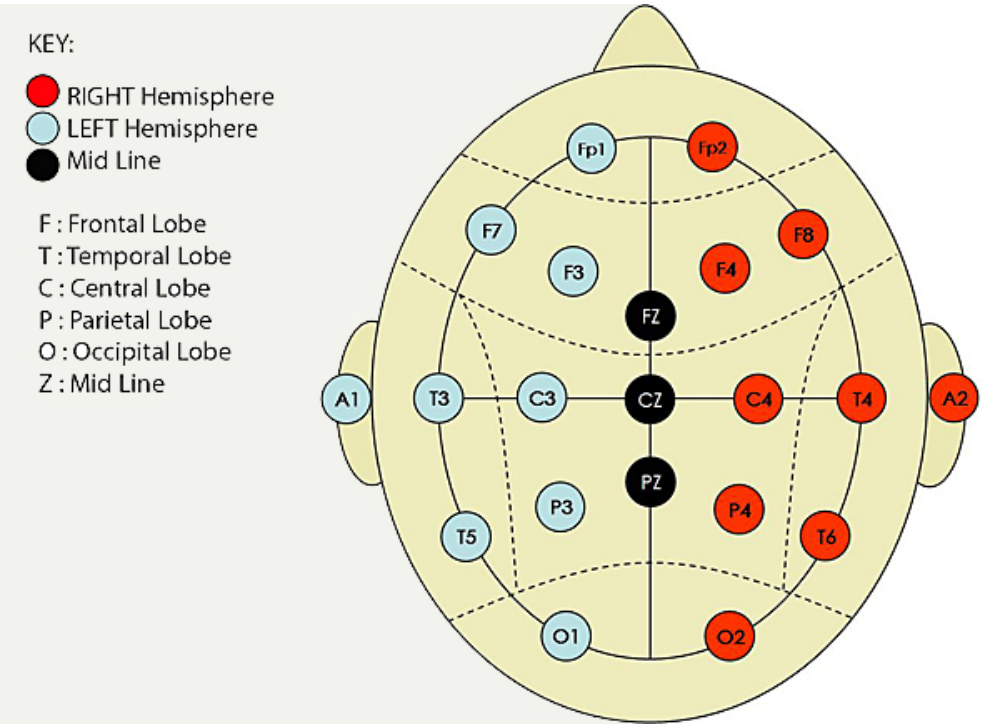


Figure 2.5: International 10-20 system for placing EEG electrodes.

Event-related potentials (ERPs)

Event-related potentials are changes in voltage which are time-locked to a particular event or stimulus. These voltage changes can be observed as a series of positive-negative deflection in the EEG which are distinguished by their scalp location and latencies. Latencies <100 ms originate primarily in the main sensory cortices and can be determined by the properties of the evoking stimulus. Later ERP components, with latencies 100-500 ms, originate in cortical areas associated with more complex brain processes (for instance the frontal lobe), and they are more variable in form and latency (Wolpaw and Boulay (2010)). The longest latency ERPs are called slow cortical potentials (SCP) which have latencies up to several seconds, even minutes in some cases, and often reflect response-oriented brain activity.

Visual evoked potentials (VEPs) are the most extensively studied ERP (Graimann et al. (2010)). VEPs involve at least three successive features that occur in the first several hundred milliseconds after a visual stimulus, and the polarities, latencies, and cortical origins of the components vary with the stimulus presented.

Table 2.1: Frequency range and mental activity of brain rhythms.

Band	Frequency range [Hz]	Mental activity
Delta (δ)	< 4	Slow-wave (deep) sleep.
Theta (θ)	4 - 7	Idling, unconscious, meditation and drowsiness.
Alpha (α)	7 - 12	Relaxation and concentration.
Mu (μ)	8 - 13	Suppression indicates that motor neurons are working.
Beta (β)	12 - 30	Alert, thinking and active concentration.
Gamma (γ)	> 30	Somatosensory processing.

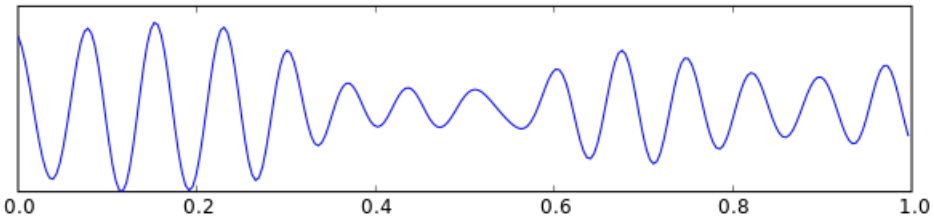


Figure 2.6: ERD and ERS in the sensory motor rhythm.

Cortical oscillations

Brain activity is reflected in the frequency domain as a variety of oscillations. So-called brain rhythms are categorized into bands that corresponds to certain mental activities. Table 2.1 presents the most common way to label frequency bands and describes their respective associated mental activities.

The μ-band is somewhat analogous to the so-called sensory motor rhythm (SMR). The SMR partially overlaps with multiple bands, but can primarily be regarded as alpha- and beta-activity in motor cortical areas. Figure 2.6 shows how the SMR change during movement of a limb. Preparation of movement causes an event-related desynchronization (ERD) which is accompanied by a decrease in the SMR. When relaxation after the movement occur the SMR increases to resting state as a result of an event-related synchronization (ERS).

2.3 Producing brain signals

Measuring activity is not enough, because a BCI cannot read the mind or decipher thoughts in general. Therefore, most BCIs employ either ERPs or cortical oscillations as mental strategies for producing command signals. The mental strategies can be distinguished by the dependency

on external stimuli.

2.3.1 Selective attention

Selective attention relies on external stimuli. The most common stimuli are visual, but it can also be auditory or somatosensory. In order to select a command, the user has to focus their attention to the corresponding stimuli. BCIs based on selective attention has shown to provide fairly reliable and rapid communication, and works across subjects without extensive training. However, they require introduction of external stimuli, which are impractical in most real-world applications. In addition, the presented stimuli may be exhausting for the user over long time use. The two most commonly used selective attention strategies are presented here.

P300

Presentation of infrequent or surprising task-relevant stimuli elicits a positive deflection in the EEG after about 300 ms. This potential is called the P300 evoked potential. A P300-based BCI system presents an array of stimuli, each of which represents a particular output. The attended stimuli elicits a P300 and the other stimuli do not. Figure 2.7 shows the nature of this ERP, and the resulting increase in signal intensity.

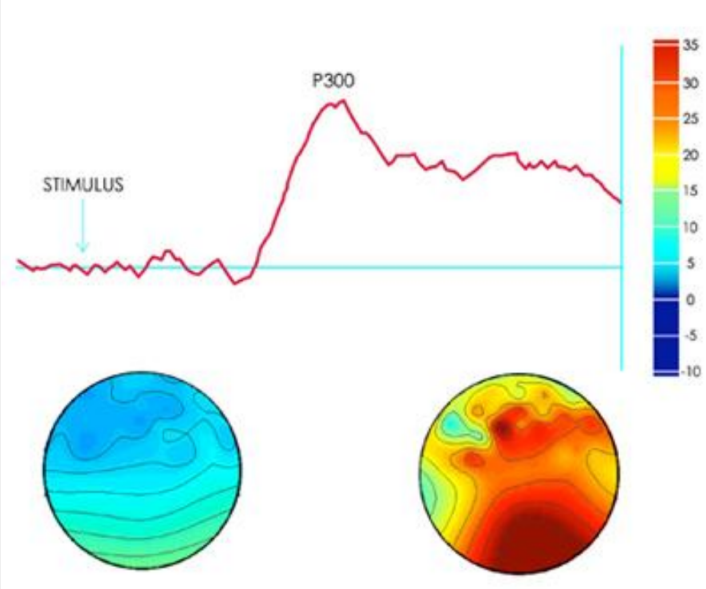


Figure 2.7: The P300 evoked potential.

SSVEP

Steady-state visual potentials (VEPs) are special VEPs that is elicited by a repetitive visual stimulus. Similar to the P300 it depends on presenting an array of selectable stimuli to the user. Attention to a flickering stimuli elicits an SSVEP in the visual cortex that has the same fre-

quency as the flickering (typically 6-30 Hz). SSVEP depends mainly on the properties of the visual stimulus and is especially resistant to variation across trials and subjects.

2.3.2 Mental imagery

Mental tasks that originate endogenously, and may be initiated without depending on external stimuli, are particularly appealing in asynchronous BCIs. A wide variety of mental tasks is observable with EEG, for instance imagined cube rotation, word association, auditory recall or calculations (Sepulveda et al. (2007)).

Motor imagery

However, the vastly most popular mental strategy for BCIs is motor imagery. That is, the imagined movement of a limb, usually an arm or both feet. The preparation of movement elicits a response in the EEG very similar to the response that resulting from *actual* movement of limbs.

As illustrated in Figure 2.6, this response can be observed as an ERD and ERS in the SMR. ERD and ERS patterns follow a homuncular organization: Activity invoked by right hand movement imagery is most prominent over C3. Activity invoked by left hand movement imagery is most prominent over C4. Patterns which originate from the same cortical area is hard to discriminate. Coordination of both feet are handled in the same cortical area, therefore it is hard to discriminate between the left and right foot. Movement of each hand originate from opposite hemispheres, and are better suited to use as two distinct control signals for BCIs.

2.4 Processing brain signals

Signal processing is a vital part of any BCI system and usually happens in three consecutive stages:

1. Preprocessing
2. Feature extraction
3. Detection and classification

These stages must be executed in real-time if the BCI in question is to operate asynchronously. Signal processing is without doubt the biggest challenge in BCI research, and the available literature reviews numerous methods which concerns all three stages (Lotte et al. (2007); Bashashati and Fatourechi (2007); Liu et al. (2013); Mansor and Khuan (2010); Cinar and Sahin (2010); Khorshidtalab and Salami (2011)). In this section we review the most important algorithms and mathematical tools in each stage.

2.4.1 Preprocessing

Preprocessing prepares raw EEG for subsequent operations and aims at improving the quality of the signal without losing relevant information. In other words, we want to filter out unwanted information to improve the SNR. In some cases the signal is analogue and needs to be sampled by a A/D-converter, but this is usually handled by embedded hardware. The signal can be filtered with respect to two main domains: Time and orientation.

Temporal filtering

Since mental tasks occur in specific frequency bands, it is advantageous to apply bandpass filters to suppress the amplitude of irrelevant frequencies. For instance, motor imagery-based BCIs commonly bandpass the μ- and β-bands. The infinite impulse response (IIR) filter and finite impulse response filter (FIR) are most commonly used for this purpose.

A digital filter can be expressed as a transfer function $H(z)$ that operates in the z-domain:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}} \tag{2.1}$$

where a_i and b_j are the filter coefficients. The order of the filter depends on the number of non-zero coefficients and is given by $\max\{N, M\}$. The impulse response h_k is the filters response to the delta-function and characterizes the filters behavior. As the name suggest, the impulse response of IIR filters may be infinite and h_k does not necessarily converge to zero, whereas for FIR filters h_k becomes zero at time $t \geq T$ for some finite T.

The Butterworth filter, which have a maximally flat impulse response, and Notch filters, which filter out a specific frequency (for instance 50 and 60 Hz power line disturbances), are

also commonly used in BCIs.

Spatial filtering

There is a wide variety of spatial filtering techniques available, which aims at removing common components of the signal or other disturbances with spatial characteristics. Here, we summarize the procedure for the three most common spatial filters:

Common spatial patterns (CSPs) is a supervised method that optimizes the maximum difference between channels and separates the EEG signal into several virtual channels.

Principle component analysis (PCA) or independent component analysis (ICA) transform the signal orthogonally into a set of linearly independent principle components.

Surface Laplacian subtracts the contribution of neighboring channels.

2.4.2 Feature extraction

Feature extraction is the process of computing representative characteristics of the data that are relevant to the task at hand. The set of features that are extracted constitute a feature vector, which are commonly passed to a classifier.

It is of high interest to find good features that describe the signal well, to reduce computational expense and the dimensionality of the feature space. Suppose we draw d features from a k-channel EEG source, then the resulting feature vector would reside in $\mathbb{R}^{d \cdot k}$ and grow exponentially as we draw more features, which makes the available data very sparse. Also, as the dimensionality arise, many various phenomena occur. This is collectively referred to as the "curse of dimensionality" and pose a major concern in BCIs (Lotte et al. (2007)). Some BCIs try to solve this by applying dimensionality reduction techniques. For instance, PCA is also used to maximize the independence of features.

Numerous methods and mathematical transformations are available to draw properties from the signal, however, we have to restrict ourself to a small subset that is commonly used in BCI applications. For convenience, we assume a continuous single-channel EEG signal $x(t)$, $t \in [0, T]$ throughout this section.

Average absolute value (AAV)

The average absolute value is a trivial, yet useful feature, given by:

$$AAV = \frac{1}{T}\int_0^T |x(t)|dt \tag{2.2}$$

Power spectral density (PSD)

The power spectral density (PSD) describes how much of the power is carried by a specific frequency. The Fourier transform expresses our signal in the frequency domain:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi i f t}dt \tag{2.3}$$

However, $x(t)$ is not infinite, so we consider the signal only over observerd time T:

$$X_T(f) = \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)e^{-2\pi i f t}dt \tag{2.4}$$

Then we can define the PSD as the Fourier transform of the autocorrelation sequence of $x(t)$:

$$S_x(f) = \lim_{T\to\infty}\frac{1}{T}E\left[|X_T(f)|^2\right] \tag{2.5}$$

Band power (BP)

By integrating equation 2.5 we obtain the total power in a certain frequency range:

$$P_{f_1,f_2} = \int_{f_1}^{f_2} S_x(f)df \tag{2.6}$$

PSD and BP relates directly to cortical oscillations and SSVEP in EEG, and are by far the most popular feature in BCIs. According to Bashashati and Fatourechi (2007) they are used in 41% of SMR-based BCIs. Other popular features includes wavelets, autoregressive parameters and higher-order statistical properties of the signal.

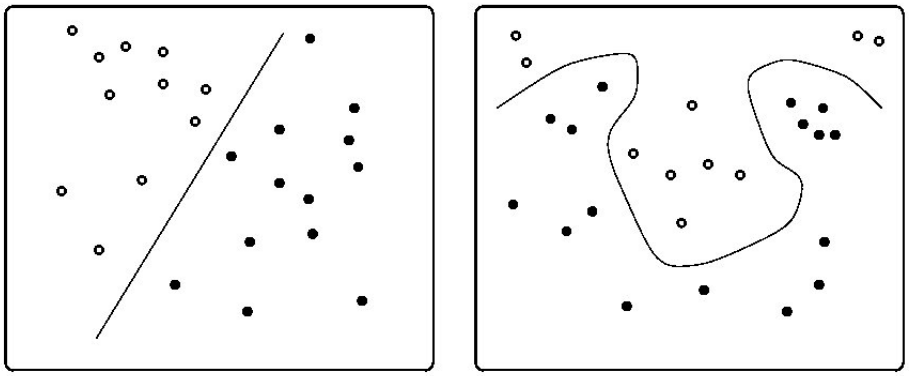


Figure 2.8: Linear classification (left) and nonlinear classification (right).

2.4.3 Classification

Classification, or pattern recognition, is the practice of discriminating between different groups of data. In our case, we use classification to translate the feature vector which carries the characteristics of the EEG, to an associated group of data. There are three main approaches to modeling:

Fixed models is used when the exact input-output relation is known, for instance a known threshold value.

Parametric models use training data or *á priori* information to tune the parameters of the model to fit the data.

Non-parametric models or clustering techniques is suited when the relationship between data and classes is not well understood.

A BCI typically use a parametric model, or a mix of parametric and non-parametric models, and rely on user specific training to tune the classifier.

General linear classifiers

Consider a two-class case with classes ω_1 and ω_2 and a feature vector $z = [z_1, z_2, ..., z_l]^T$. A linear classifier maps z to either ω_1 or ω_2 by a linear discriminant function $g : \mathbb{R}^l \rightarrow \mathbb{R}$ defined as

$$g(z) = w^T \cdot z + w_0 \tag{2.7}$$

where $w = [w_1, w_2, ..., w_l]^T$ is a weight vector and w_0 is a threshold. The decision rule becomes:

Choose ω_1 if $g(z) > 0$, else ω_2.

The left part of Figure 2.8 shows the decision line in the case where $l = 2$. In a multi-class case with c classes, we need one discriminant function for each class:

$$g_i(z) = w_i^T \cdot z + w_{i0} \tag{2.8}$$

where $i = 1, 2, ..., c$, and we get the decision rule:

Choose ω_m if $g_m(z) = \max_i \{g_i(z)\}$.

Furthermore, there are broadly two types of methods for determining the linear discriminator w.

Probabilistic linear models

Probabilistic models estimate the conditional density functions $P(z|\omega_i)$ with known probability distributions. The naive Bayes classifier take base in Bayes' rule

$$P(\omega_i|z) = \frac{P(z|\omega_i)P(\omega_i)}{P(z)}. \tag{2.9}$$

Expressed in terms of $P(z)$, we get

$$P(z) = \sum_{i=1}^{c} \frac{P(z|\omega_i)P(\omega_i)}{P(\omega_i|z)}. \tag{2.10}$$

Since $\sum_1^c P(\omega_i|z) = 1$ this reduces to

$$P(z) = \sum_{i=1}^{c} P(z|\omega_i)P(\omega_i) \tag{2.11}$$

and so the probability density function of the feature vector z can be estimated according to assumptions on the distribution for each class. For discrete features, the multinomial and Bernoulli distributions are popular, while a Gaussian distribution is a fair assumption for continuous data like EEG.

Linear discriminant analysis (LDA) is another probabilistic model that are very popular in BCIs. LDA looks for the best linear combination of variables by assuming that $P(z|\omega_i)$ is normally distributed for all classes i, and estimates mean and covariance parameters μ_i and $\sum_i$ from the training data. The resulting decision boundary is a linear hyperplane. In the special case where all covariance parameters are equal ($\sum_i = \sum$), it is called quadratic discriminant analysis (QDA).

Discriminative linear models

Discriminative models estimate the vector w by maximizing the given criteria on the training data. Linear regression is an example of this; it attempts to minimize the total distance from the regression line to each point.

The support vector machine (SVM) compares only two classes at a time. Given training data with each point z_i belonging to either $\omega_i = 1$ or $\omega_i = -1$, the SVM formulates the training procedure as an optimization problem:

Minimize $||w||_2$,

such that for any $k = 1, ..., n$

$$\omega_i(w^T \cdot z - w_0) \geq 1. \qquad (2.12)$$

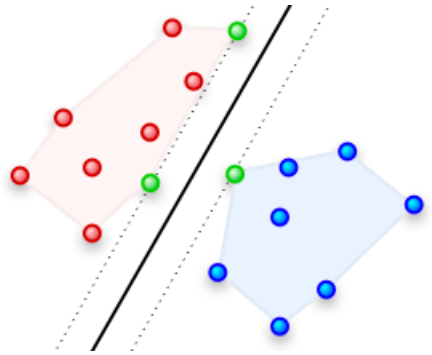


Figure 2.9: Prinicples of the SVM.

Figure 2.9 shows the resulting maximum-margin hyperplane, given by $w^T \cdot z - w_0 = 0$, which separates the blue and red dots, and the margin hyperplanes given by $w^T \cdot z - w_0 = \pm 1$. The green samples on the margin are called support vectors. SVM-based classifiers are very popular in BCI application, and has performed well with high accuracy rates (Cinar and Sahin (2010); Khorshidtalab and Salami (2011); Lotte et al. (2007)).

Nonlinear parametric models

Nonlinear, parametric classifiers are when the discriminative function is not linear as shown in Figure 2.8. These often build upon expanding linear cases. For instance, the perceptron, which is a discriminative linear classifier similar to SVMs, acts as an artificial neuron in artificial neural networks (ANN), and may also be expanded to the nonlinear multilayer perceptron. SVMs can also be generalized to a nonlinear model.

Non-parametric models

Commonly referred to as clustering algorithms, these types of models find hidden structures in unlabeled data. The most popular is the k-means method, which aims to partition n observations into k classes or clusters. Given training data z_n where each $z_i \in \mathbb{R}^l$, cluster into k sets $S = \{S_1, S_2, ..., S_k\}$. Then minimize the cumulative l_2-norm with respect to S:

$$\min_S \sum_{i=1}^{k} \sum_{z_j \in S_i} ||z_j - m_i||^2 \tag{2.13}$$

where m_i is the mean of all points in S_i.

Sequential models

In many real-world cases, the data can be expressed as a function of time and contains temporal dependencies. Sequence models does not only learn a mapping between features and labels, but also model the transitions between features. These models may rely on the presentation of correct input-output pairs (parametric methods) as well as finding hidden structure in the data (non-parametric methods). This is the case for one of the most popular sequence models, namely the hidden Markov model (HMM).

Suppose the features z and classes ω now are represented as stochastic time series, such that $z = \{z_t\}, z_t \in Z$ and $\omega = \{\omega_t\}, \omega_t \in \Omega$. Then the HMM includes an initial distribution $p(\omega_1)$, a transition distribution $p(\omega_t|\omega_{t-1})$ and a observation distribution $p(z_t|\omega_t)$. Together, they define

the joint probability distribution

$$P(\boldsymbol{\omega}, \boldsymbol{z}) = \prod_{t=1}^{T} P(z_t|\omega_t)P(\omega_t|\omega_{t-1}) \tag{2.14}$$

where the initial state distribution $p(\omega_1)$ is written as $p(\omega_1|y_0)$. Sequence models will be more closely examined in chapter 3.

Reinforcement learning

Reinforcement learning models are a mix of parametric and non-parametric models. It differs from standard parametric models because it does not only rely on *á priori* information to tune the classifier, but also on *á posteriori* information to update the classifier during operation. That is, these models are presented frequently with so-called reward-based signals, which is used as a measure for the input-output relationship. Thus reinforcement learning also rely on finding structures in the data that are hidden during training, and optimizing (learning) these structures better over time. Reinforcement learning algorithms often rely on exploration techniques in order to learn and adapt to the task at hand.

These characteristics makes them very general, and they may incorporate many aspects and methods from the previously presented classifiers. Some of them even incorporate feature extraction. Reinforcement learning techniques will also be revisited in chapter 3.

2.5 BCI performance

To analyze the performance of BCIs, some metric must be applied. The most commonly used evaluation criteria is accuracy (ACC), defined as

$$ACC = \frac{1}{N}\sum_{i=1}^{M} n_i \tag{2.15}$$

where n_i denotes the number of correctly classified samples within class i, and M is the number of classes. Another popular measure is the error rate ($ERR = 1 - ACC$).

Confusion matrices describes the relationship between the true and predicted labels. The diagonal elements represent correctly classified samples and off-diagonal elements show the

incorrectly labeled samples. This gives a clear and descriptive view of the error distribution.

Information transfer rate (ITR) is the communication capacity of a BCI, measured in bits per second. Different approaches for calculating the ITR is reviewed by Dornhege and Millán (2007b).

It is important to note that the vast majority of studies applies these metrics on off-line analyses. However, the statistics of EEG change with the introduction of feedback Shenoy et al. (2006), which indicates that the effect of real-time performance also should be considered.

2.6 Example applications

To better grasp what BCIs are capable of, some example application with focus on use in control systems are presented here. This also serves as a timeline of events with the purpose to summarize the development recent years.

In 2004, Wolpaw and McFarland (2004) disproved the widely assumed claim that multidimensional control of a robotic manipulator or a neuroprosthesis only is possible with invasive BCIs. Their non-invasive BCI, based on scalp-recorded EEG, enabled subjects to perform two-dimensional cursor control using motor imagery, and achieved results comparable with those of invasive BCIs in terms of movement time, precision and accuracy. The translation algorithm in this setup employed an adaptive algorithm that weighted activity from either the mu-rhythm band or the beta-rhythm band, based on the individual subject's performance.

Experiments with invasive BCIs may suggest what will be possible also with non-invasive BCIs, as many research groups believe that the quality of scalp-recorded EEG, even with dry electrodes, will catch up to the quality of implanted electrodes in the near future.

Hochberg et al. (2006) presents initial results for a tetraplegic human using a pilot neuromotor prosthesis. Neuronal ensemble activity was recorded using a 96-microelectrode array implanted in the primary motor cortex of the patient, and was translated to command signals by applying a linear filter to spiking activity of motor imagery actions. The patient was able to perform high precision control of a computer cursor, which was demonstrated by opening an e-mail application and drawing a circle on the computer screen. The neuromotor prosthesis was also able to provide control of a multi-jointed robotic arm, using it to grasp an object and

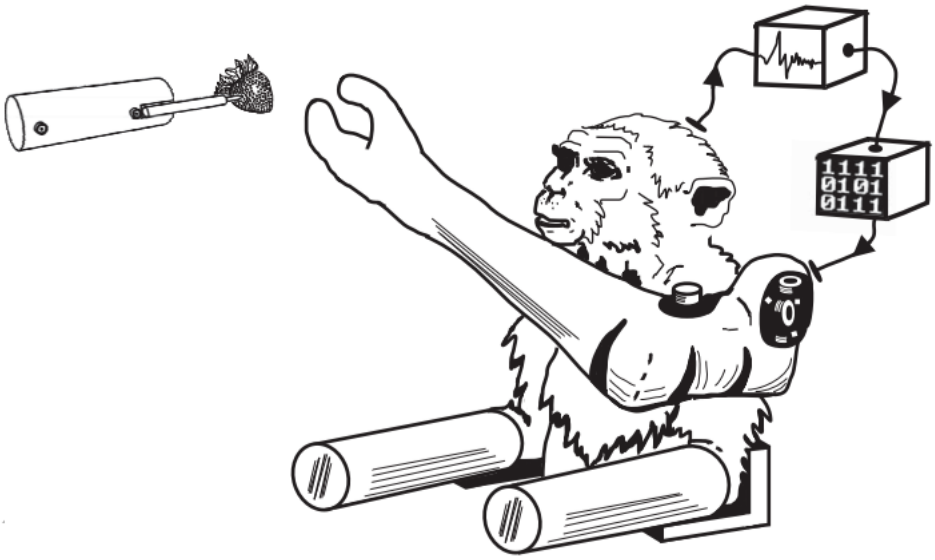


Figure 2.10: Concept illustration of the experiment conducted in Velliste et al. (2008).

transport it from one location to another.

In a similar setup by Velliste et al. (2008), which used monkeys as test subjects, intracortical recordings was used to control a robotic arm with a gripper attached at the end. Figure 2.10 illustrates how the experiment was carried out. The monkeys controlled the manipulator through the workspace in three dimensions, along with proportional control of the gripper, to perform a self-feeding task. The study suggests that BCIs could be suited as a control device for prosthetics and ultimately achieve arm and hand functions on a near-natural level. ref Figure 2.10

Recently, LaFleur et al. (2013) received extensive media attention for their demonstration of three-dimensional control of a quadcopter using a motor imagery-based non-invasive BCI. Human subjects navigated the quadcopter through a obstacle course and successfully acquired up to 90.5% of all valid targets through the course.

The presented studies strongly indicates the potential for accomplishing complex control in a 3D physical space, which in turn can be adopted by people with severe disabilities but also in industrial environments.

Even though the field of BCI research have focused mostly on control of robots, manipulators or cursors, some studies has shown the potential of BCIs in other applications. For example, Brumberg et al. (2013) uses intracortical microelectrodes to predict intended speech information directly from the activity of neurons involved in speech production, which improved sub-

jects control over an artificial speech synthesizer.

Chapter 3

Novel concepts to improve BCI performance

The field of BCI research is still in its early days, and the majority of BCI applications have been constructed and tested only in controlled laboratory settings. Many believe, however, that BCIs will increasingly be available to consumers for the years to come, and serve as control solutions for people such as disabled, prosthetic users, and gamers (Graimann et al. (2010)).

Indeed, with commercially available equipment and software on the rise, BCI applications are moving out of the laboratory environment. However, there are still many fundamental challenges that remains unsolved before regular consumers can adapt to BCIs. One major challenge is the necessary advancements in hardware, especially development of better dry electrode technology (Berger (2008)). There are also numerous challenges that reside in the software domain. Some of them, along with a selection of publications that address them, includes:

- Reduce training time and move towards "plug-and-play" functionality (Faller et al. (2012)).
- Adapt to temporal changes in the EEG (Yong et al. (2012)).
- Account for subject-specific performance variations (Grosse-wentrup and Sch (2013)).
- Employing intuitive and effective mental strategies (Blakely et al. (2013))

There are two common factors that summarize the main concern of most challenges: To increase the information transfer rate without excessive subject-specific training and customization, and make the system adapt to each user. Ultimately, the role of a BCI is to recognize the user's intent and perform actions accordingly.

These two factors primarily implies to improve the signal processing, or more specifically, the classifier of the system. The design of better classifiers and adaptive frameworks is indeed one of the main topics in many popular publications (Wolpaw et al. (2002); Shenoy et al. (2006); Yang and Fang (2007); Zou et al. (2010); Tan and Nijholt (2010); Yong et al. (2012); Oliver et al. (2012); Kus et al. (2012)), but the vast majority of these methods rely on squeezing additional functionality out of standardized paradigms (Soukup (2013)).

Objectives

Our goal is to explore new paradigms that can enable higher performance, usability and reliability in BCIs. We focus on asynchronous non-invasive EEG-based BCIs, and consider the BCI to be used in a control context.

More specifically, we seek to improve the classifier by addressing two novel concepts that were identified in our previous work: Detection of reward-based EEG components to make the classifier adaptive, and implement a framework that can account for complex temporal relationships in EEG.

3.1 Previous work

In Soukup (2013), we presented an evaluation of existing methods used in BCI systems and the most popular approaches to implement each step in Figure 2.1. We attempt to identify bottlenecks in the current trends and turn our attention to unexplored, but promising new paradigms, that may serve as catalysts for BCI performance.

First, we point out that a neuro-physiological phenomenon called error-related potentials (ErrPs), which are components in the EEG that are elicited when the user observes some sort of error, could serve as feedback in BCI applications. As a result, the system can adapt to the user, and reduce training time while enhancing performance. We looked more closely into the experiment described in Iturrate et al. (2010) and suggested a experimental setup that is more suitable for control applications.

Furthermore, we argue for that most of the commonly used classifiers does not utilize the full potential of the EEG signal, because they fail to model the statistical dependencies that exist

across time-windows. We construct a Finite State Machine (FSM) that provides the necessary structure to model temporal changes and expand it into an Hidden Markov Model (HMM). We conclude that the Markov property is an inappropriate assumption for EEG signals and suggest another probabilistic framework called Conditional Random Fields (CRF) as a new basis for our idealistic model. We then review the results from Saa and Çetin (2012), which achieves superior scores on a standard BCI classification task using the latent-dynamic CRF model (LDCRF).

This chapter extends the main findings in our previous work and describes these concepts in-depth.

3.2 Error-related potentials (ErrPs)

An error-related potential is a specific component of the EEG signal that is elicited as a result of an erroneous event, and is thus a subset of the ERPs described in section 2.2.4. The potential itself is generated by neural activity that originate from error processing mechanisms in the human brain. Only a few studies report of these potentials until the early 1990s. After that, several studies have described the presence of ErrP components such as error-related negativity and error-related positivity in many different situations (Dornhege and Millán (2007a)).

ErrPs can be divided into four main types, where each type is classified by the event that stimulate them.

Response ErrPs is the most described ErrP (Figure 3.1), and occurs in choice reaction tasks where the subject is asked to respond as fast as possible to a certain stimulus, typically by pressing a button or key. In the case of an error, the two main components of this ErrP are recognized as a sharp negative component peaking after ~100 ms and a broader positive potential with a peak between 200 and 500 ms. The negative potential is most prominent over the fronto-central lobe, while the positive potential has a centro-parietal maximum.

Feedback ErrPs is associated with reinforcement learning tasks. A system choose among several possible outcomes, and the subject is asked to favor one of them and also adapt their strategy to minimize the error. In the cases where the system choose wrong outcomes, the ErrP is elicited as a negative deflection that occurs ~250 ms after the feedback.

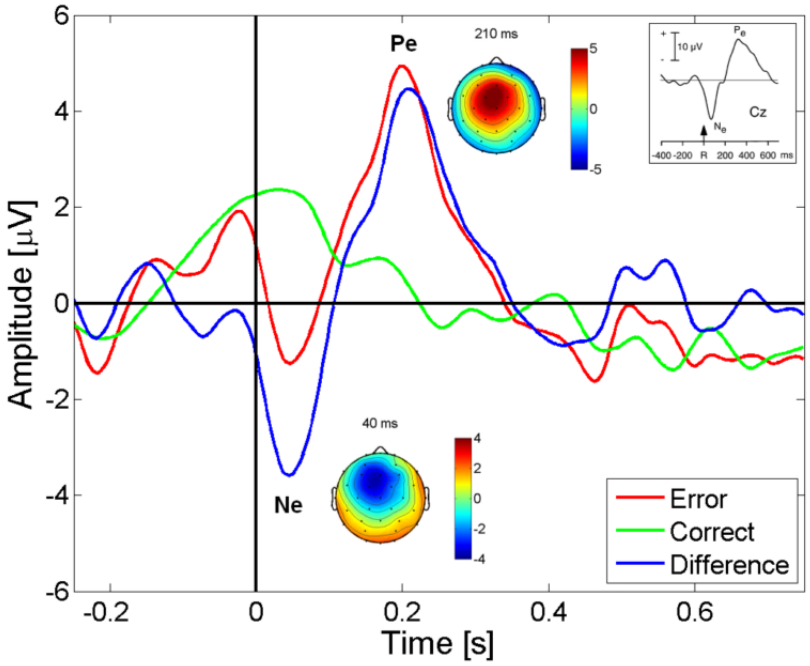


Figure 3.1: Characteristics of the response ErrP. The samples are produced by taking grand averages of EEG trials during a choice reaction task (from Dornhege and Millán (2007a)).

Observation ErrPs is elicited when the subject monitors an operator that execute a task, typically a choice reaction task. The potential has characteristics similar to the feedback ErrP.

Interaction ErrPs was first described in Dornhege and Millán (2007a), and is elicited in setups similar to the response ErrP, only that the error is performed by the interface, not the subject. The general shape of these ErrPs are quite similar to the shape of the response ErrP, but the interaction ErrP also have a second negative broad deflection. In addition, the courses and amplitudes of this ErrP vary more among subjects.

As Figure 3.1 implies, these potentials have amplitude of magnitude μV and are not easy to detect. However, ErrPs have been detected on single trials with success rates up to 80% and are reportedly very stable over time (Dornhege and Millán (2007a)). Finally, the spatial location where the potential appears the strongest is around the fronto-central and centro-paretial lobes for all types of ErrPs that have been reported so far.

Figure 3.2: Experimental setup for detecting error-related potentials in Iturrate et al. (2010).

3.2.1 Utilizing ErrPs in BCI applications

Currently, the classifier of most BCIs are based on supervised learning and has no available feedback during a session. Moreover, the performance of the classifier often decline over time because the EEG characteristics of each individual depend on the subject's state of mind, and may therefore change considerably between sessions. ErrPs provide a way to notify the classifier when commands are classified incorrectly. Consequently, ErrPs can enable BCI classifiers to also rely on reinforcement learning techniques and algorithms, which can boost performance, enhance reliability, reduce training time, and make the classifier adapt to changes in the EEG waveform.

In recent years, the idea of using ErrPs for feedback have gained support by many others in the field of BCI research (Dornhege and Millán (2007a)). There have also been attempts at using ErrPs as reward signals for reinforcement learning frameworks.

Iturrate et al. (2010) directly applies ErrP-detection to a reinforcement learning algorithm. Figure 3.2 shows the graphical interface of this experimental setup. Two subjects observed the virtual 2D robot performing a reaching task, with the objective of getting the robot to a desired basket by eliciting voluntary reward-based signals. The algorithm updates the weights that define the desirability for each basket based on whether a feedback ErrP is present or not, and makes a new selection. 92% and 75% of the executions converged towards the correct basket, however, the convergence took an average of 70 and 100 steps, respectively. This is a good example of how an application can utilize ErrPs, but it also shows the importance of sufficient single-trial detection.

Chavarriaga and Biasiucci (2010) describe an experiment that make use of the interaction ErrP. In their setup, the system recognizes gestures made by the user, and rely on detection of ErrPs to update the classifier. This study shows that even with ErrP detection rates just above random levels, the ErrPs provide sufficient information to significantly improve the overall system performance.

As we are concerned with improving the performance of BCIs in control applications, the latter study provides interesting results. The problem is quite similar: The system is to recognize commands sent by the user, and we want the system to adapt to the user and perform better over time. Even though it is similar, the nature of the erroneous events in a gesture recognition system differ from those in a BCI. Because ErrPs tend to be very event and subject specific, we have to rely on measuring them in a setup that resembles the intended application. Currently, there is a lack of experiments that describe the interaction ErrP in a BCI control setting. In chapter 4, we design a setup that resembles a typical BCI control application and conduct experiments for detecting interaction ErrPs using a consumer grade headset to obtain EEG measurements.

Moreover, it is likely that the system will adapt at a much higher rate with better single-trial detection. However, as most studies use laboratory grade equipment, a BCI that use consumer grade equipment is dependent on a high-performing classifier to at least detect the ErrPs in the first place. This is the topic for the next section, which address a type of framework that is suitable for modeling the complex dependencies in the EEG signal.

3.3 Discriminative graphical models for EEG classification

Under the assumption that there exist statistical dependencies in the temporal domain of the signal we want to model, EEG classification becomes a sequence labeling problem. That is, we want to learn a mapping between a sequence of observations $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$ and a corresponding sequence of labels $\boldsymbol{y} = \{y_1, y_2, \ldots, y_T\}$. A natural way of visualizing the resulting model is a graph G, where the vertices $V = X \cup Y$ represent the set of random variables and the weighted edges $E = P(Y|X)$ define the state transitions.

Conditional random fields (CRFs) is a class of such graphical models; a probabilistic framework intended for sequence labeling tasks. It was first presented in Lafferty et al. (2001), where

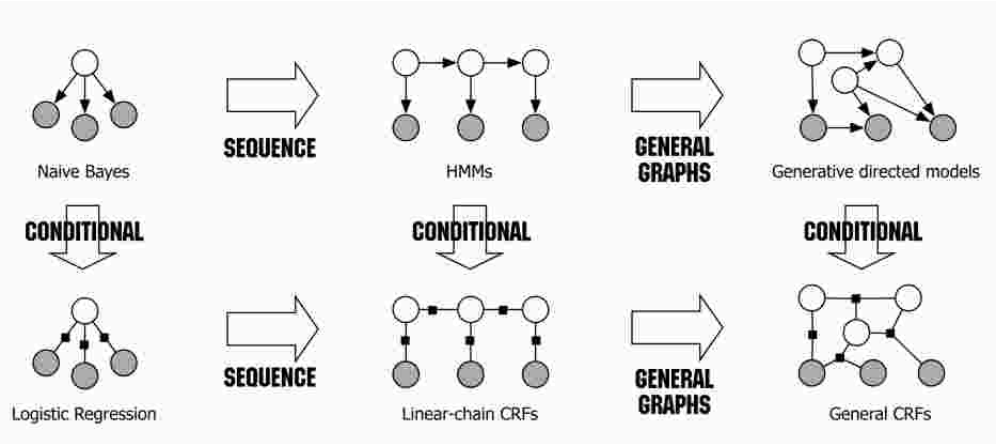


Figure 3.3: Graphical representation of generative-discriminative model pairs. (From Sutton and Mccallum (2006))

the authors point out how the CRF offer several advantages (when used in classification contexts) over conventional Markov-based models. Shortly after, there was an "explosion of interest in CRFs, with successful applications including text-processing, bioinformatics and computer vision" (Sutton and Mccallum (2006)).

In this section, we first show our motivation for this type of model by presenting an important advantage that conditional models offer. Furthermore, we give an introduction to CRFs in general and review some CRF variations that has performed favorably in BCI applications. Then we reason for our final choice of model, namely the Sparse Hidden-Dynamic Conditional Random Fields (SHDCRF), which has not yet been used in a BCI context.

3.3.1 Generative vs. discriminative models

Graphical models have traditionally been used to model the joint probability distribution $p(\boldsymbol{y},\boldsymbol{x})$ (Sutton and Mccallum (2006)), which is the case for the well-known Hidden Markov Model (HMM). This set of models, also known as Bayesian networks, is based on *directed* graphs because the outputs topologically precede the inputs. Since they describe how the outputs probabilistically generate the inputs, they are said to be *generative*.

Undirected graphical models directly model the *conditional* distribution $p(\boldsymbol{y}|\boldsymbol{x})$ and is said to be *discriminative*. While generative models must include a representation of the distribution $p(\boldsymbol{x})$, which can be difficult because of complex dependencies in the input data, discriminative

models does not explicitly model $p(\boldsymbol{x})$ and allow for the use of rich, overlapping features of the input sequence.

Generative and discriminative models that share the same topological representation is said to form a generative-discriminative pair. In order to highlight the key differences in such pairs, consider the well-known classifiers naive Bayes and logistic regression, whose graphical structure is illustrated in Figure 3.3. The naive Bayes (eq. 2.9) assume that once the class label is known, all the features are independent, and models each local dependency with a probability distribution. Logistic regression on the other hand, conditions the variables globally, typically by using optimization routines with the conditional likelihood as objective function. In fact, the difference between these two classifiers is *only* due to the fact that the one is generative and the other is discriminative. This means that if the naive Bayes is trained to maximize the conditional likelihood, we recover the same classifier as logistic regression. Conversely, if the logistic regression is interpreted generatively, we get the naive Bayes.

Finally, in order to illustrate why discriminative models is a better choice in BCI application, we construct a simple classification scheme that will serve as an example throughout the section. Consider a sequence of observations $\{x_t\}$ produced from an EEG signal that is recorded while the user performs mental imagery of either hand, whereas the left and right hand corresponds to the command signals y_{left} and y_{right}, respectively. Suppose the observations is simply the amplitude of the SMR; then we want to detect the successive ERD and ERS that occur during the imagined movement (Figure 2.6). Using a generative model to detect this would be fine as long as the features are defined in succession. For instance, the first order HMM would require only one transition for each state x_i. However, once we expand our model, such as to include detection of both hands y_{both}, or to introduce a rest state between commands, the complexity of the relationship between observations grows as they become interdependent and overlapping, and modeling the probability distribution $p(\boldsymbol{x})$ quickly becomes infeasible. By choosing a discriminative framework, we do not care about modeling $p(\boldsymbol{x})$, which is not used in classification anyways. This allows for including the long-range dependent and overlapping relationships that are present in EEG, and makes the model flexible and more receptive to expansion.

3.3.2 Introduction to conditional random fields (CRFs)

Given the random variables X and Y, consider a sequence of observations $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$ and a corresponding sequence of labels $\boldsymbol{y} = \{y_1, y_2, \ldots, y_T\}$ where each $x_t \in X$ and $y_t \in Y$. By definition, whenever x_t and y_t takes on real singe values, $\boldsymbol{x}$ and $\boldsymbol{y}$ are stochastic time series, and the set of random variables $X \cup Y$ defines a stochastic process. Now, suppose the parameters take on multidimensional values, such that $x_t \in \mathbb{R}^m$ and $y_t \in \mathbb{R}^n$. Then the pair (X, Y) define a *random field*, and is thus a generalization of a stochastic process. Lafferty et al. (2001) gives the formal definition of a *conditional random field* as follows:

> Let $G = (V, E)$ be a graph such that $Y = (Y_v)_{v \in V}$, so that Y is indexed by the vertices of G. Then (X, Y) is a conditional random field in case, when conditioned on X, the random variables Y_v obey the Markov property with respect to the graph: $P(Y_v|X, Y_w, w \neq v) = P(Y_v|X, Y_w, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G.

In other words, a CRF is a conditional distribution $p(\boldsymbol{y}|\boldsymbol{x})$ with an associated graphical structure that allows the parameters $x_t \in X$ and $y_t \in Y$ to take on multidimensional values. It is also recognized as a discriminative model because the random field (X, Y) is conditioned *globally* on the observations X.

Linear-chain CRFs

To show how the conditional distribution $p(\boldsymbol{y}|\boldsymbol{x})$ is represented mathematically, we begin by considering the joint distribution $p(\boldsymbol{y}, \boldsymbol{x})$ of an HMM. As with the relationship between naive Bayes and logistic regression, the HMM also has a generative counterpart; namely the linear-chain CRF, whose graphical structure is illustrated in Figure 3.3.

For any graphical model, the family of probability distributions factorizes according to its graph G if there exist a set of local functions Ψ_a such that

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{a=1}^{A} \Psi_a(\mathbf{y}_a) \tag{3.1}$$

where $\Psi_a(\mathbf{y}_a) \geq 0$ for all $\mathbf{y}_a$ and a indexes the factors of the graph (the factors can be seen as

the black squares between vertices in Figure 3.3). The constant Z is a normalization factor that ensures the probabilities sums to 1, and is given as

$$Z = \sum_{\mathbf{y}} \prod_{a=1}^{A} \Psi_a(\mathbf{y}_a). \tag{3.2}$$

With this convention, we can express the joint distribution of an HMM (2.14), in terms of a factor graph:

$$p(\mathbf{y},\mathbf{x}) = \frac{1}{Z} \prod_{t=1}^{T} \exp\left\{ \sum_{i,j\in S} \lambda_{ij} \mathbf{1}_{\{y_t=i\}} \mathbf{1}_{\{y_{t-1}=j\}} + \sum_{i\in S} \sum_{o\in O} \mu_{io} \mathbf{1}_{\{y_t=i\}} \mathbf{1}_{\{x_t=o\}} \right\} \tag{3.3}$$

where $\lambda_{ij} = \log p(y' = i|y = j)$ and $\mu_{oi} = \log p(x = o|y = i)$. Here, the factors are chosen such that (3.3) represent the same distribution as in (2.14).

We introduce the concept of *feature functions* to write (3.3) in a more compact way. Each feature function has the form $f_k(y_t, y_{t-1}, x_t)$, and we define a set of feature functions that satisfies

$$f_{ij}(y, y', x) = \mathbf{1}_{\{y=i\}} \mathbf{1}_{\{y'=j\}}$$
$$f_{io}(y, y', x) = \mathbf{1}_{\{y=i\}} \mathbf{1}_{\{x=o\}}$$

We assign one feature for each state transition and one feature for each state observation. Then we can write (3.3) as

$$p(\boldsymbol{y},\boldsymbol{x}) = \frac{1}{Z} \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t) \right\} \tag{3.4}$$

which still represent exactly the same family of distributions as (2.14). Finally, we can express the conditional distribution $p(\mathbf{y}|\mathbf{x})$ in terms of the joint distribution (3.4) as

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{p(\boldsymbol{y},\boldsymbol{x})}{\sum_{\mathbf{y}'} p(\boldsymbol{y}',\boldsymbol{x})} = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\} \tag{3.5}$$

where the normalization function $Z(\boldsymbol{x})$ is

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\} \tag{3.6}$$

The conditional distribution in (3.5) is the linear-chain CRF. As Figure 3.3 illustrates, the linear-

chain CRF and HMM are topologically identical, but we have already gained immediate advantages by expressing this graph by its associated conditional distribution. For instance, the transition from state i to state j in an HMM receives the same score $\log p(y_t = i|y_{t-1} = j)$ regardless of the input. In a CRF, however, we can allow the transition (i, j) to depend on current or past observations simply by including new features of the form $f_k(y_t, y_{t-i}, \mathbf{x})$.

Parameter estimation

In order to train the model we need to estimate the parameter vector $\Lambda = \{\lambda_k\} \in \mathbb{R}^K$. Given a training set that consist of N sequence pairs $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N$, where each $\mathbf{x}^{(i)} = \left\{x_1^{(i)}, x_2^{(i)}, \ldots, x_T^{(i)}\right\}$ is a sequence of input vectors, and each $\mathbf{y}^{(i)} = \left\{y_1^{(i)}, y_2^{(i)}, \ldots, y_T^{(i)}\right\}$ is a sequence of corresponding labels, we seek to maximize the conditional log likelihood

$$L(\Lambda) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|\mathbf{y}^{(i)}) \tag{3.7}$$

Because the model often involves a large number of parameters, the model is prone to overfitting. Therefore it is common to add a regularization term, which is usually based on the Euclidean norm of the parameter vector Λ and a regularization parameter $\frac{1}{2\sigma^2}$ that determines the strength of the penalty. When we add the regularization term and substitute the CRF model into (3.7), the objective function becomes

$$L(\Lambda) = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^i) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2} \tag{3.8}$$

The regularized log likelihood (3.8) is concave, which follows from the convexity of functions of the form $f(\mathbf{x}) = \log \sum_i \exp x_i$. Thus the global optimum can be found by using gradient descent methods, where the partial derivatives of (3.8) are

$$\frac{\partial L(\Lambda)}{\partial \lambda_k} = \sum_{i=1}^{N}\sum_{t=1}^{T} f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{y,y'} f_k(y, y', \mathbf{x}_t^{(i)}) p(y, y'|\mathbf{x}^{(i)}) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2} \tag{3.9}$$

We can verify the integrity of (3.9) by examining the first two terms. The first term is the expected value of f_k under the empirical distribution $\tilde{p}(\mathbf{y}, \mathbf{x})$, and the second term is the expected value of f_k under the model distribution $p(\mathbf{y}|\mathbf{x}; \Lambda)\tilde{p}(\mathbf{x})$. Consequently, at a maximum likelihood solution

of (3.7), the two terms are equal and the gradient is zero.

The choice of optimization routines are plenty, however, some are better suited for CRFs than others. The trivial approach is to optimize $L(\Lambda)$ by steepest ascent along the gradient, but this usually requires too many iterations. Newton's method, and similar routines, converge much faster because they account for the curvature of the objective function. However, as realistic applications use a high number of parameters, and the computation of the matrix of all second derivatives (the Hessian) is quadratic in the number of parameters, computing the Hessian or even storing it may not be practical.

Quasi-Newton methods, which approximate the Hessian based on the first derivatives, are better suited. The BFGS method have been successful in training CRFs, especially the limited-memory version of BFGS (Sutton (2012)). In addition, optimizers that use conjugate gradient techniques, which also approximate second-order information, have showed good performance in CRF applications as well (Sutton and Mccallum (2006)).

Inference

In classification tasks, the objective is usually to find the most likely sequence of labeling for a new test sequence $\mathbf{x}$

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) \tag{3.10}$$

However, a common key challenge in CRFs (and other graphical models) is to efficiently compute the marginal probabilities that are required for both training and inference. During training, computing the gradient (3.9) requires the marginal distribution for each edge transition $p(y_t, y_{t-1}|\mathbf{x})$, and finding the most likely labeling requires the marginal probabilities $p(y_t|x_t)$.

Computing these probabilities directly is intractable for most real-world models. The brute-force procedure is to generate all possible $T^{|Y|}$ state sequences and calculate the conditional probabilities for each one given the observed series of events, resulting in a time complexity of $O(|Y| \cdot T^{|Y|})$ for one sequence.

This is solved by *belief propagation*, for instance using the Viterbi path algorithm (Viterbi (1967)) or the forward-backward algorithm. Belief propagation relies on message passing and dynamic programming techniques. The basic principle is to compute the marginal probabilities

of the first node y_0 and the conditional (transition) probabilities of the next node y_1. Then the belief states are updated for node y_1, and the process is repeated up until node y_t. As the name of the forward-backward algorithm implies, the procedure can include propagation of beliefs in both directions, and may thus rely on both future and past evidence to achieve more exact estimations.

Belief propagation drastically reduce the inference process, and an algorithm like the forward-backward computes the marginal distributions in the gradient with time complexity $O(T \cdot |Y|^2)$. However, training CRFs can still be very computational expensive as the optimization procedure requires to compute the gradient of all training instances at each step. For data sets with a high number of states, or a very high number of training sequences, the optimization routine could require hours or even days to finish on current hardware.

General CRFs

So far, we have discussed CRFs that have a linear-chain graphical structure. The generalization to CRFs with arbitrary graph structures, as shown in the lower right of Figure 3.3, is fairly straightforward and only requires to modify the factors of the graph, such that the feature functions are functions of an arbitrary set of nodes:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(\mathbf{y}, \mathbf{x}) \right\} \tag{3.11}$$

Therefore, CRFs are quite flexible and easily modified to include complex dependencies between the entities.

However, the task of inference typically grows in complexity and may require more delicate routines, especially for graphs that contain loops. Nonetheless, a number of efficient inference approximations have been described for graphical models in general (Frey and Jojic (2005)), and many of them are also applicable to CRFs. As we primarily will focus on CRFs with linear structures, a discussion of these methods are outside the scope of this thesis.

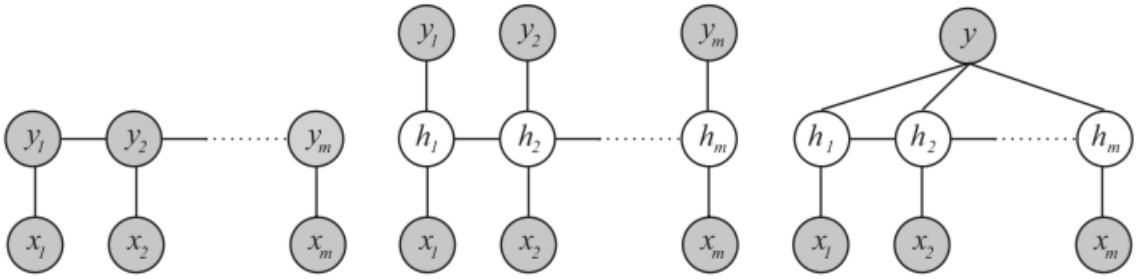


Figure 3.4: Comparison of the CRF, LDCRF and HCRF model (left to right). Grey nodes are observed variables and white nodes are hidden variables (from Morency et al. (2007)).

3.3.3 CRFs with hidden variables

Despite the flexibility of the ordinary CRF, there are certain desirable properties that the model does not inhabit. Recall that several mental strategies commonly used in BCIs produce complex patterns in the EEG, and some patterns may also be specific to the individual user. As BCIs relies on labeled training data to tune the classifier, some labels becomes very broad, such that a single label accounts for a large variety of sub-patterns. These sub-patterns are not transparent, and when EEG are labeled during BCI training it is only possible to label the intentions of the user.

As we argue for in Soukup (2013), this can be solved by assigning a set of sub-states to each observable state, such that the set of sub-states describe the intrinsic dynamics of each intent. This modeling approach enables the classifier to model transitions and observations more closely. Again, consider our classification scheme from earlier, where motor imagery of the left and right hand is subject for classification. A regular CRF models the successive ERD and ERS as a whole based on features from the entire window. By assigning two hidden states, whereas one models the ERD and the other models the ERS, it seems reasonable to believe that the classifier could infer the transition between these hidden states, and thus label the whole time window with greater accuracy.

The hidden conditional random fields (HCRF) model by Quattoni and Wang (2007) and latent-dynamic conditional random field (LDCRF) model by Morency et al. (2007) incorporate such hidden states. Figure 3.4 shows the graphical structure of both models, where the gray nodes are observable variables and the white nodes are hidden (or latent) variables. These models assign an intermediate layer of hidden states $\mathbf{h} = \{h_1, h_2, \ldots, h_t\}$ to the graph G, such that each h_t are connected by factors, or feature functions, to $\mathbf{x}$ and y_t. The difference between these two

models is the constraint imposed on the labels **y**: The HCRF model allows only a single label y_j for an entire training instance $(\mathbf{x}^{(i)}, y_j^{(i)})$, while the LDCRF allow arbitrary label sequences **y**. In addition, the LDCRF is "dynamic", because it allows multiple labels $y_j \in \mathbb{R}^m$ at each time slice, while the regular CRF and the HCRF allows only single-dimensional labels. The dynamic CRF (Sutton et al. (2007)) is a separate model, and can be described as a regular CRF with multi-dimensional labels, but without the layer of hidden states.

Although the HCRF has achieved good scores in BCI classification tasks (Saa and Cetin (2011)), we keep our focus on the LDCRF, because it represent a progression into our final choice of model.

Latent-dynamic CRF (LDCRF)

The model definition of the LDCRF can be directly deduced from its graphical structure shown in the center of Figure 3.4:

$$p_\Lambda(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{h}} p_\Lambda(\mathbf{y}|\mathbf{h})\, p_\Lambda(\mathbf{h}|\mathbf{x}) \tag{3.12}$$

However, the LDCRF impose constraints on the hidden variables **h** to keep training and inference tractable: The sets of hidden states associated with each class label y_j are required to be disjoint, such that each h_j is a member of a separate set H_{y_j} of possible hidden states for the class label y_j. With this constraint, we have $p_\Lambda(\mathbf{y}|\mathbf{h}) = 0$ for sequences that have any $h_j \notin H_{y_j}$, and the model expression (3.12) simplifies to

$$p_\Lambda(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{h}:\forall h_j \in H_{y_j}} p_\Lambda(\mathbf{h}|\mathbf{x}) \tag{3.13}$$

As in regular CRFs, the distribution $p_\Lambda(\mathbf{h}|\mathbf{x})$ is a member of the exponential family and is given by

$$p_\Lambda(\mathbf{h}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_{k=1}^{K} \lambda_k \mathbf{F}_k(\mathbf{h}, \mathbf{x}) \right\} \tag{3.14}$$

where the partition function Z is

$$Z(\mathbf{x}) = \sum_{\mathbf{h}'} \exp\left\{ \sum_{k=1}^{K} \lambda_k \mathbf{F}_k(\mathbf{h}', \mathbf{x}) \right\} \tag{3.15}$$

and $\mathbf{F}_k(\mathbf{h},\mathbf{x})$ is the feature function for the sequences, which can be rewritten as

$$\mathbf{F}_k(\mathbf{h},\mathbf{x}) = \sum_{t=1}^{T} f_k(h_{t-1}, h_t, \mathbf{x}) \tag{3.16}$$

Here, $f_k(h_{t-1}, h_t, \mathbf{x})$ can be a state function $s_k(h_t, \mathbf{x})$ or a transition function $t_k(h_{t-1}, h_t, \mathbf{x})$, whereas the transition functions t_k are responsible for capturing the hidden-dynamic relationships. Finally, the model parameters can be learned by maximizing the objective function

$$L(\Lambda) = \sum_{i=1}^{N} \log p_\Lambda(\mathbf{h}|\mathbf{x}^{(i)}) - \frac{||\Lambda||^2}{2\sigma^2} \tag{3.17}$$

and the training procedure itself is the same as for a regular CRF, with the exception that we condition the model based on hidden variables instead of observable class variables. Thus the computational complexity is also of the same magnitude.

The LDCRF serves as a good framework for modeling EEG and has achieved the best scores in common BCI classification tasks (Saa and Çetin (2012)). Yet, we argue for that the constraints imposed on the hidden labels $\mathbf{h}$ are unreasonable and fail to fully mirror realistic data. This constraint means that any hidden state h_j is strictly tied to its class label y_j, or from the viewpoint of the hidden state: It believes *only* in y_j and know of no other labels. As each h_j seeks to maximize the objective function by learning the best hidden structures in the data, it will occupy the label of that structure. Suppose at time t there is a distinct sub-structure in the data, and the confidence for a certain $h_t \in H_{y_j}$ that models this sub-structure is very high. Then, of course, the model will label this segment accordingly as $y_t = y_j$. The issue arise whenever labels share the same sub-structure. An example from the classification scheme presented earlier could be the class labels y_{left} and y_{both}. These class labels are likely to share a common subset of patterns, and if the confidence is high for a h_j that describe one of these patterns, it is a question of which label it was assigned to in the first place.

3.3.4 Sparse hidden-dynamic CRF (SHDCRF)

The SHDCRF relaxes the constraints in the LDCRF model and allows hidden states to share labels, such that $h_j \in H$ where H is the set of all hidden states. Shen et al. (2011) is the first and

only publication to describe this model and applies it to user intent understanding in internet search sessions, where the SHDCRF is superior compared to other sequential models.

SHDCRFs have the same model definition as (3.12), however, as we must assume $p_\Lambda(\mathbf{y}|\mathbf{h}) \geq 0$ the expression cannot be simplified:

$$p_\Lambda(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{h}} p_\Lambda(\mathbf{y}|\mathbf{h})\, p_\Lambda(\mathbf{h}|\mathbf{x}) \tag{3.18}$$

Each term in (3.18) are defined by the usual conditional random field formulation:

$$p_\Lambda(\mathbf{y}|\mathbf{h}) = \frac{1}{Z(\mathbf{h})} \exp\left\{ \sum_{k=1}^{p} \beta_k \mathbf{G}_k(\mathbf{y}, \mathbf{h}) \right\} \tag{3.19}$$

$$p_\Lambda(\mathbf{h}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_{k=1}^{q} \lambda_k \mathbf{F}_k(\mathbf{h}, \mathbf{x}) \right\} \tag{3.20}$$

Here, $\mathbf{G}_k$ represent the state feature functions for intent class labels and $\mathbf{F}_k$ represent state and transition functions for the hidden variables, with a grand total of $q + p = K$ feature functions for the entire model. Furthermore, λ_k and β_k are the model parameters corresponding to the feature functions $\mathbf{G}_k$ and $\mathbf{F}_k$, respectively, and the parameter vector for the whole model is given by $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_q, \beta_1, \beta_2, \ldots, \beta_p\}$. Finally, the partition functions in (3.19) and (3.20) that ensure the probabilities to add up to 1, are

$$Z_1(\mathbf{h}) = \sum_{\mathbf{y}'} \exp\left\{ \sum_{k=1}^{p} \beta_k \mathbf{G}_k(\mathbf{y}', \mathbf{h}) \right\} \tag{3.21}$$

$$Z_2(\mathbf{x}) = \sum_{\mathbf{h}'} \exp\left\{ \sum_{k=1}^{q} \lambda_k \mathbf{F}_k(\mathbf{h}', \mathbf{x}) \right\} \tag{3.22}$$

An important aspect of this model is the sparse relation imposed on the conditional distribution $p_\Lambda(\mathbf{y}|\mathbf{h})$. Since the model incorporate an intermediate layer of hidden states, and at the same time aims to learn the distribution between the hidden layer and the class labels, the likelihood function becomes non-convex. In order to avoid the bad local optima during training, another regularization term is added to the objective function. This term is based on the conditional entropy $H_\Lambda(Y|H)$ and seeks to ensure a sparse relation between the hidden variables and the class labels.

Given a training set of N labeled sequences $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, the objective function subject to optimization is

$$L(\Lambda) = \sum_{i=1}^{N} \tilde{p}(\mathbf{x}, \mathbf{y}) \log p_\Lambda(\mathbf{y}|\mathbf{x}) - \frac{||\Lambda||^2}{2\sigma^2} - \alpha H_\Lambda(Y|H) \tag{3.23}$$

where the term $\tilde{p}(\mathbf{x}, \mathbf{y})$ is the expected value of $\mathbf{F}_k$ under the empirical distribution. The conditional entropy is defined as

$$H_\Lambda(Y|H) = \sum_{\dot{h} \in H} \sum_{\dot{y} \in Y} p_\Lambda(\dot{y}|\dot{h}) \log p_\Lambda(\dot{y}|\dot{h}) \tag{3.24}$$

where the pair $(\dot{y}, \dot{h})$ is given by parameter β_k under the constraint of its associated feature function $g_k(\dot{y}, \dot{h}) = 1$. The conditional entropy is a measure of how much two random variables overlap, such that when two random variables are conditional independent, the conditional entropy is zero. With the reduction of $H_\Lambda(Y|H)$, the uncertainty of the class labels given hidden states is also decreasing, and in the special case where $H_\Lambda(Y|H)$ is minimized to zero, each hidden state correspond to a single class label (as in the LDCRF). The parameter α in (3.23) is a factor to determine how strongly this conditional entropy is punished during training, and acts like σ in $L1$-regularization terms.

Parameter estimation

When training the model, the goal is to learn the optimal parameters

$$\Lambda^* = \arg\max_{\Lambda} L(\Lambda) \tag{3.25}$$

As in the other CRF models, the optimal parameters are best learned by employing a gradient-based method. For the experiments in Shen et al. (2011) the L-BFGS (Liu and Nocedal (1989)) is used. We start by giving the partial derivatives of λ_k and β_k:

$$\begin{aligned} \frac{\partial L(\Lambda)}{\partial \lambda_k} = &\sum_{i=1}^{N} \tilde{p}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \sum_{\mathbf{h}'} p_\Lambda(\mathbf{h}'|\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) F_k(\mathbf{x}^{(i)}, \mathbf{h}') \\ &- \sum_{i=1}^{N} \tilde{p}(\mathbf{x}^{(i)}) \sum_{\mathbf{h}', \mathbf{y}'} p_\Lambda(\mathbf{h}', \mathbf{y}'|\mathbf{x}^{(i)}) F_k(\mathbf{x}^{(i)}, \mathbf{h}') - \frac{\lambda_k}{\sigma^2} \end{aligned} \tag{3.26}$$

$$\frac{\partial L(\Lambda)}{\partial \beta_k} = \sum_{i=1}^{N} \tilde{p}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \sum_{\mathbf{h}'} p_\Lambda(\mathbf{h}'|\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) G_k(\mathbf{y}^{(i)}, \mathbf{h}') - \sum_{i=1}^{N} \tilde{p}(\mathbf{x}^{(i)}) \sum_{\mathbf{h}', \mathbf{y}'} p_\Lambda(\mathbf{h}', \mathbf{y}'|\mathbf{x}^{(i)}) G_k(\mathbf{y}^{(i)}, \mathbf{h}') - \frac{\beta_k}{\sigma^2} + p(\dot{y}|\dot{h}) \left(\beta_k - \sum_{\ddot{y} \in Y} p(\ddot{y}|\dot{h}) \beta_{K(\ddot{y}, \dot{h})} \right) \tag{3.27}$$

In the last term of (3.27), $K(\ddot{y}, \dot{h})$ is the unique k^* that satisfies $g_{k^*}(\ddot{y}, \dot{h}) = 1$. These partial derivatives cannot be computed directly, because $\mathbf{h}'$, $\mathbf{y}'$, $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ are sequences, and as before we have to rely on belief propagation for efficient computation. We disengage the sequence variables and rewrite (3.26) and (3.27) to a form more amendable to implementation:

$$\frac{\partial L(\Lambda)}{\partial \lambda_k} = \sum_{i=1}^{N} \tilde{p}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \sum_{t=1}^{T^{(i)}} p_\Lambda(h_{t-1}, h_t|\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) f_k(h_{t-1}, h_t, x_t^{(i)}) - \sum_{i=1}^{N} \tilde{p}(\mathbf{x}^{(i)}) \sum_{t=1}^{T^{(i)}} p_\Lambda(h_{t-1}, h_t|\mathbf{x}^{(i)}) f_k(h_{t-1}, h_t, x_t^{(i)}) - \frac{\lambda_k}{\sigma^2} \tag{3.28}$$

$$\frac{\partial L(\Lambda)}{\partial \beta_k} = \sum_{i=1}^{N} \tilde{p}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \sum_{t=1}^{T^{(i)}} p_\Lambda(h_t|\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) g_k(h_t, y_t^{(i)}) - \sum_{i=1}^{N} \tilde{p}(\mathbf{x}^{(i)}) \sum_{t=1}^{T^{(i)}} p_\Lambda(h_t|\mathbf{x}^{(i)}) p_\Lambda(h_t|y_t) g_k(h_t, y_t^{(i)}) - \frac{\beta_k}{\sigma^2} + p(\dot{y}|\dot{h}) \left(\beta_k - \sum_{\ddot{y} \in Y} p(\ddot{y}|\dot{h}) \beta_{K(\ddot{y}, \dot{h})} \right) \tag{3.29}$$

Belief propagation

There are four conditional probability distributions in (3.28) and (3.29) that we need to estimate:

$$p_\Lambda(h_t|\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$$

$$p_\Lambda(h_{t-1}, h_t|\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$$

$$p_\Lambda(h_t|\mathbf{x}^{(i)})$$

$$p_\Lambda(h_{t-1}, h_t|\mathbf{x}^{(i)})$$

In Shen et al. (2011), these are reportedly estimated using the Viterbi path algorithm Viterbi (1967). However, we will rely on the forward-backward algorithm for this purpose. The forward-backward routine can be rewritten to a series of matrix multiplications. Consider the two conditional distributions $p_\Lambda(h_t|\mathbf{x}^{(i)},\mathbf{y}^{(i)})$ and $p_\Lambda(h_{t-1},h_t|\mathbf{x}^{(i)},\mathbf{y}^{(i)})$, then at position $t=1$ we define the vector random variable $[M_1(h_t|\mathbf{x}^{(i)},\mathbf{y}^{(i)})] = [\exp\{\phi_t(h_t|\mathbf{x}^{(i)},\mathbf{y}^{(i)})\}]$ of length $|H|$, and for all $1 < t \leq T^{(i)}$ we define the $|H| \times |H|$ matrix $[M_t(h_{t-1},h_t|\mathbf{x}^{(i)},\mathbf{y}^{(i)})] = [\exp\{\phi_t(h_{t-1},h_t|\mathbf{x}^{(i)},\mathbf{y}^{(i)})\}]$ where

$$\phi_t(h_t|\mathbf{x}^{(i)},\mathbf{y}^{(i)}) = \sum_{k=1}^{p} \beta_k g_k(h_t, y_t^{(i)}) + \sum_{k=1}^{q} \lambda_k s_k(h_t, x_t^{(i)}) \tag{3.30}$$

$$\phi_t(h_t,h_{t-1}|\mathbf{x}^{(i)},\mathbf{y}^{(i)}) = \sum_{k=1}^{p} \beta_k g_k(h_t, y_t^{(i)}) + \sum_{k=1}^{q} \lambda_k f_k(h_t, h_{t-1} x_t^{(i)}) \tag{3.31}$$

As in the LDCRF, $s_k(h_t, x_t^{(i)})$ in (3.30) is the state feature functions for hidden states, and $f_k(h_t, h_{t-1}x_t^{(i)})$ in (3.31) denotes both state functions $s_k(h_t, x_t^{(i)})$ and transition functions $t_k(h_t, h_{t-1}x_t^{(i)})$. In the forward-backward algorithm, we find the beliefs for each state h_a and each transition $h_a \rightarrow h_b$ at position $t > 1$, by propagating the beliefs backwards for each $t' = T^{(i)}..t$ and forward for each $t' = 2..t$:

$$p_\Lambda(h_t = a|\mathbf{x}^{(i)},\mathbf{y}^{(i)}) = \frac{[M_1 \times \prod_{t'=2}^{t} M_{t'}]_a \times [\prod_{t'=t}^{T^{(i)}} M_{t'} \times \mathbf{1}]_a}{M_1 \times \prod_{t'=2}^{T^{(i)}} M_{t'} \times \mathbf{1}} \tag{3.32}$$

$$p_\Lambda(h_{t-1} = a, h_t = b|\mathbf{x}^{(i)},\mathbf{y}^{(i)}) = \frac{[M_1 \times \prod_{t'=2}^{t-1} M_{t'}]_a \times M_t[a,b] \times [\prod_{t'=t}^{T^{(i)}} M_{t'} \times \mathbf{1}]_b}{M_1 \times \prod_{t'=2}^{T^{(i)}} M_{t'} \times \mathbf{1}} \tag{3.33}$$

The other two conditional probabilities have the same form, the only difference is to add the contribution from all state feature functions for intent class labels g_k, not only the correct labels. The expressions in (3.32) and (3.33) can be implemented in an efficient matter using dynamic programming techniques. In addition, many implementations rely on the logarithm of (3.30) and (3.31) instead, since the numbers may grow very large and can cause computational overflows.

Inference

Given a set of optimal parameters Λ^* and a new test sequence $\mathbf{x}$, the most likely sequence of labels can be inferred by maximizing (3.18):

$$\begin{aligned} \mathbf{y}^* &= \arg\max_{\mathbf{y}} p_{\Lambda^*}(\mathbf{y}|\mathbf{x}) \\ &= \arg\max_{\mathbf{y}} \sum_{\mathbf{h}} p_{\Lambda^*}(\mathbf{y}|\mathbf{h}) p_{\Lambda^*}(\mathbf{h}|\mathbf{x}) \end{aligned} \tag{3.34}$$

Consequently, for a position t in the test sequence, the most probable label y_t^* is given by

$$y_t^* = \arg\max_{y_t} \sum_{h_t \in H} p_{\Lambda^*}(y_t|h_t) p_{\Lambda^*}(h_t|\mathbf{x}) \tag{3.35}$$

Note that also here, an estimate of $p_{\Lambda^*}(h_t|\mathbf{x})$ requires belief propagation. Depending on whether the application have real-time demands or not, this belief propagation can include both future and past observations, or only rely on past observations. $p_{\Lambda^*}(y_t|h_t)$ can be calculated directly from (3.19) because it only depends on the state feature functions $g_k(y_t, h_t)$.

The SHDCRF comes very close to the idealistic model we have deduced by reasoning. The relaxed constraints on $p(\mathbf{y}|\mathbf{h})$ enables the model to incorporate more realistic relationships among the variables. However, not without a cost, as this also makes the model more complex. There are many statistical dependencies to account for during training, and although the computational expense is of the same magnitude as the CRF and LDCRF, the model requires computation of numerous empirical and conditional distributions that can be demanding implementation-wise.

3.4 Combining CRFs and ErrPs in BCI control applications

CRF-based models for classification and ErrPs to provide reward-based signals are exciting and new approaches to BCI design. Both concepts have shown their power on a preliminary stage, and are gaining support in BCI research communities. However, to truly enforce the performance, reliability and usability of BCIs in control applications, it is necessary to merge these concepts. By doing so, the system is provided with an adaptive, flexible and high-performing

classifier.

Given the presence of ErrPs in the EEG signal, we seek to detect that the user experiences an error in addition to the other command signals. This can be achieved by having two separate classifiers working in parallel. However, as the CRF-based models are very flexible and receptive to expansion, it might be advantageous to incorporate the detection of all labels into a single model. Indeed, since the interaction ErrP usually follows ~100 ms after the BCI performs an erroneous action, the feature functions corresponding to the ErrP detection can depend on the observations from >100 ms earlier. Using the SHDCRF model we can extract even more information: If the model were in an hidden state that have ambiguous beliefs between two class labels, it is more probable that the error was caused by a misclassification between two commands, and not as a result of a false positive detection. In addition, the dynamic property in the SHDCRF allows for multiple labels at each time step, thus ErrPs can be detected simultaneously as commands.

Recently, studies have identified other neuro-physiological phenomenon that reveals information about the users state of mind during BCI operation (Blakely et al. (2013); Grosse-wentrup and Sch (2013)). As we argue for in Soukup (2013), the incorporation of multiple mental strategies is vital for a BCI to truly understand the user's intent. Because dynamic CRF-based models are designed to include complex, long-range dependencies, and are able to detect multiple labels at once, it is a suitable model for the task. This is also a major reason for why we believe CRF-based models have a huge future potential in BCI application.

3.4.1 Making CRFs adaptive

Combining CRFs and ErrPs involves making the CRF adaptive. However, CRF models are a relatively new invention, and there are currently a lack of literature that describe how CRFs can adapt to reward-based signals.

We propose a simple reinforcement learning algorithm that can serve this purpose. Assuming a SHDCRF model, where we have a conditional probability $p(y_t = i|h_t = j)$ for each $i \in Y$ and $j \in H$, the technique relies on a backtracking algorithm to shift the conditional distribution $p(y_t|h_t)$:

1. An interaction ErrP is detected at time $t + 100$ ms. The presence of this ErrP implies a misclassification of label y_t.

2. Backtrack and identify the label $a = y_t^*$. This requires a minimum of additional storage and computation, as we already have calculated the beliefs for past hidden states while performing forward belief propagation.

3. Shift the distributions $p(y_t|h_t)$ by performing the operations $\beta_k = \beta_k + \frac{\gamma}{|Y|-1}$ and $\beta_{k^*} = \beta_{k^*} - \gamma$ and recompute the class label beliefs $p(y_t|h_t)$. Here, β_k are the parameters that correspond to all feature functions $g_k(y_t \neq a, h_t)$, β_{k^*} correspond to the unique feature function $g_{k^*}(y_t = a, h_t)$ and γ is an appropriate regularization factor.

4. Continue on with operations.

In other words, the algorithm reduce the beliefs for $p(y_t = a, h_t)$, such that next time the model are in the hidden state h_t it is more unlikely to infer $y_t^* = a$. The regularization factor γ can be a function of β_k to ensure a suitable shift, or simply a constant. Because $p(\mathbf{y}, \mathbf{h})$ is a member of the exponential family, the re-computation of $p(y_t, h_t)$ ensures that the distribution is normalized such that $\sum_{y'} p(y_t = y'|h_t) = 1$.

This algorithm only modifies state feature function of the class labels, however, it is also possible to shift the conditional distribution $p(h_{t-1}, h_t)$ and re-propagate the beliefs for $p(h_{t'-1}, h_{t'})$, $t' = t..(t + 100)$. This requires development of a more complex algorithm, and is left for future work.

Chapter 4

Experiments and realization of concepts

Using ErrPs for user feedback and CRF-based models to improve classification are indeed promising concepts to incorporate in BCI design. However, there are currently a lack of studies that address how these concepts perform in our intended BCI control system. Although several studies on ErrPs have been published in recent years, the majority conduct experiment where laboratory grade EEG equipment is used. In addition, the SHDCRF we suggest has only been used in a single classification task, and it is very uncertain how it performs in modeling and classification of EEG.

The potential of our proposed concepts can only be verified through experimentation and implementation. In this chapter, we design an experiment for detecting ErrPs in BCI control applications using a consumer grade headset to obtain EEG recordings. In addition, we implement the SHDCRF model and measure its performance on a common BCI classification task. We present the equipment and software tools we have used, describe the development process, review testing and improvements, and finally present the results for both tasks.

4.1 Equipment

4.1.1 UR5

The UR5, shown in Figure 4.1, is a lightweight industrial manipulator manufactured by Universal Robots. The robot has 6 degrees of freedom and comes with a separate control box running on a Mini-ITX PC. The control box includes a graphical user interface on a 12 inch touchscreen with functionality for initialization, configuration and programming. A full technical specification of the robot is given in Table 4.2.

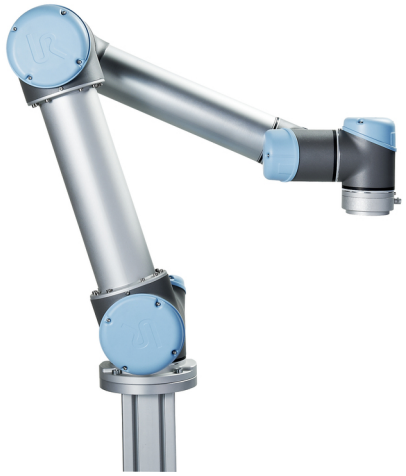


Figure 4.1: UR5 robot.

Control interface

The UR5 can be controlled by three different interfaces: The integrated touchscreen interface PolyScope, the script interface URScript and the low-level C-API. We will primarily control the robot at the script level, using a client application that connects to the robot over a TCP/IP connection.

URScript provide functions for moving the robot either in the joint-space or the tool-space of the robot. Each move has a trapezoidal velocity curve and are executed in real-time by the underlying RunTimeMachine (C-API), which also computes the kinematic transformations and generate trajectories. A selection of functions used by our client applications is given in Table 4.1, and the full URScript documentation is available online[1].

Additionally, we will use the python library `urx` (version 0.8.0), which provides another level of abstraction for communication with the UR5. This library sets up real-time monitors to send and receive data through the exposed TCP/IP interface of the robot, and implements wrappers for the functions listed in Table 4.1. The `urx` library is available through the official python package index PyPI.

[1] http://www.wmv-robotics.de/home_htm_files/scriptmanual_en_1.5.pdf

Table 4.1: Selection of functions built into the URScript programming language.

movej(q, a=3, v=0.75, t=0, r=0)
Move to position (linear in joint-space). The speed and acceleration parameters control the trapezoidal velocity curve of the move.

Parameters
- q: list of joint positions in radians, $q = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]^T$
- a: joint acceleration of leading axis [rad/s^2]
- v: joint speed of leading axis [rad/s]
- t: time [s]
- r: blend radius [m]

movel(q, a=3, v=0.75, t=0, r=0)
Move to position (linear in tool-space).

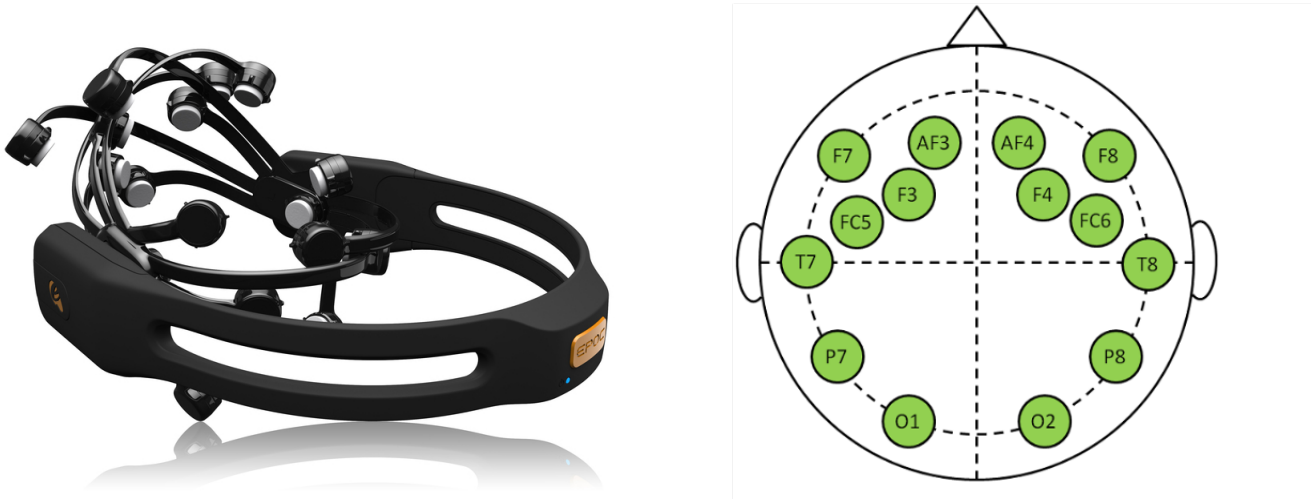
Parameters
- q: target pose, where q = position + orientation = $[x, y, z, o_x, o_y, o_z]^T$
- a: tool acceleration [m/s^2]
- v: tool speed [m/s]
- t: time [s]
- r: blend radius [m]

stopl(a)
Stop (linear in tool-space). Decellerate tool speed to zero.

Parameters
- a: tool decelleration [m/s^2]

Table 4.2: UR5 technical specifications.

Weight	18.4 kg / 40.6 lbs
Payload	5 kg / 11 lbs
Reach:	850 mm / 33.5 in
Joint ranges:	+/- 360° on all joints
Speed:	Joint: Max 180°/sec. Tool: Approx. 1 m/sec. / Approx. 39.4 in/sec.
Repeatability:	+/- 0.1 mm / +/- 0.0039 in (4 mil)
Footprint:	Ø149 mm / 5.9 in
Degrees of freedom:	6 rotating joints
Control box size (WxHxD):	475 mm x 423 mm x 268 mm / 18.7 x 16.7 x 10.6 in
I/O ports:	10 digital in, 10 digital out, 4 analogue in, 2 analogue out
I/O power supply:	24 V 1200 mA in control box and 12 V/24 V 600 mA in tool
Communication:	TCP/IP 100 Mbit: IEEE 802.3u, 100BASE-TX Ethernet socket & Modbus TCP
Programming:	Polyscope graphical user interface on 12 inch touchscreen with mounting
Noise:	Comparatively noiseless
IP classification:	IP54
Power consumption:	Approx. 200 watts using a typical program
Collaboration operation:	Tested in accordance with sections 5.10.1 and 5.10.5 of EN ISO 10218-1:2006
Materials:	Aluminium, ABS plastic
Temperature:	The robot can work in a temperature range of 0-50°C
Power supply:	100-240 VAC, 50-60 Hz
Calculated Operating Life:	35,000 Hours



(a) (b)

Figure 4.2: The Emotiv EPOC headset. (a) shows the headset and (b) shows the placement of electrodes according to the international 10-20 system.

4.1.2 Emotiv EPOC

Figure 4.2a is the Emotiv EPOC neuroheadset. It is produced by Emotiv Systems which is an Australian company that develop equipment for EEG-based BCIs. The Emotiv EPOC is a high resolution, multi-channel, wireless EEG device which is available on the consumer market. Table 4.3 shows the specification of this device, and Figure 4.2b show how the electrodes are positioned according to the 10-20 system. To achieve better signal quality a conductive solution is applied to the electrodes before usage. The EPOC also include a build-in gyroscope that record acceleration in the x- and y-direction.

Software

The Emotiv EPOC ships with a series of software suits which includes the following.

The Expressiv™Suite implements detection of facial expressions by picking up EOG induced by eyeball dipole rotation and EMG from different facial muscles.

The Affective™Suite aims at detecting emotional states of the user.

The Cognitive™Suite interprets users' conscious thoughts and intent. This suite provides an interface for training cognitive actions and mapping these to commands.

Table 4.3: Emotiv EPOC spesifications.

	EMOTIV EPOC HEADSET
Number of channels	14 (plus CMS/DRL references, P3/P4 locations)
Channel names (International 10-20 locations)	AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4
Sampling method	Sequential sampling. Single ADC
Sampling rate	128 SPS (2048 Hz internal)
Resolution	14 bits 1 LSB = 0.51µV (16 bit ADC, 2 bits instrumental noise floor discarded)
Bandwidth	0.2 - 45Hz, digital notch filters at 50Hz and 60Hz
Filtering	Built in digital 5th order Sinc filter
Dynamic range (input referred)	8400µV (pp)
Coupling mode	AC coupled
Connectivity	Proprietary wireless, 2.4GHz band
Power	LiPoly
Battery life (typical)	12 hours
Impedance Measurement	Real-time contact quality using patented system

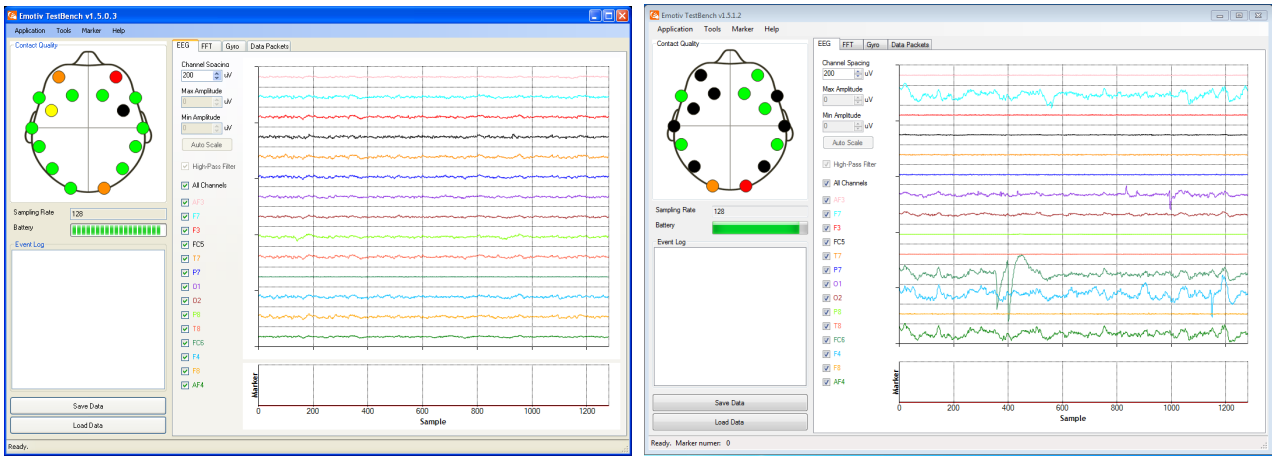
In addition, with the research edition of Emotiv EPOC, the software tool TestBench™is available. As shown in Figure 4.3, this software provides a real-time view of the data stream, including EEG, contact quality, FFT, gyro, wireless connection, marker events and headset battery level. It also provide the option to record and export the data in the `.edf` format.

There is also a python driver available for the Emotiv EPOC that provides access to the raw EEG data[2]. Even though it is not maintained by the Emotiv company, we prefer using this driver in our applications because it integrates easily with the python interface for the UR5 robot.

Limitations

The Emotiv EPOC is a commercial EEG device and does not provide the same quality as clinical equipment used in laboratories. Carrino and Dumoulin (2012) reviews the Emotiv EPOC and concludes that "The result shows that the low cost EEG device, at the actual state of the art, provide interesting results but can hardly be used for self-paced systems in error sensitive context." Additionally, the electrode placement is far from optimal when we want to detect ErrPs:

[2]Available at https://github.com/ozancaglayan/python-emotiv



(a) Headset condition in October 2013 (from Soukup (2013))

(b) Current headset condition

Figure 4.3: Screenshots of TestBench™

Figure 4.2b shows that the position of the electrodes are anywhere *but* the fronto-central and centro-parietal areas that ErrPs originate from.

Furthermore, our equipment is not in mint condition, and has decayed considerably since the work on this thesis started. Figure 4.3a shows the condition of the headset and the quality of each channel during the work in Soukup (2013), and Figure 4.3b shows the current condition of the headset. Many of the channels are either dead or contains unprovoked large fluctuations, and the headset does not deliver reliable EEG from each channel.

4.2 Experiments for detecting error-related potentials

The purpose of this experiment is to determine whether ErrPs is detectable in a real-life application using a consumer grade headset like the Emotiv EPOC to obtain EEG measurements. The motivation for introducing ErrP detection in BCI applications is huge, as we have argued for in the preceding chapter, because they can provide feedback to the system and ultimately enable BCI training to rely on a reinforcement learning scheme.

Our experiment is mainly inspired by Iturrate et al. (2010), where ErrP detection is directly applied to a reinforcement learning algorithm. However, the experimental setup described in Iturrate et al. (2010) differs from a typical BCI control context. First, the subjects elicit *voluntary* reward-based signals to correct a virtual 2D robot into reaching for the desired target. In

other words, the ErrPs itself are used for control, whereas in a typical BCI the control signals are produced using endogenous mental strategies. These voluntary ErrP may also have quite a different characteristic than ErrPs that are elicited spontaneously. Moreover, the experiment in Iturrate et al. (2010) use a series of static images (where each image represent a new position of the virtual robot) which are displayed at a rate of one image per second. Therefore, this is a discrete-time simulation and the results does not necessarily generalize to a continuous-time context.

The experiment has been designed to mimic the intended BCI applications as closely as possible. Recall from the previous chapter that there has been reported a variety of ErrPs that have different characteristics, which can mainly be classified by the type of erroneous event that occur. Therefore, it is important that the subject's experience of the situation that elicit the potentials resembles the experience of an error when using a BCI. We consider typical control applications where the BCI acts as a controller for anything that can give a visual confirmation of the commands. Examples of such setups could be control of a quadcopter as in LaFleur et al. (2013), or simply control of a computer cursor as in the popular publication by Wolpaw and McFarland (2004).

4.2.1 Experiment description

In our setup, the subject controls the UR5 robot in two dimensions using the arrow keys on a computer keyboard. Each arrow key corresponds to moving the UR5's tool linearly in the given direction. The Emotiv EPOC records the EEG during the whole session, and the subject is told to avoid unnessecary head movement or facial expressions that could contaminate the recordings. The experimental procedure is as follows:

1. Before a session, the subject decides to control the robot in a pattern of their choosing. The subject performs a training run where no errors are produced, for the sake of getting used to controlling the robot.

2. In the first part of a session, the robot moves according to the arrow keys to let the subject get going.

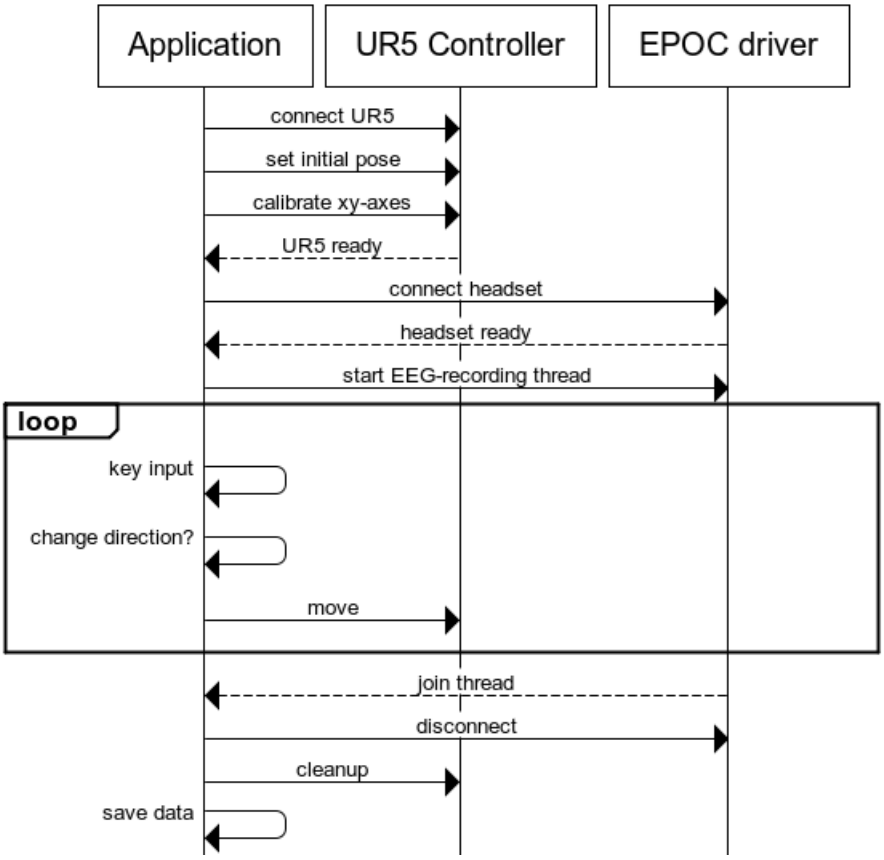


Figure 4.4: ErrP experiment sequence diagram.

3. After a given time, the controller starts to produce errors. Each time the subject initiate or change movement, there is a chance of the controller choosing another direction at random. After an error occurs, the subject continue the control pattern as before.

4.2.2 Implementation

The experiment is realized using the Python programming language, because we already have Python drivers available for both the Emotiv EPOC headset (`python-emotiv`) and the UR5 robot (`urx`). Since URScript only supports point-to-point movement with a trapezoidal velocity curve, as specified in Table 4.1, we have written an UR5 controller on top of `urx` that works around these limitations and acts as a continuous velocity controller for linear trajectories. The UR5 controller also implements functionality for calibrating the reference coordinate system used in the experiment, so that the arrow keys corresponds to movement along a 2D plane projected by

the subject, and also enables the application to switch between control spaces (base joint and tool).

Initialization and calibration

Figure 4.4 illustrates the program flow and communication between modules. The program is called by `./errp.py [save-file]` in the terminal of a UNIX-based operative system. First, the application connects with the robot and sets the initial joint configuration. At this point, the control mode is set to "joint", and the user has the ability to rotate the base joint such that the tool of the robot are in a direct frontal position. The robot calibrate the reference xy-axes and the control mode are set to "cartesian", such that the arrow keys correspond to the axes of the new coordinate system. The application connects with the Emotiv EPOC and initiates a countdown for the session to start.

Control loop

Before the session starts, the application initiate a separate thread that records the EEG. The program then enters the control loop where it polls for key inputs and send velocity vectors to the UR5 controller. After a time into the session, the velocity vectors are changed with a certain probability when the user controls the robot in a new direction. In the case of a produced error, there is a cooldown before another error is allowed to ensure a minimum spacing between produced errors.

Cleanup and saving data

When the session expires, the EEG-recording thread are joined and the interfaces to the headset and the robot are disconnected. Finally, the application synchronize the timestamps of the produced errors with the EEG recording, and the data is saved together as a `.mat`-file to the path given in the input argument.

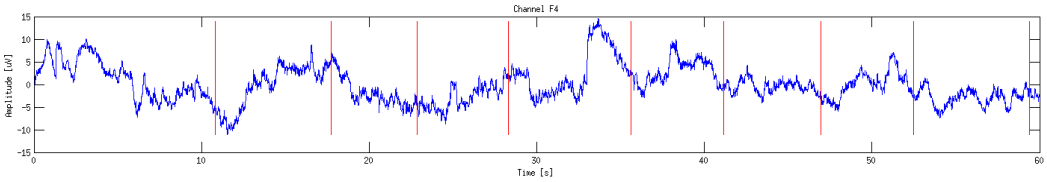


Figure 4.5: ErrP trial (channel F4) during initial testing. The red vertical bars indicate the event of a produced error.

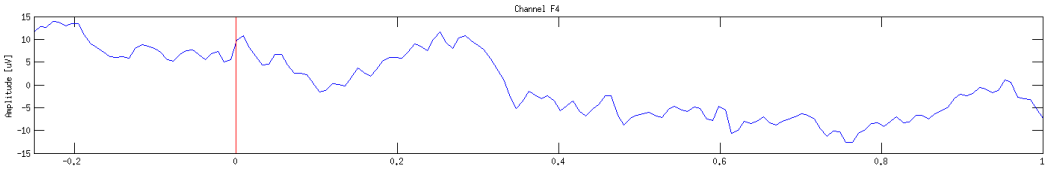


Figure 4.6: Grand average of 21 ErrP segments from initial trials. The red bar indicate the time of the produced errors.

4.2.3 Initial results and improvements

Initial tests and recorded trials during the development process revealed two issues: First, the condition of the headset had become worse, and the quality of the EEG data was not satisfactory. We verified this by inspecting the headset quality in Testbench™, and as Figure 4.3b shows, some electrodes are indeed not in a healthy condition. Furthermore, raw EEG plots from the test trials indicated that many of the channels either are dead, or contains random fluctuations well above the accepted level. However, as we are only interested in observing signals that originate from the frontal and parietal lobe, we eventually managed to get a healthy signal from channel F3 and F4 by switching and replacing electrodes. Figure 4.5 shows the raw trial from channel F4, where the vertical bars indicate the times of produced errors. Even though the signal is noisy, the amplitude is within $\pm 10 \mu V$, which is sufficient resolution to detect ErrPs. Despite the irregularities in the EEG, we decided to continue the experiment.

The second issue was that even with a healthy signal, the results indicated that ErrPs were not clearly observable in the trials. Figure 4.6 shows the grand average over 21 segments of produced errors. Comparing this result with Figure 3.1 and considering the description of ErrPs in the preceding chapter, our grand average may resemble the same characteristics as the interaction ErrP, as it indicates a negative peak after ~100 ms, a broader positive potential after 200 ms, and a second broad negative potential, but it is too vague to conclude upon. Moreover, it would prove difficult to detect in a classification task.

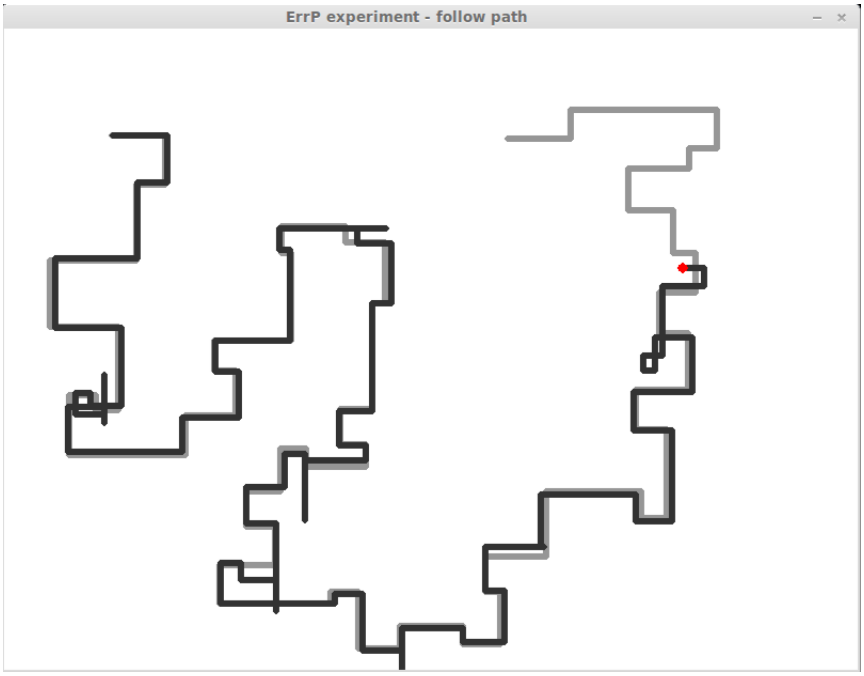


Figure 4.7: Graphical interface of another ErrP experiment.

In an attempt to verify the absence or presence of ErrPs, we implemented the experiment with another control scheme as well. In this setup, the subject is presented with the screen shown in Figure 4.7 and is instructed to guide the red dot on top of the gray path. We believe this interface gives a clearer indication of errors and makes a bigger commitment to executing the task successfully, and may produce errors that affect the subject to a greater extent. The implementation of a slightly different setup is also motivated by having a wider basis to conclude upon.

The program for this setup use the same procedure, and follows the same protocol shown in Figure 4.4, with the exception that the UR5 robot is not connected and the user instead controls the red dot. Additionally, we have written a simple path generator that records control sessions and saves the set of points to a path repository. Paths are selected randomly from this repository upon program execution.

4.2.4 Results

Three healthy subjects with age 24-27 years participated in the experiment and did three trials in each setup. The duration of a trial is 60 seconds, the time interval before the program starts

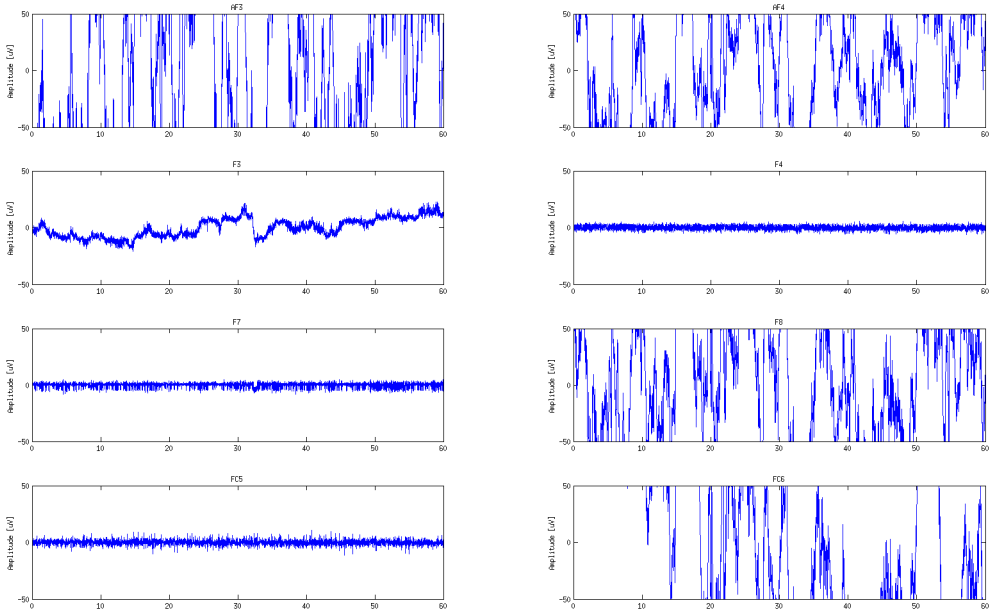


Figure 4.8: Raw EEG from the frontal lobe area in the robot ErrP experiment.

producing errors is 10 second, the cooldown between errors is 5 seconds, and the chance of producing an error during an allowed interval was set to $p = 0.3$.

Before we present the results, we wish to emphasize once again that the state of the Emotiv EPOC headset has decayed drastically during the work done in this thesis. As for the condition of the headset at the time of conducting the experiments, despite efforts to improve the quality of the channels in the frontal lobe section before sessions, is really not good enough as a basis for conclusions. The headset was also damaged during a trial; while mounting it on to a subject (with a rather wide head) the curved part broke, such that the headset lost tension and is not placed as firmly as it should.

Figure 4.8 shows the raw EEG of a trial from 8 channels in the frontal lobe section. The other 17 trials have similar characteristics. Again, most channels are either dead or contain large fluctuations, and the only acceptable channel to use is F3, even though also this contains strange characteristics. Figure 4.9 shows the results of taking the grand average of all 1-second segments of produced errors, for each subject, in both setups. The grand average is also smoothed out using a moving average with a 5-step span.

There are no indication of the presence of ErrPs in these results. The hypothesis of this exper-

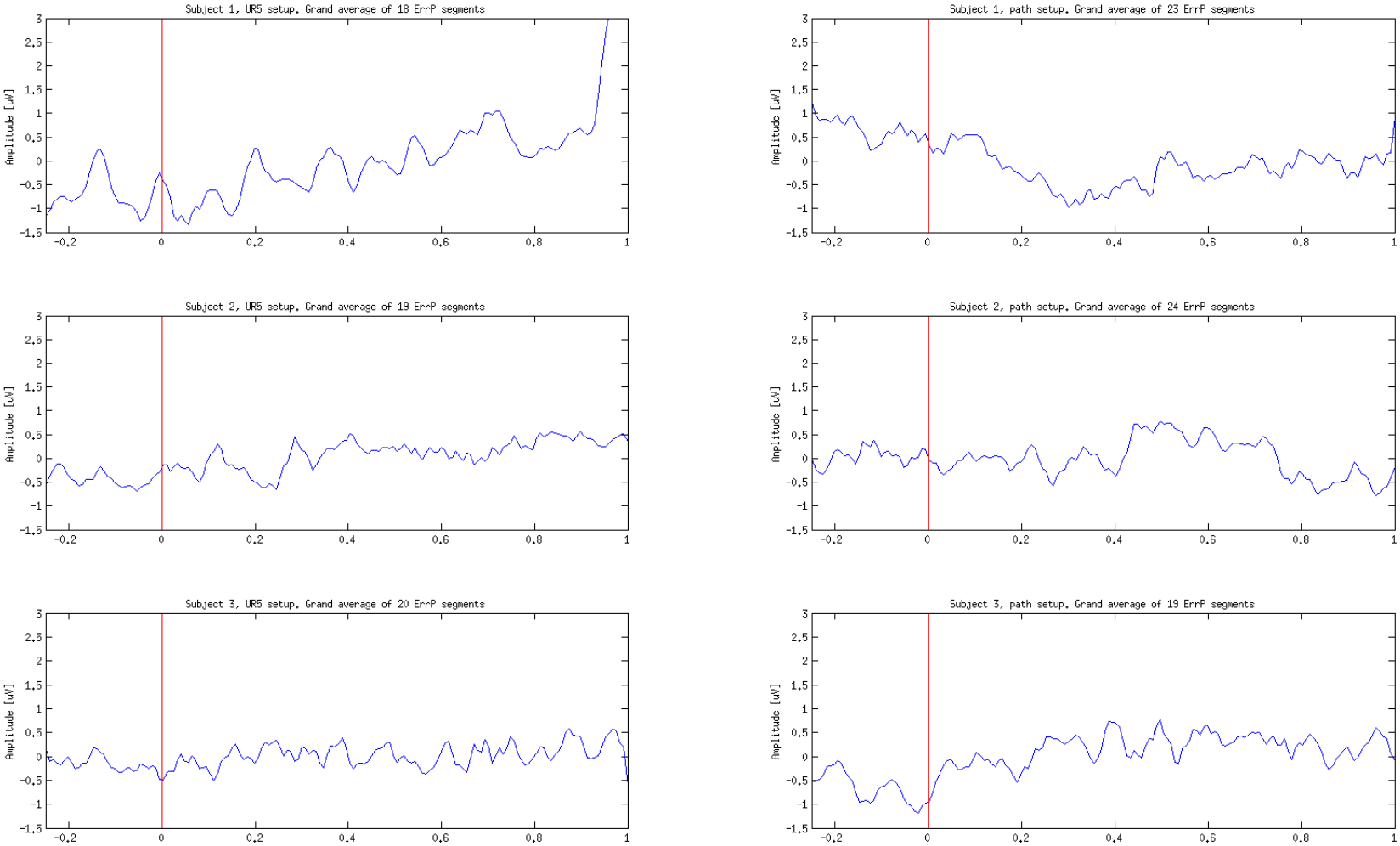


Figure 4.9: Grand average of ErrP segments from each subject in two ErrP experimental setups.

iment was whether ErrPs are detectable in real-life applications using a consumer grade headset for EEG measurements. Because of the integrity of the results, we will not deny or verify whether this hypothesis is true or not. Furthermore, we suggest that these experiments, or similar setups, are repeated using EEG equipment in appropriate condition.

4.3 Implementation of the SHDCRF model

The CRF, HCRF and LDCRF models have shown superior performance in standard BCI classification tasks (Saa and Cetin (2011); Saa and Çetin (2012)). However, we believe there is room for further improvements by relaxing the constraints on hidden variables in the LDCRF model. The SHDCRF model solves this and allows the hidden states to have shared, sparse beliefs between class variables.

In this section, we first present the main library we have used as a foundation for our implementation. Then we describe the development process towards our final implementation and present the results on a common BCI classification task.

4.3.1 HCRF2.0b

Discriminative sequence labeling frameworks can be quite complex compared to a trivial classifier. As a starting point for our implementation, we have chosen to fork the library HCRF2.0b[3] which implements the CRF, HCRF and LDCRF models in the C++ programming language. The library is developed by the authors of Quattoni and Wang (2007) and Morency et al. (2007), among others. HCRF2.0b also offers Matlab and Python interfaces. We will use Matlab running on an Ubuntu-based distribution (Linux Mint) for manipulating EEG data and to produce the necessary features, therefore we will primarily use the Matlab interface.

The library uses a class inheritance architecture, and all the modules for the different models typically originate from a superclass. A brief description of the top-level classes follows.

Toolbox acts as the main interface (API) for external calls, and exposes the necessary functions for initialization, set/get parameters, train and test a model.

[3] Available from http://sourceforge.net/projects/hcrf/

Dataset holds the data for training and testing. Instances of this class reside outside the toolbox and is passed in as a pointer when training and testing a model.

Model is the internal container for model variables. This class have variables that define the graphical structure and dynamics of the model, as well as the weights itself. It also holds the penalization constants used in the regularization terms.

Optimizer implements the optimization routine, and holds optimization specific control variables.

Gradient implements the function to compute the gradient in parallel using a map-reduce technique. During the map stage it computes all the local gradients corresponding to one parameter, and in the reduce stage the local gradients are summed together. The function that computes the gradient for a whole dataset returns the value of the objective function.

InferenceEngine has the main responsibility for belief propagation, and implements methods to estimate conditional distributions and partition functions.

FeatureGenerator works as a feature register. It keeps track of metadata that describe the feature types and features functions for a given model, and generate lists of virtual features upon request.

Evaluator implements inference of the most likely labels, and is typically called during testing. It also provides a function to compute the function error that are used by some optimizers.

Figure 4.10 illustrates the simplified top-level program flow for a general training procedure, and also shows the role of specific modules and how they interact. The dependencies are quite high across the lower-level routines that does the heavy lifting. This makes the low-level program flow somehow intertwined, so we have omitted some details in this illustration.

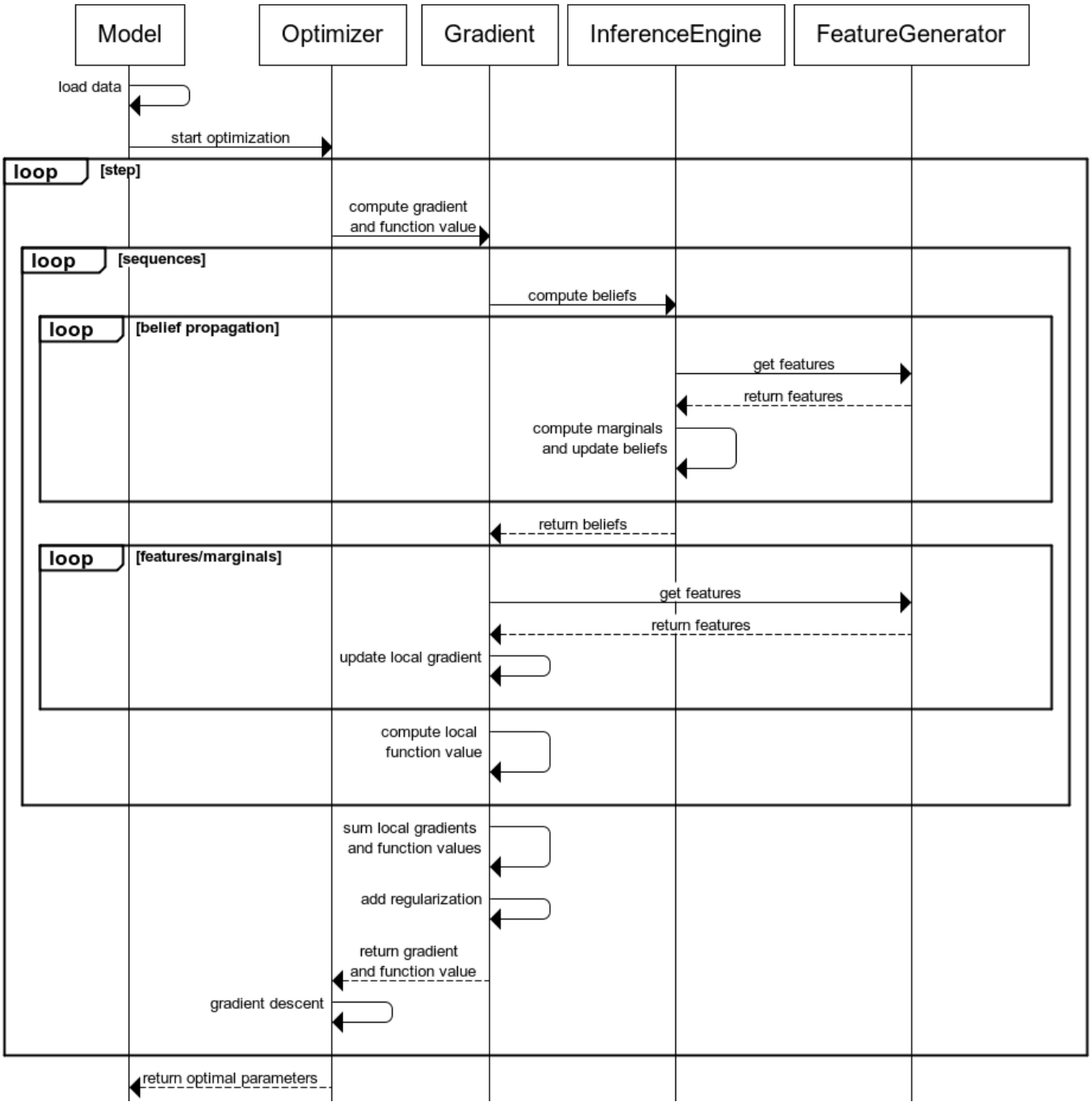


Figure 4.10: Top-level program flow of the training procedure in CRF models.

4.3.2 Implementation and code contribution

Most of our implemented modules follow the architecture imposed by the library and inherit from their respective superclass. However, although the SHDCRF is equal to the LDCRF in graphical structure, the SHDCRF requires quite different routines than all the other models, because it aims to learn the relationship between three dependent random variables (Y, H, X) instead of two ((Y, X) for CRF and (H, X) for HCRF and LDCRF). Therefore, we must override quite many of the methods provided by the superclasses. Additionally, this have lead to some workarounds of the program flow imposed by the library. We focus on describing the changes made in the training and testing stage of our model, as the other parts are somehow trivial.

We use Figure 4.10 as reference and briefly go through the significant parts of the training routine for the SHDCRF. First of all, the optimizer class for our model use an L-BFGS (Liu and Nocedal (1989)) implementation provided by an external library, which is independent of HCRF2.0b. The L-BFGS minimize the objective function by default, thus the term for the objective function (3.23) and the terms for the partial derivatives (3.28) and (3.29) are multiplied by -1. The `GradientSHDCRF` class implements two functions for computing gradients: One for sequences and one for the whole dataset. The `computeGradient` function for the dataset calls the `computeGradient` function for sequences in parallel and joins the threads, then sums up each local contribution before adding the regularization terms and the conditional entropy in (3.24). Each call to the `computeGradient` for the sequences computes the terms for a single sequence i in (3.28) and (3.29).

Our implementation `InferenceEngineFBShared` estimate the four conditional distributions in (3.28) and (3.29) by the forward-backward algorithm described in (3.32) and 3.33, except that we take the logarithm of (3.30) and (3.31) instead. We have added the feature type `SharedFeatures` to `FeatureGenerator` that defines the feature functions g_k for the label variables, which are taken into account by `InferenceEngineFBShared`.

The `EvaluatorSHDCRF` infers the most likely sequence of labels given by (3.34). The first term in (3.35) is calculated using (3.19) and the second term of (3.35) is estimated by `InferenceEngineFBShared`. `EvaluatorSHDCRF` returns the whole confidence for each y_t, not only y_t^*.

4.3.3 Testing

The tests have been performed on dataset V from BCI Competition III[4], which is a common dataset to measure the performance of classifiers. This dataset contains data from 3 normal subjects during 4 sessions. The subjects perform three mental tasks:

1. Imagination of repetitive self-paced left hand movements (class 0).
2. Imagination of repetitive self-paced right hand movements (class 1).
3. Generation of words beginning with the same random letter (class 2).

The dataset also have precomputed PSD features for the band 8-30 Hz, with a frequency resolution of 2 Hz, for the 8 centro-parietal electrodes C3, Cz, C4, CP1, CP2, P3, Pz and P4. We have used a subset of these PSD features for testing. Additionally, the data has been sliced to appropriate size to reduce the time required for training each model. Furthermore, the model parameters Λ are initialized randomly in the interval $[-0.5..0.5]$ and the number of hidden states have been varied from 2 to 6.

Initial testing revealed that the implementation of SHDCRF does not behave as expected. More specifically, the optimization routine exhibit strange behavior and does not always reach a minimum. We found that the initial values of the parameters Λ have a huge impact on the final solution, and that the optimization only *sometimes* terminates such that the model performs acceptably on the testing set.

This strongly suggest that the objective function (3.23) indeed is non-convex. However, the L-BFGS routine returns a variety of strange return codes. The majority of these return codes say that there is something wrong with the line-search procedure that the BFGS use to find the next step, including that the line-search step is too small, too big, or that is has reached the maximum iterations for finding a suitable step. Moreover, changing the control parameter that determine these values did not solve this issue. Therefore, this could also indicate a certain disharmony between the value of the objective function and the values in the gradient vector.

We investigate this matter further by observing the behavior of the conditional entropy and experimenting with the sparsity factor α. As stated by Shen et al. (2011): "...it can be concluded

[4]The dataset is publicly available. A full description of the dataset and directions for downloading can be found at http://bbci.de/competition/iii/desc_V.html

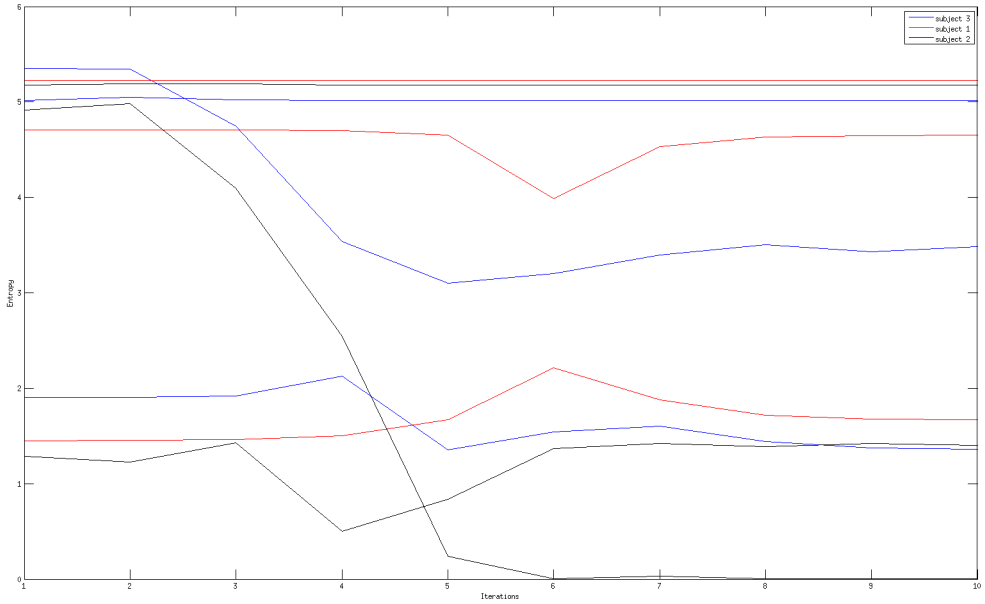


Figure 4.11: Conditional entropy vs iterations.

that the sparsity condition in SHDCRF is essential to avoid trapping into the bad local maxima and ensuring good experimental results." Figure 4.11 shows how the conditional entropy $H_\Lambda(Y|H)$ evolves through the steps of the BFGS routine, with multiple runs on the data from all three subjects. This plot summarize the general behavior of the optimizer: For most runs, the optimizer get stuck somewhere and cannot manage to minimize the conditional entropy and maximize the model distribution, but once in a while it finds a good solution. We also found that different values for the sparsity factor α did not affect the course of the entropy significantly, and it certainly did not get the classifier out of the bad states. Finally, we also used another optimization algorithm, namely the conjugate gradient method, which also did not change this behavior.

It is hard to identify the true reason for this issue, because it is a complex model that rely on many factors. The size of the parameter vector Λ are in the range 50-100, and it is quite difficult to visualize the landscape of an objective function of 50-100 dimensions. Anyhow, based on the test results and error probing we suggest two probable causes for this behavior:

- The characteristics of the features produced from EEG data might be quite different than those produced in the user intent understanding task in Shen et al. (2011). Consequently,

the model may simply not work with the objective function of an EEG sequence labeling task.

- Because Shen et al. (2011) is the only publication that describe this method, the proposed model is open to misinterpretation. Although we have verified the equations and expressions given in this paper, there might be something wrong in our implementations.

4.3.4 SHDCRF*

It is clear that a BCI cannot rely on an unstable model: We cannot accept a model that only works sometimes, especially when there are high computational demands for training. The test results may imply that the model proposed in Shen et al. (2011) does not work well with EEG data. However, we still believe that the model itself is good, and that the training procedure is the main issue.

We propose another version of the SHDCRF proposed in Shen et al. (2011) whose training procedure is guaranteed to converge. This model seeks to learn the model parameters of the SHDCRF in a different way, and relies on two consecutive stages to learn the optimal parameters. The underlying graphical structure of the proposed model can still be described as a dynamic conditional random field, with an intermediate hidden layer, and a sparse relation imposed on the distribution $p(\mathbf{y}|\mathbf{h})$. Therefore, for the convenience of distinguishing the two models, we denote this model by SHDCRF*.

The proposed SHDCRF* model is in many ways a hybrid of the LDCRF and the SHDCRF. The SHDCRF* relies on the same initialization and training as the LDCRF, however, after LDCRF training it relaxes the constraints on the hidden variables and enters a second phase of training, where it seeks to optimize the distribution $p(\mathbf{y},\mathbf{h})$. The two stages of the training can be summarized as:

1. At initialization, impose constraints on the hidden states, such that each h_j is a member of a separate set H_{y_j} of possible hidden states for the class label y_j. Train the parameters λ_k using the same training scheme as the LDCRF.

2. First, infer the beliefs $\mathbf{h}_\lambda^{(i)} = p_\lambda(\mathbf{h}|\mathbf{x})$. This is only required to do once per sequence, as the belief states of each h_t is not subject to change. Then relax the constraints on the hidden

states, and maximize the objective function

$$L(\beta) = \sum_{i=1}^{N} \log p_\beta(\mathbf{y}^{(i)}, \mathbf{h}_\lambda^{(i)}) - \frac{||\beta||^2}{2\sigma^2} - \alpha H_\lambda(Y|H) \tag{4.1}$$

The term $p_\beta(\mathbf{y}^{(i)}, \mathbf{h}_\lambda^{(i)})$ is given by (3.19), as before. We restate it here for convenience, using our new notation:

$$p_\beta(\mathbf{y}|\mathbf{h}_\lambda) = \frac{1}{Z(\mathbf{h}_\lambda)} \exp\left\{ \sum_{k=1}^{p} \beta_k \mathbf{G}_k(\mathbf{y}, \mathbf{h}_\lambda) \right\} \tag{4.2}$$

However, now the inferred hidden states are fixed and will not change, and the class labels $\mathbf{y}$ only depend on $\mathbf{h}_\lambda$. This yields the partial derivatives

$$\frac{\partial L(\beta)}{\partial \beta_k} = \sum_{i=1}^{N} \sum_{t=1}^{T^{(i)}} g_k(y_t^{(i)}, h_{\lambda,t}^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{h}_\lambda) - \frac{\beta_k^2}{2\sigma^2} + p(\dot{y}|\dot{h}_\lambda) \left(\beta_k - \sum_{\ddot{y} \in Y} p(\ddot{y}|\dot{h}_\lambda) \beta_{K(\ddot{y}, \dot{h}_\lambda)} \right) \tag{4.3}$$

where $Z(\mathbf{h}_\lambda)$ is given by (3.21). Finally, finding the most likely solution for position t in a new test sequence is the same as in (3.34), and given by

$$y_t^* = \arg\max_{y_t} \sum_{h_{\lambda,t} \in H} p_\beta(y_t|h_{\lambda,t}) p_\lambda(h_{\lambda,t}|\mathbf{x}) \tag{4.4}$$

The important aspect to note is that the transition and state distribution of the hidden states will not change after the first stage of the training. This enables us to treat the hidden distributions as fixed during the second stage. Also note that the effect of the sparsity condition is questionable here. Since the first part of the training finds optimal structure and transitions of hidden states with a fixed initial distribution $p(\mathbf{y}, \mathbf{h})$, the model is "locked" into this configuration, and the model may not benefit from changing the distribution $p(y_t|h_{\beta,t})$ significantly.

This model have the same properties as the SHDCRF proposed in Shen et al. (2011), but guarantees a global optimal solution on every run. Moreover, it improves the LDCRF with a minimal addition in computational expense, since we are only required to infer N belief propagations.

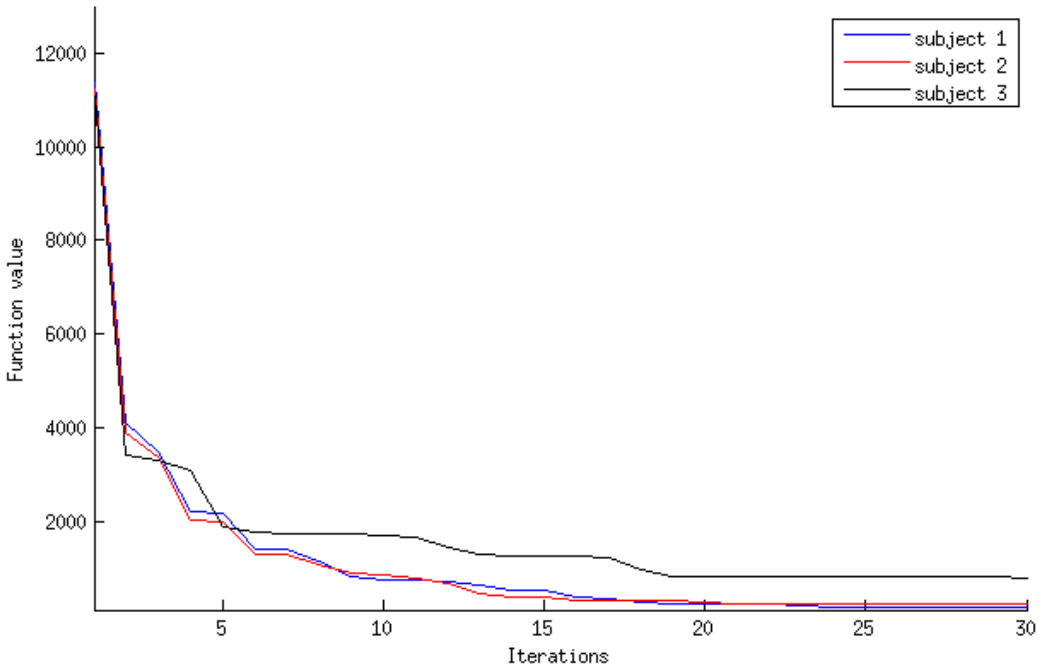


Figure 4.12: Convergence of the objective function in the first training stage of SHDCRF*.

4.3.5 Results

We have tested and compared the CRF, LDCRF and our proposed SHDCRF* on the same dataset that was used during testing (dataset V from BCI Competition III). We have used the same subset of features that are described in the publication by Saa and Çetin (2012), where the performance of CRF and LDCRF on the same dataset are compared to other classifiers. Saa and Çetin (2012) first computes the average PSD values of the alpha, sigma and beta-bands (Table 2.1) and finds an optimal subset of features (bands+channels) for each subject by using an sequential floating feature selection (SFFS) method. The basic principle of SFFS is to add or remove features after each run, and choose the subset of features that maximize the results. As this is very computational demanding for CRF-based models and would require significant time or computational power, we use the features from Saa and Çetin (2012) directly.

We trained the SHDCRF* model with the described procedure, and Figure 4.12 shows that the objective function converge during the first stage of the training, for all three subjects. The whole training procedure finishes in around 10 minutes. Furthermore, the SHDCRF* have 6 hidden states for all subjects and the sparsity factor is set quite low to allow larger shifts in the

distribution, with a value of $\alpha = 0.1$. The CRF and LDCRF models are also trained using the same features and parameters as in Saa and Çetin (2012), where the LDCRF have 2 hidden states per label (for a total of 6 hidden states).

Our classification results are given in Table 4.4, where each score is the success rate given in percent. The proposed SHDCRF* model compares favorably to the CRF and LDCRF models, and achieves superior performance on all subjects. We did not achieve the same scores as in Saa and Çetin (2012) although we used the same features and parameters. However, the paper mention that they have achieved the scores using additional post-processing methods, and states that the results were "...obtained by cross validation in the training set using CRF and LDCRF". We will not concern ourself with performing this post-processing stage, as we are not interested in achieving record-breaking scores by voting between classifiers, but rather want to verify that our model is superior to the CRF and LDCRF. Additionally, we are interested in developing a good classifier for practical BCIs, and using multiple CRFs and voting between classifiers in real-time is not feasible. However, we will mention that the LDCRF and the results from Saa and Çetin (2012) in Table 4.5 are currently the best results on this dataset. Considering that our proposed SHDCRF* model have obtained better scores than the LDCRF in our own runs, it is a strong indication of that we have developed a top-performing framework for EEG classification.

Figure 4.13 gives a visual representation of the results, where predicted labels are red, and the true labels are blue. Subject 3 is of particular interest. As the results in Table 4.4 shows, this is the subject with the lowest success rate, but it is also for this subject that the SHDCRF* yields the highest performance gain over the other classifiers. Indeed, we can confirm the performance gain visually by comparing the rightmost plots in Figure 4.13. This is a most desirable result as one of the main challenges to make BCIs more reliable is to reduce performance variations across users.

In Table 4.6, Table 4.7 and Table 4.8 we have given the full conditional distributions for subject 1, 2, and 3 respectively. In a LDCRF with 2 hidden states per label, this distribution is given as double diagonals with 1's, which means we can see how the SHDCRF* have shifted the distributions. Again, we turn our attention to subject 3. From Figure 4.13 we can see that both the CRF and LDCRF mistakenly predicts the true label 0 as label 2. Looking at the conditional distribution for subject 3 in Table 4.8, we can see that the SHDCRF* has made a large shift for the

Table 4.4: Our classification results on dataset V from BCI Competition III.

Subject	S01	S02	S03	Average
CRF	58.99	71.20	38.40	56.20
LDCRF	60.62	69.04	50.67	60.11
SHDCRF*	**60.84**	**74.42**	**58.52**	**64.59**

Table 4.5: Classification results in Saa and Çetin (2012) on dataset V from BCI Competition III.

Subject	S01	S02	S03	Average
CRF	89.34	78.08	59.73	75.72
LDCRF	91.55	83.89	59.30	78.25

hidden states h_6 and h_5, and a major shift for h_3, whereas all shifts concerns the beliefs of label 0 and 2. All this indicate that for some reason, the LDCRF failed to find a good relation between hidden states for subject 3. Moreover, it shows by a direct example how the SHDCRF* finds a more optimal solution.

The previous example illustrates another advantage with the SHDCRF*. As we have pointed out in chapter 3, the motivation behind introducing hidden states in the model is their ability to model sub-structure in the EEG that are impossible to label during trials. The conditional distribution optimized by the SHDCRF* reveal information about how the underlying sub-structures relates to each other. For instance, we can conclude by examination of the Table 4.6, Table 4.6 and Table 4.6 that the hidden states have high confidence in separating left hand and right hand motor imagery, but have lower confidence in separating motor imagery and word association. As we know that left and right hand motor imagery elicit a reduction of the PSD in the mu-band at opposite hemispheres, and that the features are based on the PSDs including this frequency range, we are not surprised. However, suppose we label other mental strategies that are not well understood, then the SHDCRF* can provide valuable information about how the hidden states relate. This way, the model could also act as a framework for exploring the sub-structure of mental states, and ultimately make room for labeling EEG with higher resolution.

Table 4.6: Sparse conditional distribution $p(\mathbf{y}|\mathbf{h})$ for subject 1 in the SHDCRF* model.

	left hand MI	right hand MI	word association
h_1	0.99	0.01	0.00
h_2	0.99	0.01	0.00
h_3	0.01	0.96	0.03
h_4	0.03	0.96	0.01
h_5	0.02	0.01	0.97
h_6	0.08	0.27	0.65

Table 4.7: Sparse conditional distribution $p(\mathbf{y}|\mathbf{h})$ for subject 2 in the SHDCRF* model.

	left hand MI	right hand MI	word association
h_1	0.56	0.10	0.34
h_2	0.07	0.86	0.07
h_3	0.06	0.85	0.09
h_4	0.27	0.54	0.19
h_5	0.00	0.27	0.73
h_6	0.00	0.24	0.76

Table 4.8: Sparse conditional distribution $p(\mathbf{y}|\mathbf{h})$ for subject 3 in the SHDCRF* model.

	left hand MI	right hand MI	word association
h_1	0.91	0.08	0.01
h_2	0.97	0.02	0.01
h_3	0.01	0.06	0.93
h_4	0.09	0.78	0.13
h_5	0.27	0.11	0.62
h_6	0.53	0.01	0.46

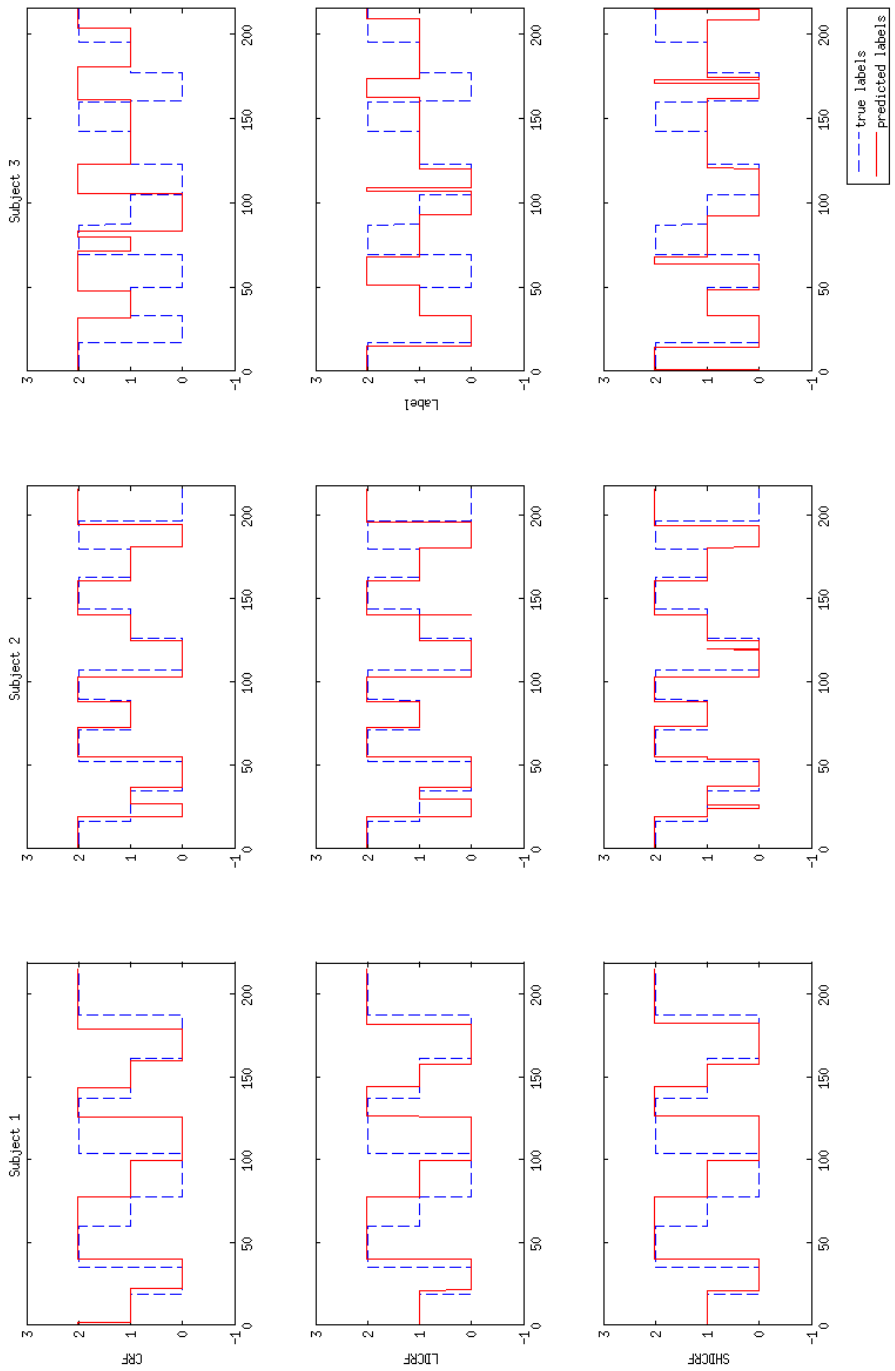


Figure 4.13: Visual presentation of the classification results for the CRF, LDCRF and SHDCRF* models for all subjects.

Chapter 5

Conclusions and future work

BCIs still suffer from poor performance despite the increased interest in the field of BCI research recent years. BCIs are not practical to use quite yet. There are several challenges that must be solved before BCIs can serve as solutions for regular consumers.

We have studied two novel concepts that can boost the performance, usability and reliability of BCIs intended at control applications: The utilization of ErrPs and using a CRF-based model for EEG classification, and we have argued for why we believe these concepts are suitable for such BCIs.

Studies have successfully detected single trial ErrPs in laboratory environments, however, we argued for that ErrP detection in control applications, using consumer grade EEG equipment, are not fully explored. Additionally, the CRF-based model we suggest has never been employed in a BCI context. With this motivation, we have stated our objective as to verify the potential of these concepts through experimentation and implementation.

ErrP experiments

Our goal was to detect ErrPs in a realistic BCI application using a consumer grade headset for EEG measurements. We have designed an experimental setup with two different control schemes, whereas one is to control an UR5 robot and the other is to control a dot through a path on a computer screen.

We could not verify the presence of ErrPs in the recorded trials. However, the available head-

set suffer from poor condition, and the EEG quality have decayed at a steady rate throughout our previous work in Soukup (2013) up until the experiments done in this thesis. We see the EEG measurements as unreliable and refrain from drawing conclusions upon the results.

SHDCRF* model

We implemented the SHDCRF described in Shen et al. (2011), which seeks to learn both the intrinsic dynamic of observable states and a sparse relation between the latent variables and the class variables. However, we experienced problems during the training of this model. As the paper states, the objective function is not convex and relies heavily on an entropy-based regularization term in order to reach a global optimum. In addition, there is currently only one publication that describe this model, where it is used on an entirely different sequence labeling task. Consequently, we are uncertain of the model's compatibility with other data domains such as EEG, and it also makes the model more receptive to misinterpretation.

We developed a new model that solves the training problem by splitting the training procedure into two consecutive stages. Our proposed SHDCRF* model benefits from the robust training procedure of an LDCRF, and inhabits the ability to share beliefs on class labels among the hidden states, as in the SHDCRF described by Shen et al. (2011).

The SHDCRF* was superior to the regular CRF and the LDCRF on a common BCI classification task, and we showed by direct example why the model performs better. Moreover, we pointed out how the model can be used as a tool for explaining the relationship between hidden states. This completes our objective, as we argue for that the SHDCRF* is a framework most suitable for BCI applications.

Future work

An immediate goal is to verify the presence of ErrPs in typical BCI control setups. We suggest that the same or similar experiments is re-taken with a consumer grade headset in appropriate condition. Assuming that ErrPs can be detected, another important matter is to develop a reinforcement learning algorithm for CRF-based models.

We also suggest further experiments of the SHDCRF* model, preferably in a real-time setup, to tune the model and measure performance in real-world BCI control applications.

Appendix A

Attachments

The folders in the attachments are structured as in Figure A.1. A short description of each top-level folder follows:

- `bci` is the implementation of the two setups in the ErrP experiment.
- `data` is the directory for EEG data from BCI Competition III and recordings from the ErrP experiments. BCI Competition data is left out because it is ~370 Mb in size.
- `fig` was used as a directory for plots during development. The program `ploterrp.py` generate plots of trials (`.mat`-file is given as argument) and saves a plot of each channel to this directory.
- `HCRF` is our fork of the library HCRF2.0b, and contains the implementation of the

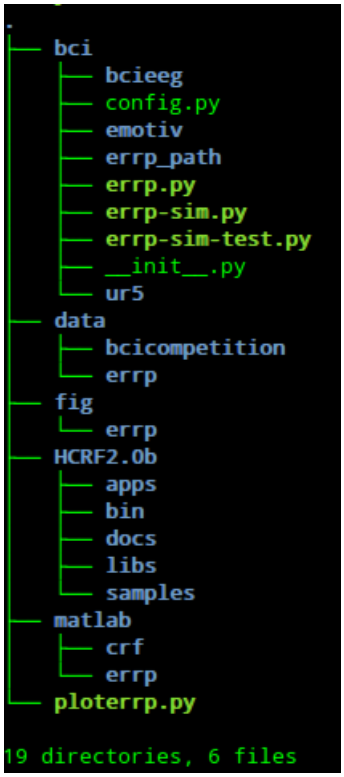


Figure A.1: Files and folders in attachments.

probabilistic frameworks tested in this thesis.

- `matlab` contains the Matlab-scripts for the analyses associated with the ErrP experiment and CRF-based models.

Appendix B

Acronyms

AAV Average Absolute Value

ACC Accuracy

ANN Artificial Neural Networks

ASL Amyotrophic Lateral Sclerosis

BCI Brain-Computer Interface

BP Band Power

CRF Conditional Random Field

CSP Common Spatial Pattern

ECoG Electrocorticography

EEG Electroencephalography

EMG Electromyography

EOG Electrooculography

ERD Event-Related Desynchronization

ERP Event-Related Potentials

ERS Event-Related Synchronization

FIR Finite Impulse Response

fMRI functional Magnetic Resonance Imaging

FSM Finite State Machine

HCI Human-Computer Interaction

HMM Hidden Markov Model

IIR Infinite Impulse Response

ITR Information Transfer Rate

LDA Linear Discriminant Analysis

LDCRF Latent-Dynamic Conditional Random Field

MEG Magnetoencephalography

NIRS Near-Infrared Spectroscopy

PCA Principal Component Analysis

PSD Power Spectral Density

QDA Quadratic Discriminant Analysis

VEP Visual Evoked Potential

SCP Slow Cortical Oscillations

SHDCRF Sparse Hidden-Dynamic Conditional Random Field

SMR Sensory Motor Rhythm

SNR Signal-to-Noise Ratio

SSVEP Steady-State Visual Evoked Potentials

SVM Support Vector Machine

Bibliography

Bashashati, A. and Fatourechi, M. (2007). A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals. *Journal of Neural ...*, 4.

BCI Competitions. BCI Competitions website. \url{http://www.bbci.de/competition/}.

Berger, T. (2008). *Brain-Computer Interfaces: An international assessment of research and development trends*.

Blakely, T., Miller, K., Ojemann, J., and Rao, R. (2013). Exploring the Cortical Dynamics of Learning by Leveraging BCI Paradigms. pages 53–60.

Brumberg, J., Guenther, F., and Kennedy, P. (2013). An Auditory Output Brain–Computer Interface for Speech Communication. *Brain-Computer Interface ...*, 52(4):367–379.

Carrino, F. and Dumoulin, J. (2012). A self-paced BCI system to control an electric wheelchair: Evaluation of a commercial, low-cost EEG device. *... (BRC), 2012 ISSNIP*.

Chavarriaga, R. and Biasiucci, A. (2010). Adaptation of hybrid human-computer interaction systems using EEG error-related potentials. *... in Medicine and ...*, 2010:4226–9.

Cinar, E. and Sahin, F. (2010). A study of recent classification algorithms and a novel approach for EEG data classification. *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 3366–3372.

Dornhege, G. and Millán, J. d. R. (2007a). Error-Related EEG Potentials in Brain-Computer Interfaces. *ieeexplore.ieee.org*, 3928.

Dornhege, G. and Millán, J. d. R. (2007b). Evaluation Criteria for BCI Research. *ieeexplore.ieee.org*.

Dunne, S., Leeb, R., Nijholt, A., and Millán, J. (2008). Towards Practical Brain-Computer Interfaces. *Springer*.

Faller, J., Torrellas, S., Miralles, F., Holzner, C., Kapeller, C., Guger, C., Bund, J., Müller-Putz, G. R., and Scherer, R. (2012). Prototype of an auto-calibrating, context-aware, hybrid brain-computer interface. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2012:1827–30.

Frey, B. and Jojic, N. (2005). A comparison of algorithms for inference and learning in probabilistic graphical models. *Pattern Analysis and Machine Intelligence, ...*, 27(9):1–25.

Graimann, B., Allison, B., and Pfurtscheller, G. (2010). *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*. Frontiers collection. Springer.

Grosse-wentrup, M. and Sch, B. (2013). A Review of Performance Variations in SMR-based Brain-Computer Interfaces (BCIs). pages 1–15.

Guger, C., Allison, B., and Edlinger, G. (2013). *Brain-Computer Interface Research: A State-of-the-Art Summary*.

Hamadicharef, B. (2010). Brain-Computer Interface (BCI) literature-a bibliometric study. *... and their Applications (ISSPA), 2010 10th ...*, 1(Isspa):626–629.

Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. a., Saleh, M., Caplan, A. H., Branner, A., Chen, D., Penn, R. D., and Donoghue, J. P. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–71.

Iturrate, I., Montesano, L., and Minguez, J. (2010). Robot reinforcement learning using EEG-based reward signals. *2010 IEEE International Conference on Robotics and Automation*, pages 4822–4829.

Kanoh, S., Murayama, Y.-M., Miyamoto, K.-I., Yoshinobu, T., and Kawashima, R. (2009). A NIRS-based brain-computer interface system during motor imagery: system development and online feedback training. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2009:594–7.

Khorshidtalab, a. and Salami, M. J. E. (2011). EEG signal classification for real-time brain-computer interface applications: A review. *2011 4th International Conference on Mechatronics (ICOM)*, (May):1–7.

Kus, R., Valbuena, D., Zygierewicz, J., Malechka, T., Graeser, A., and Durka, P. (2012). Asynchronous BCI based on motor imagery with automated calibration and neurofeedback training. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 20(6):823–35.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001(Icml):282–289.

LaFleur, K., Cassady, K., Doud, A., Shades, K., Rogin, E., and He, B. (2013). Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain-computer interface. *Journal of neural engineering*, 10(4):046003.

Liu, D. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45:503–528.

Liu, Y.-H., Huang, C.-W., and Hsiao, Y.-T. (2013). Comparsion of methods for a motor imagery-based two-state self-paced brain-computer interface. *2013 International Conference on Advanced Robotics and Intelligent Systems*, pages 174–178.

Lotte, F., Congedo, M., Lécuyer, a., Lamarche, F., and Arnaldi, B. (2007). A review of classification algorithms for EEG-based brain-computer interfaces. *Journal of neural engineering*, 4(2):R1–R13.

Mansor, W. and Khuan, L. Y. (2010). A Review of Signal Processing in Brain Computer. (December):443–449.

Marshall, D., Coyle, D., Wilson, S., and Callaghan, M. (2013). Games, Gameplay, and BCI: The State of the Art. 5(2):82–99.

Mladenov, T. (2012). Accurate motor imagery based dry electrode brain-computer interface system for consumer applications. *Consumer Electronics (...*, pages 1–4.

Morency, L., Quattoni, A., and Darrell, T. (2007). Latent-dynamic discriminative models for continuous gesture recognition. *...and Pattern Recognition, ...*

Oliver, G., Sunehag, P., and Gedeon, T. (2012). Asynchronous brain computer interface using hidden semi-Markov models. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2012:2728–31.

Quattoni, A. and Wang, S. (2007). Hidden conditional random fields. *IEEE transactions on ...*, 29(10):1848–1853.

Saa, J. D. and Çetin, M. (2012). Discriminative methods for classification of asynchronous imaginary motor tasks from EEG data. *IEEE Transactions on Neural ...*, 21(5):716–724.

Saa, J. D. and Cetin, M. (2011). Hidden conditional random fields for classification of imaginary motor tasks from EEG data.

Sepulveda, F., Dyson, M., Gan, J. Q., and Tsui, C. L. (2007). A Comparison of Mental Task Combinations for Asynchronous EEG-Based BCIs. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2007:5055–8.

Shen, Y., Yan, J., Yan, S., Ji, L., Liu, N., and Chen, Z. (2011). Sparse hidden-dynamics conditional random fields for user intent understanding. *Proceedings of the 20th international conference on World wide web - WWW '11*, page 7.

Shenoy, P., Krauledat, M., Blankertz, B., Rao, R. P. N., and Müller, K.-R. (2006). Towards adaptive classification for BCI. *Journal of neural engineering*, 3(1):R13–23.

Soukup, M. (2013). Brain-Computer Interface In Control Systems. (December).

Sutton, C. (2012). An Introduction to Conditional Random Fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

Sutton, C. and Mccallum, A. (2006). 1 An Introduction to Conditional Random Fields for Relational Learning.

Sutton, C., McCallum, A., and Rohanimanesh, K. (2007). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine . . .*, 8:693–723.

Tan, D. and Nijholt, A. (2010). *Brain-Computer Interfaces: applying our minds to human-computer interaction*.

Velliste, M., Perel, S., Spalding, M. C., Whitford, A. S., and Schwartz, A. B. (2008). Cortical control of a prosthetic arm for self-feeding. *Nature*, 453(7198):1098–101.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, pages 260–269.

Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., and Vaughan, T. M. (2002). Brain-computer interfaces for communication and control. *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology*, 113(6):767–91.

Wolpaw, J. R. and Boulay, C. B. (2010). Brain-Computer Interfaces. pages 29–46.

Wolpaw, J. R. and McFarland, D. J. (2004). Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 101(51):17849–54.

Yang, Q. and Fang, Y. (2007). Online and Offline Implementation of Time-frequency Template Matching Method for Classifying Motor Imagery in Brain Computer Interface. *. . . Imaging of the Brain and . . .*, (1):78–81.

Yong, X., Fatourechi, M., Ward, R. K., and Birch, G. E. (2012). Adaptive classification in a self-paced hybrid brain-computer interface system. *Conference proceedings : ... Annual Interna-*

tional Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference, 2012:3274–9.

Zou, L., Wang, X., Shi, G., and Ma, Z. (2010). EEG feature extraction and pattern classification based on motor imagery in brain-computer interface. *9th IEEE International Conference on Cognitive Informatics (ICCI'10)*, pages 536–541.