

A Brain-Computer Interface for the Classification of
Motor Imagery

by

ALEXANDRA CONSTANTIN

Andrea Danyluk, Advisor

A thesis submitted in partial fulfillment
of the requirements for the
Degree of Bachelor of Arts with Honors
in Computer Science

WILLIAMS COLLEGE

Williamstown, Massachusetts

April 27, 2007

Abstract

Recent advances in computer hardware, neuroscience, signal processing, and machine learning have made it possible to use human EEG signals or “brain waves” to communicate with an electronic device, such as a computer. Devices capable of this type of direct communication between a person’s brain and an external device are called brain-computer interfaces (BCIs).

In this thesis, I consider Brain-Computer Interface paradigms in which the subject needs to perform one of two or more mental activities according to a cue. The goal is to recognize patterns in the EEG signals associated with each mental activity and to use them to perform a task. For example, imagined movement of the left hand might be associated with movement of a mouse cursor to the left, while imagined movement of the right hand might be associated with movement of a mouse cursor to the right.

For this thesis I have implemented several variations of a deformable Markov model for time-series pattern matching. This is a model that has previously been used for detecting specific patterns or shapes in time-series data. The patterns of interest are modeled as K -state deformable Markov models where each state is responsible for the generation of a component of the overall shape using a state-based regression function. The distance between segments is modeled as a semi-Markov process, allowing flexible deformation time. Recognition of a pattern in a new time series is achieved by a dynamic programming algorithm.

I evaluate the implemented models on data provided at BCI Competitions II and III. I compare the results to those obtained by the researchers at Graz University and by the participants in the two competitions.

Contents

1	Introduction	6
1.1	Motivation	7
1.2	BCI Task and Data	8
1.3	Previous Approaches	8
1.4	Proposed Algorithm	9
1.5	Goals and Evaluation	9
1.6	Organization	9
2	Background	11
2.1	BCIs and Their Many Uses	11
2.2	Brain Activity Monitoring Methods	12
2.3	Mental Tasks	12
2.3.1	Evoked-Response BCIs	12
2.3.2	Operant Conditioning Based BCIs	13
2.4	Acquisition of EEG Signals	15
2.5	Preprocessing	16
2.6	Operating Protocols	16
3	Feature Extraction Procedures	17
3.1	Spatial Filters	17
3.2	Time-Frequency Analysis	18
3.2.1	Time Domain Analysis	18
3.2.2	Frequency Domain Analysis	19
3.2.3	Combining Time and Frequency Analysis	20
3.3	AR Model	20
3.3.1	ARMA Model	21
3.3.2	MAR Model	21
3.3.3	AAR	21
3.4	Nonlinear Feature Extraction Methods	23
4	Classification Algorithms	24
4.1	Linear Functions	24
4.2	Linear Discriminant Analysis (LDA)	24
4.3	Learning Vector Quantization (LVQ) and Dynamic Scaled Learning Vector Quantization (DSLQ)	25
4.4	Adaptive Logic Network	26
4.5	SVM	28
4.6	Evaluation of Classification Algorithms	28

5	BCI Task and Data	30
5.1	BCI Competition II Data Set	30
5.1.1	Task	30
5.1.2	Physical Set-Up	30
5.1.3	Experimental Design	31
5.1.4	Data Format	31
5.2	BCI Competition III Data Set	31
5.2.1	Task	32
5.2.2	Physical Set-Up	33
5.2.3	Experimental Design and Data Format	33
6	Deformable Markov Model Templates for Waveform Pattern Matching in the context of BCIs	34
6.1	General Description	34
6.2	Bottom-Up Segmentation	35
6.3	Hidden Markov Models	36
6.4	Constructing a Waveform Model from the Data	37
6.4.1	Bottom-Up Segmentation for Computing the Probability of an Output Sequence	38
6.5	Producing a Classification	38
6.6	Variations of the Model	38
6.6.1	Transforming the Data	39
6.6.2	Model 1	39
6.6.3	Model 2: Dynamic Programming Algorithm for Computing the Most Likely State Sequence	40
6.6.4	Model 3	40
6.6.5	Model 4	40
7	Evaluation Metrics	42
7.1	Accuracy	42
7.2	Mutual Information	42
8	Experiments and Results	44
8.0.1	Model 3	44
8.0.2	Model 5	44
8.0.3	Models 6 and 7	44
8.0.4	Model 9	45
8.1	BCI Competition 2003 Data Set	45
8.2	BCI Competition 2005 Data Set	46
8.3	Comparison to Other Algorithms	47
8.3.1	Adaptive Autoregressive Parameters and Linear Discriminant Analysis	47
8.3.2	Group 1	48
8.3.3	Group 2	48
8.3.4	Group 3	49
8.3.5	Group 4	49
8.3.6	Group 5	49
8.3.7	Group 6	49
8.3.8	Group 7	50
8.3.9	Group 8	50

8.3.10 Comparison	50
8.3.11 BCI 2005 Baseline Algorithm	50
9 Analysis and Future Work	52

List of Tables

8.1	Summary of Models	45
8.2	Classification Accuracy and Mutual Information for Model Variations, for channels C3, C4, and their combination	45
8.3	Classification Accuracy obtained by comparing a new wave with the most recent examples of that wave for channels C3, C4, and their combination	46
8.4	Classification Accuracy obtained on 2005 Data Set for three subject, using the best channel described in [61], in order to be able to produce a comparison	46
8.5	Classification Accuracy obtained on 2005 Data Set for three subject, using the best combination of 3 channels described in [61], in order to be able to produce a comparison	47
8.6	Classification accuracy and mutual information for the algorithms submitted during the 2003 BCI competition	47
8.7	Classification Accuracy obtained for baseline algorithm on 2005 Data Set for three subject, using the best channel [61]	51
8.8	Classification Accuracy obtained for baseline algorithm on 2005 Data Set for three subject, using the best combination of 3 channels [61] .	51

List of Figures

1.1	32-channel electrode cap made by Neuromedical Supplies [5]	6
4.1	ALN: X = explanatory variables that represent the input variables to the neural network; C = class labels [11]	27
5.1	Electrode positions (left) and timing scheme (right). [1]	31
5.2	Example of EEG signal.	32
5.3	Timing of a trial. [2]	33
6.1	A simple illustration of a signal with three segments. The solid lines show the underlying deterministic components of the regression models within each state, and the points show the actual noisy observations.	35
6.2	Architecture of an HMM. [68]	37

Chapter 1

Introduction

The interaction between humans and computers has been an expanding field of research and development in recent years. The last two decades have witnessed the emergence of innovative human-computer interfaces that use voice, vision, haptics, and a combination of these as communication support. [21]

Recent advances in computer hardware, neuroscience, signal processing, and machine learning have made it possible to use human EEG signals or “brain waves” to communicate with an electronic device, such as a computer. Devices capable of this type of direct communication between a person’s brain and an external device are called brain-computer interfaces (BCIs). Most current BCIs are not invasive. They consist of electrodes applied to the scalp of an individual or worn in an electrode cap such as the one in Figure 1.1.



Figure 1.1: 32-channel electrode cap made by Neuromedical Supplies [5]

These electrodes pick up the brain's electrical activity (at the microvolt level) and carry it into amplifiers, which amplify the signal approximately ten thousand times and then pass it via an analog-to-digital converter to a computer for processing [5]. The computer processes the EEG signal and uses it in order to accomplish tasks such as communication and environmental control. BCIs are slow in comparison with normal human actions, due to the complexity and noisiness of the signals used, as well as the time necessary to complete signal processing and recognition.

Previous work on BCIs has focused on extracting relevant features from the EEG trials through statistical analyses of their time, frequency and phase properties [43, 54, 52, 19]. Features are grouped into feature vectors that are used to build recognition models using machine learning algorithms.

The recognition model used in this thesis is a deformable Markov model for time-series pattern matching described in [22]. This is a model for detecting specific patterns or shapes in time-series data. The patterns of interest are modeled as K -state deformable Markov models where each state is responsible for the generation of a component of the overall shape using a state-based regression function. The distance between segments is modeled as a semi-Markov process, allowing flexible deformation time. Recognition of a pattern in a new time series is achieved by a dynamic programming algorithm.

1.1 Motivation

On December 8, 1995, Jean-Dominique Bauby, the editor-in-chief of French *Elle*, suffered from a severe stroke and became locked-in. His story is described in [5]. Bauby was left with an unimpaired mind and the ability to blink his left eye. He decided to write a book about his condition and did so by dictating it through blinking his eye. Each potential letter choice was placed in Bauby's field of view in the order of its frequency in the French language. Whenever Bauby wanted to choose a letter, he blinked. Bauby used this method to record his book, as well as to communicate with others. In his book, Bauby recounted the horrors of twenty-four hour care. He hated it when the nurse left the television set on because he had no way of changing the channel or turning it off. The nurse did not always pay attention, even when she was around. Bauby talked about being ignored while madly blinking at the nurse to turn the TV off.

Many different disorders can disrupt the neuromuscular channels through which the brain communicates with and controls its external environment. Amyotrophic lateral sclerosis (ALS), brainstem stroke, brain or spinal cord injury, cerebral palsy, muscular dystrophies, multiple sclerosis, and numerous other diseases impair the neural pathways that control muscles or impair the muscles themselves. They affect nearly two million people in the United States alone, and far more around the world. Those most severely affected may lose all voluntary muscle control, including eye movements and respiration, and may be completely locked-in, unable to communicate in any way [69].

With the availability of technology today, the life of patients such as Bauby could be significantly improved by giving them back even a little control over their environments or the ability to communicate. Thus a great deal of research has focused on BCIs as communication systems. The immediate goal of this research is to provide basic communication capabilities to users who may be completely paralyzed, allowing these users to express their wishes to care givers or even control a

word processing application, a computer, or neuroprosthesis. Typical BCI systems for communication include control of the elements in a computer-rendered environment such as cursor positioning [20, 70], spelling programs (e.g. virtual keyboard [50]), and command of an external device (e.g. TV switch, robot [41], or prosthesis [53]).

While the most important goal in creating BCIs is to give some form of control back to the most handicapped individuals, there are other motivations for creating a brain-computer interface, some of which are discussed in Section 2.1.

1.2 BCI Task and Data

This thesis considers BCI paradigms in which the subject needs to perform one of two or more mental activities according to a cue. The goal is to recognize patterns in the EEG signals associated with each mental activity and to use them to perform a task. For example, imagined movement of the left hand might be associated with movement of a mouse cursor to the left, while imagined movement of the right hand might be associated with movement of a mouse cursor to the right.

The data to be used in this thesis have been made available by the Laboratory of Brain-Computer Interfaces at Graz University of Technology through two online competitions: BCI Competitions II and III. Each data set consists of single trials of spontaneous brain activity, one part labeled (the training data) and another part unlabeled (the test data), and a performance measure. The goal is to infer labels (and their probabilities) for the test set from the training data. The labels should maximize the performance measure for the true test labels. There are many possible performance metrics, such as classification accuracy or mutual information. The latter combines classification accuracy with the confidence in classification, providing a better metric when comparing several algorithms.

1.3 Previous Approaches

A great deal of research focuses on creating non-invasive BCI applications that facilitate communication. Subjects control the electronic devices by performing mental tasks which are associated with actions specific to each BCI application. The association between mental tasks and actions requires the selection of a set of mental tasks to which the BCI responds (controlling set), and the identification of patterns in the brain activity that characterize each mental task in the controlling set. These patterns are identified through the analysis of the electrophysiological signals recorded during the performance of the mental tasks in the controlling set.

The first step in analyzing the electrophysiological signals is for the user to execute a training phase. As the user performs the mental tasks in the controlling set, machine learning algorithms analyze the brain signals recorded from sensors on the scalp and learn the patterns specific to each mental task. Once the training phase is over, the BCI has a model of the brain signal patterns associated with each movement in the controlling set. During a testing phase, the brain signals recorded as the user performs the mental tasks in the controlling set are analyzed using the model learned during training in an attempt to identify the mental task being performed. There are two types of adaptation necessary for a successful BCI: the recognition models need to adapt to each individual subject, and the subject

needs to adapt to the recognition model. Indeed, based on feedback from the BCI, the subject can learn to modulate his brain activity to improve the classification accuracy of the model.

Many of the algorithms used for BCIs can be characterized procedurally in the following way:

- 1) find an approximate representation of the EEG (for example, Fourier coefficients);
- 2) define a flexible function that measures the distance between two waves;
- 3) provide an efficient algorithm using this representation and distance function.

While these approaches employ useful heuristics for waveform matching, they do not explicitly account for the uncertainty in the matching process. Consequently, one cannot quantify in general the inherent uncertainty associated with any detection decision [22].

1.4 Proposed Algorithm

The recognition model used in this thesis provides a probabilistic framework that accounts for the uncertainty in the matching process. The model is a deformable Markov model for time-series pattern matching described in [22]. As described above, this is a model for detecting specific patterns or shapes in time-series data. The patterns of interest are modeled as K -state deformable semi-Markov models where each state is responsible for the generation of a component of the overall shape using a state-based regression function. The distance between segments is modeled as a semi-Markov process, allowing flexible deformation time. Recognition of a pattern in a new time series in this model is typically achieved by a dynamic programming algorithm.

1.5 Goals and Evaluation

Unlike some of the previous approaches, this recognition model provides a probabilistic framework that can quantify the inherent uncertainty associated with any detection decision. Additionally, recognition of a pattern in a new time series is achieved by an algorithm that allows flexible deformation time. Scalability is very important because BCIs are very slow in comparison to human actions, and this is why this thesis also presents variations of the model that skip the dynamic programming step during recognition in order to speed up classification.

For this thesis, I implement the deformable semi-Markov model for time-series pattern matching and test this algorithm and several variations of it on data provided at BCI Competitions II and III. I compare the results obtained using this method to those obtained by the researchers at Graz University and by the participants in the two competitions.

1.6 Organization

The remainder of this thesis is organized as follows.

Chapters 2, 3 and 4 are background chapters describing the basic components and current standards in BCI research, augmented with examples from previous work in the field. Chapter 2 introduces brain activity monitoring methods, and continues with a description of the mental tasks used for controlling a BCI and the brain signals analyzed for each task. Chapters 3 and 4 introduce two main components in the analysis of EEG signals in a BCI: extracting relevant features of the brain signal (presented in Chapter 3), and using machine learning algorithms to identify patterns in these features (discussed in Chapter 4).

Chapter 5 provides a detailed description of the BCI task and data used in this thesis.

Chapters 6 gives a formal description of deformable Markov model templates for time series pattern matching, the recognition method used in this thesis. It describes how these models can be constructed from the data and how they can be used for online detection.

Chapter 7 describes the methods used to evaluate BCI performance: accuracy and mutual information. The results obtained by the variations of the algorithm explored in this thesis are presented in Chapter 8. This chapter also compares the results with those obtained by participants in the online BCI competitions.

Chapter 9 outlines the contributions and limitations of the recognition method used in this thesis and of existing BCIs. It provides a description of possible future work and of other approaches that may be used in an attempt to improve the flexibility and speed of BCI systems.

Chapter 2

Background

After a brief description of BCI systems, this chapter introduces brain activity monitoring methods. It then focuses on mental tasks used to control BCIs and the brain signals analyzed for each task.

2.1 BCIs and Their Many Uses

Recent achievements demonstrate that it is currently possible to implement crude brain-computer interfaces - brain dishes - that allow in vitro neuronal clusters to directly control computers. A brain dish is a direct mind-computer interface in the form of a small cluster of neurons in a petri dish that have wired themselves to electrodes. Thomas DeMarse at Florida University created a brain dish that was able to learn to fly the flight simulator of an F-22 [14].

Cyberkinetics, in conjunction with the Department of Neuroscience at Brown University, built a brain-computer interface called BrainGate, used to restore functionality lost due to paralysis. After having about 100 electrodes implanted on the surface of his brain, Matt Nagel became the first quadriplegic to use a BCI. He can now control a computer mouse cursor, using it to press buttons that can control a television, and check e-mail. He can even draw (although the cursor control is not yet very precise). He can also send *open* and *close* commands to an external prosthetic hand [13].

Dobelle created a bionic eye that uses a digital video camera mounted on glasses to capture an image and send it to a small computer on a patient's belt. The image is processed and sent to electrodes implanted in the patient's visual cortex. The electrodes stimulate the brain, producing a pattern of bright spots that form an image. Similar devices for restoring hearing have also been designed [15].

Scientists at NASA have developed a way for humans to control a computer map by translating brain waves into computer commands. A person can guide the direction of a moving map by simply attending to the direction in which he/she wants to go [45].

BCIs have also been used for games, such as fMRI Pong [46] and balancing a character on a tightrope [35].

2.2 Brain Activity Monitoring Methods

Brain activity produces a variety of phenomena that can be measured with adequate sensors and have potential use in BCIs [21]. Current monitoring methods include electrical potential measurements (electroencephalogram or invasive electrophysiological methods), functional magnetic resonance imaging (fMRI), magnetoencephalogram (MEG), and positron-emission tomography (PET). Among the current monitoring methods, scalp recorded electroencephalogram (EEG) constitutes an attractive choice for BCI systems because of its non-invasiveness, relative simplicity and low cost. Therefore, most BCI researchers focus their attention on EEG-based BCIs [21].

2.3 Mental Tasks

Typical mental tasks include: evoked responses to external stimuli, motor imagery, and spatial, geometrical, arithmetical, and verbal operations. Following the type of mental tasks they employ, BCIs are categorized into evoked response and operant conditioning based ones. Evoked-response BCIs rely on the subject's attention focusing on particular stimuli that are associated with actions. Operant-conditioning BCIs react to particular controlling mental tasks executed by the subject.

2.3.1 Evoked-Response BCIs

External visual or auditory events (e.g. flashing objects on a computer screen or brief sounds) elicit EEG voltage deviations that are known as event related potentials (ERPs). When a subject pays attention to a particular stimulus, an ERP appears in his EEG.

P300

P300 is a type of ERP characterized by a positive deflection of the EEG voltage that can usually be detected approximately 300 ms after the presentation of an unexpected stimulus [16, 62, 66].

In P300-based BCIs, the subject is presented with a sequence of events that can be classified into two categories. The subject is assigned a task that involves categorizing the events. In general, events in one of the two categories are rarely presented. In these experiments, events in the rare category generate a prominent P300. The BCIs can use this effect to determine the subject's intent.

P300 has been successfully used in the design of a BCI keyboard [38]. The subject faces a 6 x 6 matrix of symbols. Every 125 ms, a single row or column flashes. The events are categorized into flashings that contain the relevant cell and flashings that do not contain the relevant cell. The P300 is prominent when the row or column of the desired symbol flashes. Thus, the intended symbol is at the intersection of the row and column that elicited a P300 [16].

The apparent advantage of a P300-based BCI is that it requires no initial training.

Steady State Visually Evoked Response

Flicker stimuli of various frequencies elicit a steady state visually evoked response

(SSVER) in the EEG which is characterized by an oscillation at double the frequency of the stimulus [10]. An SSVER can be detected by examining the spectral content of the signals in the visual cortex.

When actions are associated with target flickers of different frequencies the subject can control a BCI by gazing at the target corresponding to the desired action [45, 10, 36, 40, 44, 18, 32].

Scientists at NASA have developed a way for humans to control a computer map by using SSVER. The four directions of map rotations are marked by small flickering checkerboards on the display, each one flickering at a slightly different frequency. As the user views the moving map and attends to the desired flickering bar, the computer analyzes the brain waves recorded from sensors on the scalp and picks out the specific frequencies and sensors correlated with the flicker. During a training phase, separate peak frequencies are detected in the EEG of each user, for each direction of map rotation [45].

SSVERs generated in response to phase-reversing checkerboard patterns have been used efficiently for binary control in a visually elaborate immersive 3D game. The object of the game was to gain 1D control of the balance of an animated character on a tightrope, by attending to checkerboards positioned on either side of the character [36].

SSVERs have also been used to control the roll position of a simple flight simulator. A display in the simulator presents a series of commands requiring the operator to roll right or left. The SSVER is elicited using a visual stimulus generated by fluorescent tubes that are luminance-modulated at certain frequencies and mounted behind a translucent diffusing panel [40].

2.3.2 Operant Conditioning Based BCIs

Generation of EEG activity used to control BCIs is often achieved by the subject performing certain cognitive tasks. These tasks require voluntary control over a type of brain activity that the person is usually unaware of and unable to sense. Generation of the EEG activity used to drive a BCI is usually achieved, at least initially, by the subject performing certain cognitive tasks. In time, subjects may be able to automatize the skill of controlling EEG components, thus increasing their control over the BCI in times of cognitive fatigue.

Functional brain imaging studies monitoring changes in regional cerebral blood flow revealed similar patterns of activity during motor imagery and actual movement performance [52], making motor imagery a good mental task for the control of BCIs [12].

BCI research groups have used motor imagery tasks that include: left hand movement, right hand movement, foot movement, finger movement, and tongue movement [67, 52, 48, 54, 59, 63, 27, 30, 28, 4].

Some BCI research groups have instructed subjects to think of anything they want, as long as they can achieve control of the BCI. The idea is that with the aid of feedback the subject's brain will learn to control EEG components in an appropriate, automated way [12].

The Graz BCI system [44, 52, 48, 54, 59, 63, 27, 24, 43, 69, 58, 56, 49, 55, 50, 53] is a motor imagery-based BCI that analyzes parameters derived by autoregressive frequency analysis. This research has focused on distinguishing between the EEG associated with imagination of different simple motor actions, such as right or left hand or foot movement. Each of the imagined movements is associated with an

action, thus enabling the user to control a cursor or an orthotic device that opens and closes a prosthetic hand. For example, imagined left hand movement could be used to signal the user's intent to open the prosthetic arm, and imagined right hand movement could be used to signal the user's intent to close the arm. The mapping between the imagined movements and the actions is not important, as long as the EEG of the imagined movements can be discriminated.

In the standard protocol, the user participates in an initial session to select a motor imagery paradigm. In each series, the user imagines one of several specified movements. For each imagined movement, a feature vector is defined. The vectors establish a user-specific linear or non-linear classifier that determines from the EEG which action the user is imagining. In subsequent sessions the system uses the classifier to translate the user's motor imagery into a continuous output (e.g., extension of a lighted bar or cursor movement) or a discrete output (e.g., selection of a letter or other symbol). Normally, the classification algorithm is adjusted between daily sessions, by incorporating the most recent data. Over 6-7 sessions with two-choice trials, the users can reach accuracies of over 90%.

The Berlin BCI [8] focuses on distinguishing between EEG associated with different motor imagery tasks. The EEG is recorded from 128 different channels. Common Spatial Pattern analysis is used to yield a set of features that are then used by Linear Discriminant Analysis to classify different types of motor imagery tasks. The results from a feedback study with six healthy subjects demonstrate that the system provides reliable feedback after a short calibration measurement and machine training without the need for the subject to adapt to the system.

Slow Cortical Potential Shifts

Slow Cortical Potential Shifts (SCPs) are shifts in the depolarization level of the upper cortical dendrites [7]. They indicate the overall preparatory excitation level of a cortical network and they are universally present in the human brain [21]. Negative SCPs are usually associated with movement and other functions involving cortical activation, while positive SCPs are typically associated with reduced cortical activation [7]. SCPs can be obtained using low pass filtering or wavelet decomposition [26].

Subjects can learn through operant conditioning to produce a SCP shifts in an electrically positive or negative direction for binary control [7]. SCPs were used to implement a spelling program through which locked-in patients could communicate at a rate of one word per minute [7].

The Tübingen BCI [7, 26], called the Thought Translation Device, trains locked-in patients to self-regulate slow cortical potentials of their EEG. After operant learning of SCP self-control, patients select letters from a spelling device. The alphabet is first split into letter-banks which are presented successively at the bottom of the screen for several seconds. If the subject selects the letter bank being shown by generating an SCP shift, the letter bank is then split into two, and so on, until one letter remains.

A training day usually consists of 6-12 sessions each of which lasts about 5-10 minutes. Patients are trained several times per week. Patients had to be trained for several months to achieve 75% accuracy.

Oscillatory Sensorimotor Activity

Populations of neurons can form complex networks whereby feedback loops are

responsible for the generation of oscillatory activity. The usual classification of EEG rhythms based on their frequency ranges is as follows: delta (2 to 4 Hz), theta (4 to 8 Hz), alpha (8 to 13 Hz), beta (13 to 30 Hz), and gamma (higher than 30 Hz).

The alpha rhythm, discovered by Hans Berger in 1929, is typical of a resting condition and disappears when the subject perceives a sensory signal or when he makes mental efforts [21].

The theta rhythm originates from interactions between cortical and hippocampal neuronal groups. It appears in periods of emotional stress and during rapid-eye-movement sleep [21].

The delta rhythm appears during deep sleep and anesthesia, and is also present during various meditative states involving willful and conscious focus of attention in the absence of other sensory stimuli [21].

The neural oscillators that generate the beta rhythms are located inside the cortex. The beta rhythm is typical of periods of intense activity of the nervous system and occurs principally in the parietal and frontal regions [21].

The basis of gamma oscillations is interneuronal feedback with quarter-cycle phase lags between neurons situated close to each other in local areas of the cortex. It is thought that gamma oscillations are associated with attention, perception, and cognition [21].

Two types of oscillations are particularly important for BCIs: the alpha and the central beta rhythms, both originating in the sensorimotor cortex [52]. Sensorimotor stimulation, motor behavior, and mental imagery can change the functional connectivity within the cortex and result in an amplitude suppression [event-related desynchronization (ERD)] or in amplitude enhancements [event-related synchronization (ERS)] of alpha and central beta rhythms [52]. Preparation and planning of self-paced hand movement results in a short-lasting ERD of the alpha and central beta rhythms [52, 67, 48]. The similarity in the rhythm changes produced by preparation and actual movement suggests the feasibility of BCI systems.

An example of a BCI that allows people with or without motor disabilities to learn to control mu and beta rhythm amplitude and use the control to move a cursor in one or two dimensions to targets on a computer screen is the Wadsworth BCI [70, 39, 69]

For each dimension of cursor control, a linear equation translates mu or beta rhythm amplitude from one or several scalp locations into cursor movement. Users learn to control the mouse cursor over a series of 40-minute sessions. They participate in 2-3 sessions per week and most acquire significant control within 2-3 weeks. In the initial sessions most employ motor imagery to control the cursor. As training proceeds, control becomes automatic.

Research with this BCI has focused on defining the topographical, spectral, and temporal features of mu and beta rhythm control and on optimizing the mutually adaptive interactions between the user and the BCI system.

2.4 Acquisition of EEG Signals

The data used by a BCI is acquired during a BCI experiment. In a standard BCI experiment, the subject is generally seated in a comfortable chair in a dimly illuminated room. Electrodes are placed on his head, usually using a cap with the electrodes fixed to it. The cap ensure that the inter-electrode distance is appropriate and that the electrodes are placed in roughly the same position in all experiments.

To improve the conduction between the skin and the electrode surfaces, electrode gel or salt solutions are used. Reference electrodes which are placed on a presumed inactive zone are then chosen. Alternatively, the EEG may be recorded with any scalp electrode as a reference, and then the average reference can be computed as a mean of all electrodes [21]. The EEG signals, usually confined to the 0-40 Hz band, are then recorded, as the subject performs a series of tasks.

2.5 Preprocessing

During preprocessing, the external noise (e.g. power line noise) is removed from EEG trials and the presence of artifacts is detected. Electromagnetic and EEG equipment noise are narrow band pass signals. Thus, filtering them out is straightforward. Typically, EEG signals are filtered in the 0-40 Hz frequency band [48, 54, 59, 39, 42, 53]. This filtering removes most of the noise, as power line and other electromagnetic noise sources have frequency supports beyond 40 Hz [21].

In general, EEG trials containing artifacts are discarded, because the relevant information contained in the trials is masked by the artifacts [70, 24]. For instance, at frontal, temporal or occipital locations, ocular artifacts can exceed EEG in amplitude [25]. In some studies, EEG artifacts are detected visually, by observing the subject as he is performing the experiment [38]. However, artifacts can be automatically detected using an outlier-detection procedure [21].

2.6 Operating Protocols

Each BCI has a protocol that guides its operation. This protocol defines how the system is turned on and off, whether communication is continuous or discontinuous, whether the message transmission is triggered by the system or by the user, the sequence and speed of interaction between user and system, and the feedback provided to the user [69].

Ideally, a BCI would be available at all times (asynchronous BCI). Continuous translation can, however, produce unintended or random output. One research group successfully implemented a continuous translation procedure and was able to obtain up to 94% classification accuracy in detecting imagined motor movement [47]. Another possibility is to have a BCI-based on/off key, such as a distinct pattern of signal features that is extremely unlikely to occur spontaneously (e.g., a specific pattern of SCP shifts) [7].

In contrast, synchronous BCIs usually control timing and rate and indicate them to the user by visual or auditory means. For example, the experimenter decides when a task begins and ends and signals that to the user. The EEG signal of the user is only recorded and analyzed during this time, when it is known that the user is trying to execute the required task. To some extent, the system control of timing and rate results from the requirements of research: to assess BCI performance, it is necessary to know user intent, and the simplest way to do this is to have the system tell the user what to communicate and when.

Chapter 3

Feature Extraction Procedures

The performance of a BCI, like that of other communication systems, depends on its signal-to-noise ratio. The goal is to recognize and execute the user's intent, and the signals are those aspects of the recorded electrophysiological activity that correlate with and thereby reveal that intent. The BCI system's first task is to measure the signal features accurately, so as to maximize the signal-to-noise ratio. The goal of feature extraction is to map the EEG trial space into a feature space suitable for discrimination of EEG trials resulting from the performance of different mental tasks. This chapter described common mappings that result from space [43], frequency (parametric [54] and nonparametric [52]) and time-frequency analysis [19].

3.1 Spatial Filters

In many experiments, the EEG covers large parts of the brain and is recorded on a multitude of channels. Given that sensorimotor rhythms originate from very localized areas in the cortex, it is expected that not all signals recorded from different sites contribute the same amount of information to the classification, and some may only contribute noise. The signals from different electrodes thus have to be weighted in some way, in order to reflect their relevance for the task. Additionally, the correlation between signals from neighboring electrodes can be used to suppress the noise in individual channels [43].

The simplest spatial filter is the bipolar derivation, which derives the first spatial derivative and thereby enhances differences in the voltage gradient in one direction [69].

The Laplacian derivation is the second derivative of the instantaneous spatial voltage distribution, and thereby emphasizes activity in radial sources immediately below the recording location. It can be computed by combining the voltage at the location with the voltages of surrounding electrodes. As the distance to the surrounding electrodes decreases, the Laplacian becomes more sensitive to voltage sources with higher spatial frequencies (more localized sources) and less sensitive to those with lower spatial frequencies (more broadly distributed sources) [69].

Laplacian and common average reference spatial filters apply a fixed set of

weights to a linear combination of channels. Both use weights that sum to zero so that the result is a difference. Principal component, independent components, and common spatial patterns analyses are alternative methods for deriving weights for a linear combination of channels. In these methods, the weights are determined from the data.

In [43], the raw signals are decomposed into spatial patterns that are extracted from the data of two populations of EEGs in a manner that maximizes their difference. The spatial patterns provide a weighing of the electrodes, which is derived directly from the data. The high-dimensional, spatial-temporal raw signals are projected onto a few specifically designed spatial filters. These filters are designed in such a way that the variances of the resulting signals carry the most discriminative information. The adjuncts of the filters are called Common Spatial Patterns, and they are obtained from a set of calibration data by a method called CSP. Given two distributions in a high-dimensional space, the CSP algorithm finds directions that maximize variance for one class and at the same time minimize the variance for the other class [8].

By using this method in different frequency bands, [20] derive a set of space frequency filters to characterize each mental task in the controlling set. Furthermore, [19] combine the diagonalization of the mean autocorrelation matrices with the analysis of time-frequency correlations to derive a set of features to discriminate among three mental activities, namely imagined left and right index finger movements and mental counting.

3.2 Time-Frequency Analysis

EEG signals are composed of the univariate signals recorded at each electrode. They can thus be modeled as realizations of a multivariate stochastic process. Time-frequency (TF) analysis of multivariate signals aims at describing the time variations of their intra- and inter-component spectral properties by means of a time-frequency representation (TFR). TF analysis is particularly useful for non-stationary signals for which analysis restricted to time or frequency is not sufficient to describe their dynamics [21].

Time-frequency representations of signal can be divided into two groups according to the nature of their transformations: linear (Short-time Fourier Transform), and quadratic (based on the Wigner-Ville distribution) [19]. The latter may be further subdivided as power or correlation based.

Following is a description of time, frequency, and time-frequency analysis of univariate signals. The analysis is generalized to multivariate signals in [21].

Let y be a univariate stochastic signal of length T , composed of random variables $\{y(t)|t = 0, \dots, T - 1\}$, where t is the time index. A description of time-frequency analysis methods is presented below [21]:

3.2.1 Time Domain Analysis

The properties of y can be described in time using first and second order moments computed on the random variables $y(t)$. These moments are:

1. *The expectation of $y(t)$: $E_{p(y(t))}[y(t)]$, where $p(y(t))$ is the probability density function associated with $y(t)$.*

2. *The expectation of the product $y(t_1)y(t_2)$: $E_{p(y(t_1),y(t_2))}[y(t_1)y(t_2)]$, where $p(y(t_1),y(t_2))$ is the joint probability density function of $y(t_1)$ and $y(t_2)$.*

Expectations taken with respect to the probability density functions associated with the random variables $y(t)$ are called ensemble averages. For convenience of notation, any ensemble average over y is denoted as $E_y[\cdot]$.

The signal power P_y and time autocorrelation function $R_y(t, \tau)$ are defined as:

$$P_y = \frac{1}{T} E_y \left[\sum_{t=0}^{T-1} |y(t)|^2 \right]$$

$$R_y(t, \tau) = E_y [y^*(t - \tau)y(t)]$$

where $*$ stands for the complex conjugate operator, t is the time at which R_y is computed, and $\tau \in \{-T + 1, \dots, T - 1\}$ is the time lag. Since P_y can be written as an average over time of $E_y[|y(t)|^2] = R_y(t, 0)$, it follows that $E_y[|y(t)|^2]$ can be considered as the signal power density in the time domain (or power time density, PTD). Thus, $E_y[|t(y)|^2]$ can be used to compute the average, over the PTD of any time function $\gamma(t)$ as follows:

$$(\gamma(t))_{PTD} = \frac{1}{T} \sum_{t=0}^{T-1} \gamma(t) E_y[|y(n)|^2]$$

3.2.2 Frequency Domain Analysis

Most current BCIs use features based on the parametric and nonparametric spectral representations of EEG signals.

Nonparametric spectral representations are obtained through the discrete Fourier transform.

The frequency properties of y can be examined using its discrete Fourier transform defined as:

$$\hat{y}(\psi) = \sum_{t=0}^{T-1} y(t) \exp(-j \frac{2\pi t \psi}{T})$$

where ψ is the frequency index. The correspondence between the frequency index ψ and the actual frequency f (in Hz) is given by [21]:

$$f = \frac{f_y \psi}{N} \quad \psi = 0, \dots, \frac{T}{2}$$

where f_y is the sampling frequency.

Using the discrete inverse Fourier transform, y can be obtained from \hat{y} as follows [21]:

$$y(t) = \frac{1}{T} \sum_{\psi=0}^{T-1} \hat{y}(\psi) \exp(j \frac{2\pi t \psi}{T})$$

Similarly to the time domain, first and second order moments can be defined on the random variables $\hat{y}(\psi)$. In particular, the frequency autocorrelation function can be defined as:

$$R_y(\psi, \nu) = \frac{1}{T} E_y[\hat{y}^*(\psi - \nu)\hat{y}(\psi)]$$

where ν is the frequency lag and $\frac{1}{T}$ is the normalization factor.

The PTD of y can be obtained by taking the Fourier transform with respect to the time lag τ by the sum over t of the time autocorrelation functions $R_y(t, \tau)$:

$$(exp(j\frac{2\pi t\nu}{T}))_{PTD} = \frac{1}{T} \sum_{t=0}^{T-1} E_y[|y(t)|^2] exp(j\frac{2\pi t\nu}{T})$$

Several studies successfully used frequency analysis for feature extraction. In [52], the power in the alpha and beta bands at electrodes located near the motor cortex are used to discriminate between EEG trials produced from the imagination of left and right hand movements. In [69], the powers in the alpha band at frontal, central and occipital electrodes are used in a 1D cursor positioning application. In [42, 41], the powers in 2 Hz wide frequency bands from 8 to 30 Hz at every electrode are used as features to discriminate among five mental tasks, namely relaxing, imagination of left and right hand movements, rotation of a cube, and arithmetic.

3.2.3 Combining Time and Frequency Analysis

The analyses in the time and frequency domains can be combined into an analysis in the time-frequency plane.

The fundamental power-based time-frequency representation (TFR) of a signal is its Weigner-Ville transform (WVT):

$$W_y(t, \psi) = \frac{1}{T} \sum_{t=0}^{T-1} y^*(t - \tau)y(t) exp(-j\frac{2\pi\tau\psi}{T})$$

The time-frequency autocorrelation can be defined as:

$$R_y(t, \tau, \psi, \nu) = \frac{1}{T} E_y[W_y^*(t - \tau, \psi - \nu)W_y(t, \psi)]$$

where t and ψ are the time and frequency at which the TF correlation is computed, and τ and ν are the time and frequency lags respectively.

Nonparametric spectral representations are often too slow for online BCI experiments. Parametric spectral representations have proved to be more practical for online BCI. The latter include: autoregressive (AR) [51] and autoregressive moving - average (ARMA) [31] models for each EEG channel, multivariate autoregressive models (MAR) that characterize all the channels simultaneously [3], and adaptive autoregressive models that do not assume the EEG is stationary [54, 58].

3.3 AR Model

The notation $AR(p)$ refers to an autoregressive model of order p . The $AR(p)$ model can be written as [9]:

$$y_t = \sum_{i=1}^p a_i y_{t-i} + \epsilon_t$$

where a_1, \dots, a_p are the parameters of the model and ϵ_t is an error term.

The AR parameters can be calculated using the Yule-Walker equations [9]:

$$\gamma_m = \sum_{k=1}^p a_k \gamma_{m-k} + \sigma_\epsilon^2 \delta_m$$

where $m = 0 \dots p$, γ_m is the autocorrelation function of y , σ_ϵ is the standard deviation of the input noise process, and δ_m is the Kronecker delta function.

For $m = 0$ the equation:

$$\gamma_0 = \sum_{k=1}^p a_k \gamma_{-k} + \sigma_\epsilon^2$$

can be used to solve for σ_ϵ^2 . The remaining p equations are used to solve for the AR parameters $a_1 \dots a_p$, which are the p unknowns in the system of equations.

3.3.1 ARMA Model

The notation $MA(q)$ refers to the moving average model of order q [9]:

$$y_t = \epsilon_t + \sum_{i=1}^q b_i \epsilon_{t-i}$$

where b_1, \dots, b_q are the parameters of the model and $\epsilon_1, \dots, \epsilon_t$ are the error terms.

The notation $ARMA(p, q)$ refers to the model with p autoregressive terms and q moving average terms. This model contains the $AR(p)$ and $MA(q)$ models [9]:

$$y_t = \epsilon_t + \sum_{i=0}^p a_i y_{t-i} + \sum_{i=0}^q b_i \epsilon_{t-i}$$

ARMA models can, in general, after choosing p and q , be fitted by least squares regression to find the values of the parameters that minimize the error term.

3.3.2 MAR Model

In a multivariate model with c variables, an observation y_t is a c -dimensional vector and all of the equations above still hold [3].

3.3.3 AAR

An adaptive autoregressive (AAR) model describes a signal y_t in the following form [54]:

$$y_t = \sum_{i=1}^p a_{i,t} y_{t-i} + E_t$$

where, in the ideal case, E_t is a purely random or white noise process with mean zero and variance σ_E^2 . The difference to an AR model is that the parameters $a_{1,t} \dots a_{p,t}$ can vary with time [54].

For simplicity, the AR parameters and the past p samples of the time series are defined as vectors [58]:

$$a_t = [a_{1,t} \dots a_{p,t}]'$$

$$y_t = [y_t \dots y_{t-p+1}]'$$

The Mean Square Error (MSE) is thus defined as [58]:

$$MSE = \frac{1}{T} \sum_{t=1}^T E_t^2$$

and the total power of the EEG

$$MSY = \frac{1}{T} \sum_{t=1}^T y_t^2$$

The relative error variance, which is a measure for the goodness-of-fit of the model, is defined below:

$$REV = MSE/MSY$$

The Least Mean Square Estimation method can be characterized by the following formulas [54, 58]:

$$E_t = y_t - a_{1,t-1}y_{t-1} - \dots - a_{p,t-1}y_{t-p}$$

$$a_{i,t} = a_{i,t-1} + cE_t y_{t-i}, \quad i = 1 \dots p$$

with E_t being the prediction error and c an update ratio.

$$c = UC/MSY$$

where UC is called the update coefficient. This has the advantage that the adaptation ratio can be defined independently of the signal power, which is equivalent to normalizing the signal to power of unity [58].

An important issue is the initial value of a_0 . There are two possibilities that were examined in [58]: initialization with zero and with average values estimated by the Yule-Walker equation.

The parameters of the model can also be estimated using the Recursive Least Squares Algorithm (RLS), which is characterized by the following equations [58]:

$$E_t = y_t - a_{t-1}'y_{t-1}$$

$$r_t = \lambda^{-1}A_{t-1}y_{t-1}$$

$$k_t = \frac{r_t}{y_{t-1}'r_t + 1}$$

$$a_t = a_{t-1} + k_t E_t$$

$$A_t = \lambda^{-1}A_{t-1} - k_t r_t'$$

k_t is called the Kalaman Gain Vector. The term λ^{-1} serves a similar purpose to that of the UC update coefficient, to which it is connected by the formula:

$$\lambda^{-1} = \frac{1}{1 - UC}$$

The AAR parameters are estimated for every sample time point for all EEG channels. They are then used to form a matrix of data vectors $d_t(k)$, where k denotes the trial number, t denotes the sample number within the trial, and c_j denotes an EEG channel. [49]:

$$d_t(k) = [a_t(k), c_j]$$

3.4 Nonlinear Feature Extraction Methods

Nonlinear (or deterministic chaos) feature extraction methods have been sporadically used in the BCI context. In [65], phase space reconstruction of the chaotic attractor associated with signals recorded at electrodes $O1$ and $O2$ are used to discriminate between three cognitive mental states: eyes open-subject alert, eyes closed-subject alert, and eyes closed-subject performing visualization tasks. Phase synchronization measures (ERS and ERD) [6] appear to be adequate for the recognition of certain mental states from EEG. Furthermore in [64] measures of the EEG complexity are used to provide 1D control.

Chapter 4

Classification Algorithms

BCI translation algorithms convert signal features into device control commands. Commands may be continuous (e.g., cursor movements) or discrete (e.g., letter selection) [69]. The success of a BCI is determined by the appropriateness of the selected signal features, by how well it encourages and facilitates the user's control of these features, and by how effectively it translates this control into device commands.

This chapter discusses use a variety of translation algorithms, including linear equations, discriminant analysis, and neural networks [70, 56, 34], which are used in existing BCIs.

4.1 Linear Functions

In the simplest case, in which only one signal feature is used, the output of the translation algorithm can be a linear function of the feature value. The algorithm needs to use appropriate values for the intercept and slope of this function. If the command is vertical cursor movement, the intercept should ensure that upward and downward movement are equally possible for the user. It is believed that the mean value of the signal feature over some interval immediately preceding performance provides a good estimate of the proper intercept [69]. The slope determines the scale of the command (e.g., the speed of the cursor movement). When a single signal feature is used to select among more than two choices, the slope also affects the relative accessibility of these choices [69].

4.2 Linear Discriminant Analysis (LDA)

LDA is a machine learning technique used to find the linear combination of features which best separates two or more classes of objects or events. In the context of BCIs, it is most frequently applied to experiments with continuous feedback based on AAR parameter estimation [58, 54, 24, 49] or CSPs [8, 56].

Consider a BCI application for which the translating algorithm needs to discriminate between two commands (e.g., left versus right imagined hand movement). Consider a set of features, x for each EEG sample, associated with a known class c . This set of samples is called the training set. The classification problem is to find a good predictor for the class c of any sample of the same distribution (not

necessarily from the training set) given only an observation x . LDA performs the classification with the aid of a discriminant function g , where

$$g(x) = w'x + w_0$$

If $g(x) > 0$ then x is classified as belonging to one class, otherwise then x is classified as belonging to the other class. $g(x)$ can be viewed geometrically as a hyperplane separating the two classes.

Assuming that the probability density functions $p(x|c = 1)$ and $p(x|c = 0)$ are both normally distributed, with identical full-rank covariances $\Sigma_{c=0} = \Sigma_{c=1} = \Sigma$, the values of w and w_0 that maximize the likelihood of the data are [17]:

$$w = \Sigma^{-1}(\mu_1 - \mu_0)$$

$$w_0 = -\frac{1}{2}(\mu_1 - \mu_0)' \Sigma^{-1}(\mu_1 - \mu_0) + \ln\left(\frac{P(c = 0)}{P(c = 1)}\right)$$

where μ_0 and μ_1 represent the means of the features in classes 0 and 1 respectively.

The two-class example is generalized for multiple classes and unequal covariances in [17].

In [56] the AAR parameters of the EEG channels or the variance time series of the CSPs are linearly combined and a time-varying distance (TSD) function is calculated using LDA [58, 54, 24, 49]. For example, in [49], the AAR parameters in matrix d_t (as defined in the previous chapter) are combined using weight vector w with which a distance D_t is computed:

$$D_t(k) = w'd_t(k) - w_0$$

The result and the certainty of classification can be presented to the subject by continuously moving a feedback bar representing the D_t . In [56], the lowest online errors obtained using LDA for three subjects were 1.8, 6.8, and 12.5% for the CSP method, and around 5, 9, and 9 % for the AAR method.

4.3 Learning Vector Quantization (LVQ) and Dynamic Scaled Learning Vector Quantization (DSLQVQ)

LVQ and DSLQVQ are predominantly used in online experiments with delayed feedback presentation.

LVQ is a type of artificial neural network. It is a nearest-neighbor classifier which generates a code book of labeled reference vectors in such a way that, for each training example, the nearest vector is, with high probability, of the same category as the training example. In this way, LVQ tries to optimize the representation of the training examples by the set of reference vectors, and at the same time to minimize the classification error. The LVQ code book is generated using an iterative learning procedure: in each iteration a randomly selected training example of known classification is compared to the code book and the two closest reference vectors are found. Depending on the class labels of the given example and of the two reference vectors, these vectors are updated by either pushing them towards the example if the vectors' class labels are identical to the known classification of the example or by pushing them away from the example in the case of incorrect classification.

The overall aim of this learning algorithm is to increase the probability of correct classification the next time the same example is presented [55].

The DSLVQ algorithm is an improved learning vector quantization classifier which uses a weighted distance function and adjusts the influence of different input features through a supervised learning algorithm. The influence of a single feature is modified according to its contribution to correct/wrong classifications of the system. To store and control the influence of each feature, DSLVQ uses a scaling or weight vector w , where each value w_i reflects the importance of the corresponding feature i of the data vectors [55].

In [55], DSLVQ was used on spectral components of the recorded data. The DSLVQ weight values were analyzed to find the most relevant frequency components for each subject, suitable for discrimination between right and left motor imagery.

4.4 Adaptive Logic Network

An ALN (adaptive logic network) is a kind of artificial neural network. It is a special case of the familiar multilayer perceptron (MLP) feedforward network and has been designed to perform basic pattern classification computations at extremely high speed. The *adaptive* term in an ALN refers to the changeable, or adaptive, parameters in an arbitrary number of linear functions. These are functions of the form [11]:

$$L_j = \sum_{i=1}^n w_{ij} X_j - C$$

where X is a vector of independent explanatory variables that represent the input variables to a neural network ($X_0 = 1$ is a bias term), while C is a vector of dependent variables that represent the network output (class labels). If $L_j = 0$, we have the definition of a line for $n = 1$, a plane for $n = 2$, or a hyperplane for $n > 2$.

An ALN changes the weights w_{ij} in each of these linear functions to produce a desired result.

The ALN incorporates the linear functions into the leaf nodes of a decision tree. Each leaf of the tree evaluates whether $L_j \geq 0$, resulting in a value of *true* or *false*. An evaluation of *true* means that a linear threshold has not been reached for a particular combination of values for the input and output variables. The parent nodes of the linear threshold units and all other parent nodes in the tree are restricted to be either *and* or *or* logic functions. An example of an ALN is presented in a figure from [11]:

We can see that an ALN divides the space of solutions into convex regions delimited by the linear functions. A special example of an ALN is the multiple regression model, which is obtained when the ALN consists of only one single linear threshold unit [11].

The number of possible ALN structures is boundless, but fortunately in practice only a reasonable number need to be considered. If there are n independent variables X , each linear piece will have $n + 1$ variable weights - we should not have more weights to be estimated than there are observations in our estimation sample. In addition, many ALN structures that appear to be different are really logically equivalent, so we don't need to consider all of them. For example, $AND(L_1, AND(L_2, L_3))$ is equivalent to $AND(L_1, L_2, L_3)$.

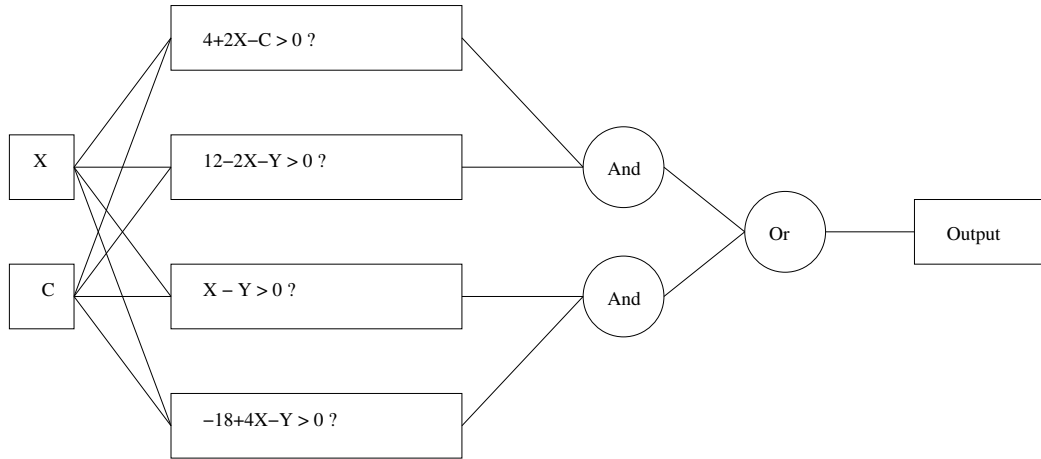


Figure 4.1: ALN: X = explanatory variables that represent the input variables to the neural network; C = class labels [11]

Given a decision tree, an ALN uses least squares in a sequential fashion to estimate its parameters. First, parameter values are randomly assigned to all linear functions in the ALN. Next, training values in the estimation sample are input into the ALN. Truth values are then propagated through the tree, producing the output. For a training example (X_j, C_j) , the active surface is defined as $C = \sum_{i=1}^n w_{ij} X_j$, $C \neq C_j$. Iterative least squares is used to modify the active surface C , moving it closer to the training value [11].

The active surface responsible for the output is determined using logic rules. Every logic node in an ALN can be assigned formal responsibility using the rules of logic. The assignment of responsibility proceeds from the output backwards. By definition, the output node is always responsible. An *and* or *or* node is responsible only if a change in its truth value would change the output of the network. This algorithm is quite efficient, as it eliminates a large part of the network from potential responsibility [11].

For example, if presented with the training example $(x = 3, c = 2)$ for the network in the figure, the linear functions would evaluate to *true*, *true*, *true*, *false*. The logical expression evaluated will be $OR(AND(true, true), AND(true, false)) = true$. If the example were in fact *false*, blame would be assigned to the *AND* on top, which would generate a change of weights for the two linear functions on top.

[34] used ALNs to classify EEG patterns in an on-line experiment. Autoregressive features of the EEG were extracted and presented to the ALN for training. The resulting ALN was then used to determine the direction of cursor movement and update the cursor position on the screen. Subjects were able to control the cursor in one dimension with 100% accuracy, while in two dimensions accuracies were 63% and 76%.

4.5 SVM

Support vector machines (SVM) have been successfully applied for classification and regression in various domains of pattern recognition. In [29], SVMs are used to detect the presence or absence of the P300 component in EEG event related potentials, which is crucial for building a P300-based speller paradigm.

An easy way to perform binary classification is to construct a hyperplane described by a weight vector w and a bias term w_0 . Based on a training set of n examples, with data vectors y_i corresponding to labels c_i , a machine learning algorithm needs to find such a hyperplane according to some suitable optimality criterion. In a test phase, the class label of a new data vector y can be predicted by projecting y on the weight vector w [29]:

$$f(y) = w'y + w_0$$

The sign of the projection would reveal the predicted class label. As a suitable optimality criterion, one may choose the maximum margin criterion, which favors the hyperplane with the largest separation margin, λ . To describe the optimal hyperplane, only the vectors on the margin, the so-called support vectors, are necessary [29]. The weight vector w that maximizes the margin is $w = \sum_i c_i \alpha_i y_i$, where α_i are positive Lagrangian multipliers, resulting in [29]:

$$f(y) = \sum_i c_i \alpha_i (yy_i) + b$$

The replacement of the dot product yy_i by a positive definite symmetric Kernel function $K(y, y_i)$ results in a nonlinear discriminant function [29].

Using this method, [29] obtained an accuracy of 84.5% for separation of P300 from non-P300 epochs for the speller paradigm described in the P300 chapter.

4.6 Evaluation of Classification Algorithms

The evaluation of a translation algorithm reduces to determining how well it accomplishes three levels of adaptation [69]:

1. Initial adaptation to a new user's signal features
2. Periodic online adjustments to reduce the impact of spontaneous variation in a user's signal features
3. Continuing adaptation that encourages and guides the user's adaptation to the BCI

So far, most evaluations have concentrated on the first level of adaptation. In these evaluations, alternative algorithms are applied offline to data gathered from one or more users. Part of the data is used to estimate the parameters of the algorithm, and the rest is used for testing [69].

The second level of adaptation can be addressed by offline analysis that mimics the online situation. The need for this second level of adaptation tends to favor simpler algorithms. Parameter adaptation is likely to be more difficult and more vulnerable to instability and require more data for more complex algorithms [69].

The third level of adaptation is not amenable to offline evaluation. The goal of this adaptation is to induce the user to develop and maintain the highest possible level of correlation between his or her intent and the signal features that the BCI employs to determine that intent. The algorithm can presumably accomplish these aims by rewarding better performance [69].

Chapter 5

BCI Task and Data

Two data sets are used in this thesis. The primary data set consists of EEG signals for imagined left and right hand movement. The second data set also includes tongue and foot movement. The goal is to recognize patterns in the EEG signal associated with each mental activity and to use them to perform a task.

The data to be used in this thesis have been made available by the Laboratory of Brain-Computer Interfaces at Graz University through two online competition [1, 2]. In this chapter I describe the task given to the subjects for which the data was collected, the data collection methodologies and the format of the data itself, for each of the two data sets.

5.1 BCI Competition II Data Set

This section describes the data set made available during BCI Competition II. This data set consists of single trials of spontaneous brain activity for imagined left and right hand movement, one part labeled and one part unlabeled.

5.1.1 Task

The data set was collected from one subject. The task given to her was to control a feedback bar by means of mental imagery. The subject was instructed to imagine left-hand movements when she wanted to make the bar grow to the left and right-hand movements when she wanted to make the bar grow to right. The order of left and right cues was random. Each trial the subject performed began with 2 seconds which were quiet. At $t = 2s$ an acoustic stimulus indicated the beginning of the trial and a cross "+" was displayed for 1 second; then at $t = 3s$, an arrow (left or right) was displayed as a cue, indicating that the subject was to make the bar grow in the direction of a the cue. During the trial, the subject was given feedback consisting of the direction and amount of growth of the bar. The timing scheme of one trial is illustrated in figure 5.1.

5.1.2 Physical Set-Up

The EEG data were recorded from a healthy 25 year old female subject. She sat in a relaxing chair with armrests and was asked to perform the task described above. The recording was made using a G.tec amplifier and Ag/AgCl electrodes. Three

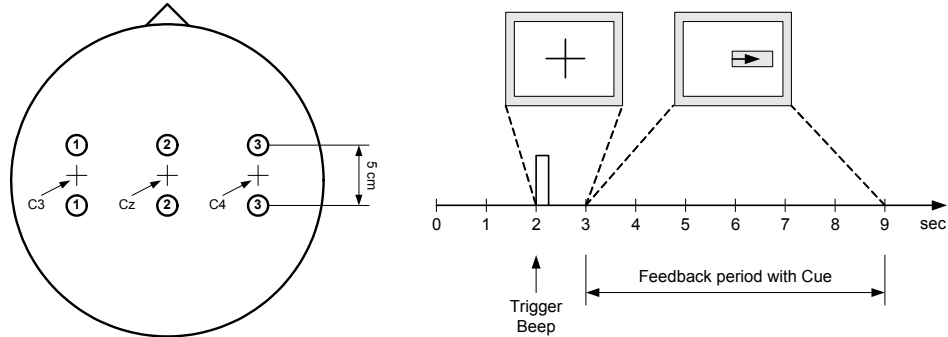


Figure 5.1: Electrode positions (left) and timing scheme (right). [1]

bipolar EEG channels were measured over C3, Cz and C4. The location of these channels on the human scalp is illustrated in figure 5.1. The EEG was sampled at 128Hz, and filtered between 0.5 and 30Hz. An example of one trial of EEG recordings is given in figure 5.2.

5.1.3 Experimental Design

The experiment consisted of 7 runs with 40 trials each, resulting in 280 trials each of which is 9 seconds in length. Because the EEG was sampled at 128Hz, there were 128 readings per second per channel, resulting in 1152 readings per trial per channel. The trials for training and testing were randomly selected in order to prevent any systematic differences between the two.

All runs were conducted on the same day with several minutes break in between. The feedback given to the subject was based on Adaptive Autoregressive (AAR) parameters of channels C3 and C4, combined with a discriminant analysis into one output parameter whose sign represented the direction of bar-growth and whose absolute value determined the length of the bar [60, 49].

5.1.4 Data Format

The data are provided in Matlab file format. It consists of 140 trials of training data and 140 trials of test data. Each trial contains 9 seconds of microvolt readings sampled at 128Hz from 3 EEG channels, as outlined in the task description. In addition, the training trials also contain labels indicating whether the performed motor imagery corresponded to left or right movement.

5.2 BCI Competition III Data Set

An second data set was used to test the flexibility of the model, and to extend its application to multi-class classification. The algorithm was optimized for the BCI Competition II data set and then applied to this second set to assess its flexibility.

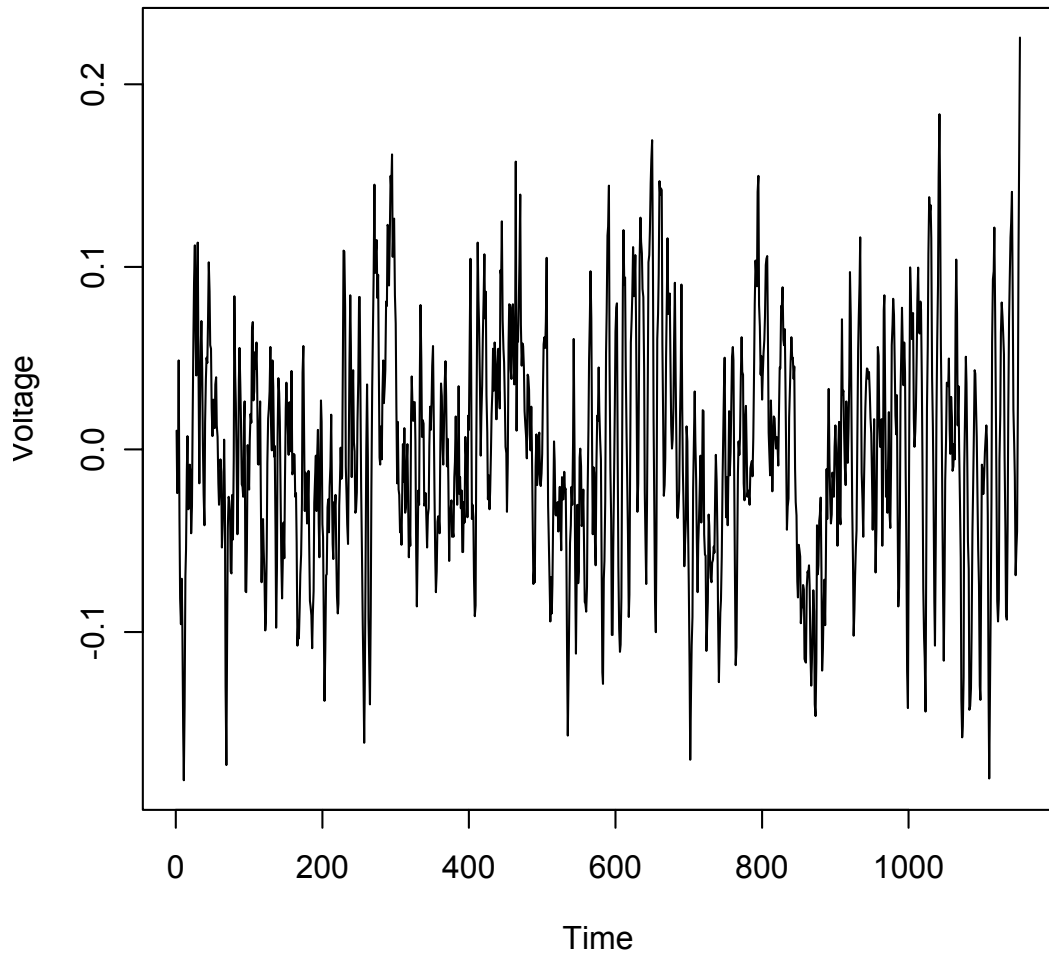


Figure 5.2: Example of EEG signal.

This data set was made available by the Laboratory of Brain-Computer Interfaces at Graz University during BCI Competition III [2]. The data set consists of multi-class motor imagery with four classes of imaginary movement (left hand, right hand, foot, and tongue). Data from 60 EEG channels was recorded for this task for three different subjects.

5.2.1 Task

As noted above, the task was to perform imaginary left hand, right hand, foot or tongue movements according to a cue. The order of cues was random. The first 2 seconds of each trial were quiet. At $t = 2s$ an acoustic stimulus indicated the

beginning of the trial, and a cross + was displayed. Then, from $t = 3s$, a cue in the form of an arrow to the left, right, up or down was displayed for 1 second, indicating that the subject was to imagine movement corresponding to the cue until the cross disappeared at $t = 7s$. The timing scheme described above is illustrated in figure 5.3.

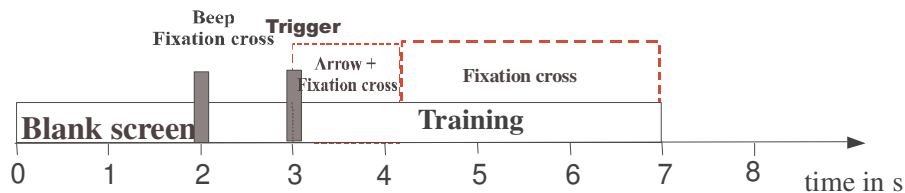


Figure 5.3: Timing of a trial. [2]

5.2.2 Physical Set-Up

The data was recorded from three subjects. During the experiment, a subject sat in a relaxing chair with armrests. The recording was made with a 64-channel EEG amplifier from Neuroscan, using the left mastoid for reference and the right mastoid as ground. The EEG was sampled with 250 Hz and filtered between 1 and 50Hz with Notchfilter on. Sixty EEG channels were recorded.

5.2.3 Experimental Design and Data Format

The experiment for each subject consists of several runs (at least 6) with 40 trials each. Each of the 4 cues was displayed 10 times within each run in a randomized order, resulting in 60 trials per class.

The data was provided in Matlab file format, in a separate file for each subject.

Chapter 6

Deformable Markov Model Templates for Waveform Pattern Matching in the context of BCIs

The first step in creating a BCI for the Graz task involves the application of a machine learning algorithm to characterize the patterns specific to each mental task. The algorithm explored in this thesis is a deformable Markov model for waveform pattern matching. In section 6.1 of this chapter, I will give a general description of the algorithm. In the following section, I will describe the preprocessing of the data, the construction of the model from data, and how the model can be applied to perform classification.

6.1 General Description

The algorithm explored in this thesis forms a probabilistic model for each type of imaginary movement. The same kind model is used for both types of movement, but the parameters learned for each of them are different. The construction of the model for one type of movement begins by segmenting each training trial into K pieces, using a standard segmentation algorithm, bottom-up segmentation. The K segments are formed so as to maximize the similarity of the data in each segment according to a specified criterion. For example, a piecewise linear representation of the signal minimizes the sum of squared residuals of the regression lines fitted through the data corresponding to each segment. An example of segmenting data into two similar pieces is presented in the figure 6.1:

Parameters corresponding to each of the K segments are learned by averaging the information for each segment over all 140 trials. In the case of piecewise linear segmentation, the parameter learnt can be the slope of a the regression line fitted through each segment.

This summary of the data into K segments is equivalent to the construction of a K -state deformable Markov model, each state of which corresponds to a segment in the representation, as described in section 6.4.

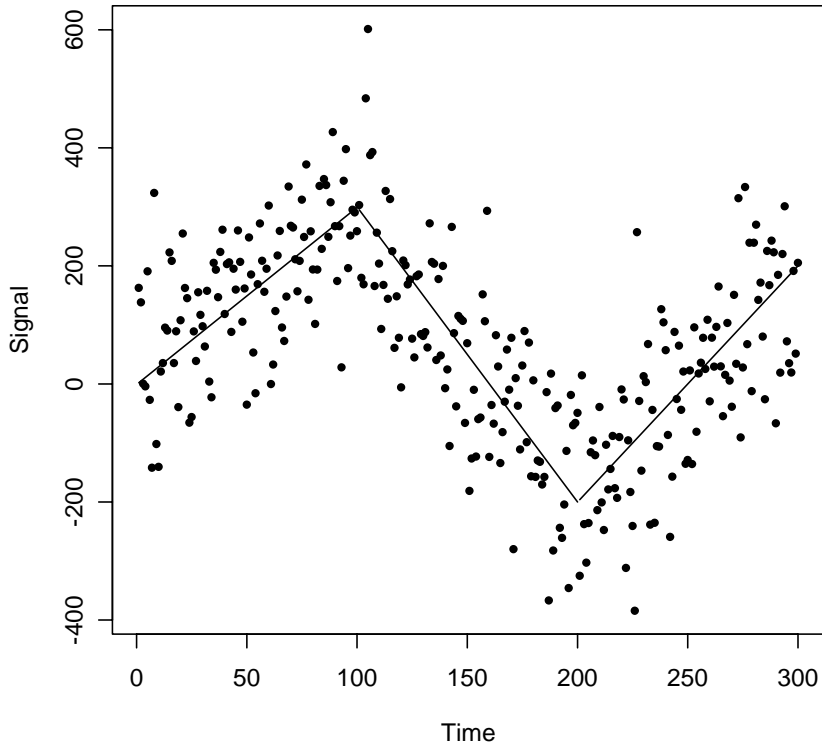


Figure 6.1: A simple illustration of a signal with three segments. The solid lines show the underlying deterministic components of the regression models within each state, and the points show the actual noisy observations.

Once we have a deformable Markov model for each type of imagined movement, a basic task is identifying the likelihood that a new data sequence was generated by each model. Each new trial is segmented into K pieces that maximize the similarity of each segment according to the criterion used for building the model. The likelihoods that the segmented data sequence was generated by each model are calculated, and the data trial is given the labeled corresponding to the highest likelihood. Some variations of the classification model use a dynamic programming algorithm to segment a new trial into K pieces in a way that maximizes the likelihood that the data was generated by each model. The maximized likelihoods corresponding to each model are then compared, and the data trial is given the label corresponding to the highest likelihood.

6.2 Bottom-Up Segmentation

The training algorithm implemented in this thesis begins by segmenting each data trial into K pieces. The segmentation is accomplished with the Bottom-Up algo-

rithm [33], a standard technique for time series segmentation. Given the signal for one trial, y_1, \dots, y_T , Bottom-Up segmentation starts from the finest possible approximation of the time series, where each pair of adjacent samples forms a segment. Next, the cost of merging each pair of adjacent segments is calculated. In the case of piecewise linear segmentation, the cost of merging two segments will be the difference in the sum of squared residuals of the regression line fitted through the new segment and the sum of the sum of squared residuals of the two lines fitted through the smaller segments. Then, the algorithm begins to iteratively merge the lowest cost pair until a stopping criterion is met (in this case, until K segments are left). When a pair of adjacent segments are merged, the cost of merging the new segment with its left and right neighbors is calculated. There is no need to recalculate the cost of merging any of the other segments. The number of segments that each data trial is divided into, K , is set empirically. The value of K used in this thesis is 5.

6.3 Hidden Markov Models

Once the training trials have been segmented into K segments each, we construct a K -state deformable Markov model (a special case of a hidden Markov model (HMM)), with each state corresponding to one segment in the piecewise representation.

A hidden Markov model (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters, and the goal is to determine the hidden parameters from the observable parameters [68]. The extracted model parameters can then be used to perform further analysis, for example for analyzing the similarity of two EEG signals.

In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible [68]. These variables will be denoted as θ_i in our model. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states.

Figure 6.2 shows the general architecture of an HMM. Each oval shape represents a random variable that can adopt a number of values. The random variable $x(i)$ is the value of the hidden variable of state i . The random variable θ_i is the value of the observed variable corresponding to state i . The arrows in the diagram denote conditional dependencies.

From the diagram, it is clear that the value of the hidden variable $x(i)$ (for state i) only depends on the value of the hidden variable $x(i-1)$ (for state $i-1$). This is called the Markov property. Similarly, the value of the observed variable $y(i)$ only depends on the value of the hidden variable $x(i)$.

A general HMM process can start in any state, according to an initial state distribution π , where π_i represents the probability of the process starting in state i . The process can then transition from one state to another according to a transition matrix A , where $A_{i,j}$ represents the probability of going to state j given that the process is in state i .

The probability of observing a sequence $\Theta = \theta_1, \dots, \theta_K$ of length K is given by:

$$P(\Theta) = \sum_X P(\Theta | X)P(X)$$

where the sum runs over all possible hidden node sequences $X = x(1), x(2), \dots, x(K)$.

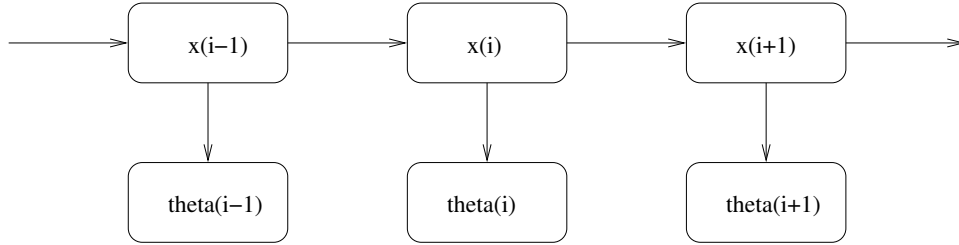


Figure 6.2: Architecture of an HMM. [68]

There are three problems associated with HMMs [68]:

- Given an output sequence or a set of such sequences, find the most likely set of state transition and output probabilities. In other words, train the parameters of the HMM given a data set of sequences.
- Given the parameters of the model, compute the probability of a particular output sequence.
- Given the parameters of the model, find the most likely sequence of hidden states that could have generated a given output sequence.

The solutions to the first, second and third problems for the deformable Markov Model used in this thesis are described in sections 6.4, 6.4.1 and 6.6.3 respectively.

For more information about HMMs and how to solve the associated three problems in general, see [57].

6.4 Constructing a Waveform Model from the Data

The deformable Markov model constructed in this thesis is a special case HMM. It is an adaptation of the model presented in [23]. The model describes a time series in which each of the K segments have to follow each other sequentially. Because one can only start from segment 1, the initial state distribution will be $\pi = [1, 0, \dots, 0]$. Furthermore, since the model must transition from segment 1 to segment 2, and so on, the transition matrix A will be diagonal ($A_{i,i+1} = 1$, $A_{i,j} = 0$ if $j \neq i+1$). The output probability distribution of state i will be of the form

$$p(y_{m+1}y_{m+2} \dots y_{m+d_i} | s_i) = p(d_i | s_i)p(\theta_i | s_i)$$

where

- $y_1 \dots y_T$ is the brain signal for one channel for one trial
- d_i = duration (length) of segment i in time
- $p(d_i | s_i)$ = left-truncated Gaussian with mean $mean(l_{i,n})$ and variance $var(l_{i,n})$, for all $n \in \{1 \dots numTrainingTrials\}$, where $l_{i,n}$ is the length of the i^{th} segment of the n^{th} trial

- θ_i = state i 's parameter (for example, the slope of the least square regression line estimate)

The training procedure essentially amounts to setting the means and variances in an appropriate data-dependent manner based on multiple waveform observations.

This model thus summarizes a time series by dividing it into K segments, each of which is described by a duration, d_i , and a parameter, θ_i . The probability that a portion of the observed time series, $y_{m+1}y_{m+2}\dots y_{m+d_i}$, corresponds to state i is determined by the probability that the length of this segment is consistent to the length of segments in state i and by the probability that the parameter θ of this segment is consistent to the parameter θ_i of state i .

6.4.1 Bottom-Up Segmentation for Computing the Probability of an Output Sequence

Once we have a deformable Markov model for a type of imagined movement, a basic task is identifying the likelihood that a new data sequence y_1, \dots, y_T was generated by that model.

We begin by segmenting the new data sequence into K pieces of lengths d_i , using the Bottom-Up algorithm described in section 6.2. Each piece is characterized by a parameter θ_i .

The likelihood that the segmented data sequence was generated by the model is calculated using the model parameters determined during the training phase:

$$p(y_1 \dots y_T | model) = \prod_{i=1}^K p(d_i | s_i) p(\theta_i | s_i)$$

6.5 Producing a Classification

For each testing trial, the algorithm calculates the likelihood that this trial is an example of imagined movement of each type. This is done by multiplying the probabilities that the signal recorded by each channel was generated by the model of imagined movement of that type corresponding to that channel:

$$p(y_1 \dots y_T | movement_i) = \prod_{c \in channels} p(y_1 \dots y_T | model_{i,c})$$

where $p(y_1 \dots y_T | model_{i,c})$ is given by the equation in the previous section.

The test example is labeled as being an example of movement generated by the model with the highest likelihood:

$$label = argmax_i \{p(y_1 \dots y_T | movement_i)\}$$

6.6 Variations of the Model

Piecewise linear segmentation was given as an example when describing the training and testing algorithms in sections 6.2, 6.4 and 6.4.1. Performing piecewise linear segmentation and treating the slope of the regression line fitted through each segment as a parameter does not summarize the Graz data set well, because the signal does not vary linearly in time. Another disadvantage of using the slope of a regression line as a parameter is the high performance cost: regression is computationally expensive. However, the data can be transformed to obtain easy-to-compute parameters that summarize the signal well.

6.6.1 Transforming the Data

Examining the plotted signals, such as the one in figure 5.2, one can observe that what seems to characterize different segments of the signal is the volatility of the signal on those segments, and not the slopes of the regression line fitted through the data points. There are regions of the signal where the difference in voltage between adjacent data points is very small, and regions for which this difference is very large. Thus, the initial voltage data is transformed into volatility data, as described below:

Given the original voltage signal,

$$y_0(1), \dots, y_0(T + 1),$$

the transformed volatility signal is:

$$y(i) = [y_0(i + 1) - y_0(i)]^2, \text{ for } i = 1 \dots T$$

6.6.2 Model 1

The first version of the algorithm uses the deformable Markov model with the transformed data.

The data is summarized into K segments, based on the similarity in the volatility of adjacent data points, using Bottom-Up segmentation. The cost of merging two adjacent segments s_i and s_j , with data points $y_{start_i}, \dots, y_{end_i}$ and $y_{start_j}, \dots, y_{end_j}$, with $start_j = end_i + 1$, is:

$$\sum_{t=start_i}^{end_j} \{y(t) - mean[y(start_i : end_j)]\}^2$$

The deformable Markov model is now formed using the K segments obtained after performing Bottom-Up segmentation. The parameter θ_k corresponding to each state (or segment) in the model is:

$$\theta_k = \frac{\sum_{t=start_k}^{end_k} y(t)}{end_k - start_k + 1},$$

the mean of the volatility signal corresponding to segment k .

The duration d_k of each segment is simply the length of that segment:

$$d(k) = end_k - start_k + 1$$

The means and variances in the probabilities modeled by the deformable Markov Model are set in a data-dependent manner, as described in section 6.4.

Once a model has been formed for each channel of the EEG signal, for each type of imaginary movement, the classification algorithm described in section 6.4.1 is applied to a testing example. The algorithm finds the likelihood that the signal is an example of each type of imaginary movement. The testing example is labeled to be an example of the imaginary movement with the largest likelihood, and the likelihood is returned as a measure of the confidence in the accuracy.

6.6.3 Model 2: Dynamic Programming Algorithm for Computing the Most Likely State Sequence

The second version of this algorithm explores using dynamic programming for testing. Training is performed as described for Model 1, to form a model for each type of imaginary movement.

In order to find the likelihood that a new signal is an example of movement of a specific type, the new trial is segmented into K pieces in a way that maximizes the likelihood that the data was generated by this model. In order to maximize the likelihood that signal y_1, \dots, y_T was generated by a certain model, we use a dynamic programming algorithm that is an adaptation of a Viterbi-like algorithm presented in [23]. At each time step t , the algorithm calculates the maximum likelihood that signal $y_1 \dots y_t$ was generated by the first i states of the model ($i \leq K$).

$$\hat{p}_i^{(t)} = \max_s \{p(s|y_1 \dots y_t) | s = s_1 \dots s_j, s_j = i\}$$

This is equivalent to finding the most likely segmentation of signal y_1, \dots, y_t into i segments similar to the first i segments of the model.

The recursive function for calculating $\hat{p}_i^{(t)}$ is:

$$\hat{p}_i^{(t)} = \max_{d_i} \{\hat{p}_{i-1}^{(t-d_i)} p(d_i | s_i) p(y_{t-d_i+1} \dots y_t | \theta_i)\}$$

The maximum likelihood that time series $y_1 \dots y_T$ was generated by the model is $\hat{p}_K^{(T)}$.

The dynamic programming algorithm is applied to a testing example to find the maximum likelihood that the signal was generated by each model of imaginary movement. The testing example is labeled to be an example of the imaginary movement with the largest maximum likelihood, and the likelihood is returned as a measure of the confidence in the accuracy.

6.6.4 Model 3

The third version of the algorithm explores using the dynamic programming algorithm described in section 6.6.3 for both training and testing.

In order to build a model for one type of imaginary movement, the first example of that type of movement is segmented using Bottom-Up segmentation, as in Model 1. For each of the following examples, the data is segmented using the Dynamic Programming algorithm in section 6.6.3, in the same way the test trials were segmented for Model 2. The current training example is compared to the model obtained from all previous examples. Then the parameters obtained using the dynamic programming segmentation are averaged with the parameters of the model obtained using all previous examples.

Testing is identical to that of Model 2.

6.6.5 Model 4

In order to give more flexibility to the algorithm, Model 4 allows for segments to be skipped.

Building a model for one type of imaginary movement begins with segmenting the first training example for that movement using Bottom-Up segmentation. The next training example is segmented using Bottom-Up segmentation, and the

highest-likelihood common subsequence of the new example and the existing model is computed. This way, segment j of the new example can be matched with a segment different from segment j in the existing model. Segments can also be dropped. The parameters for the matched segments are averaged with those for the existing model.

During testing, a new example is again segmented using Bottom-Up segmentation, and the likelihood of the highest-likelihood common subsequence is found for each type of movement. The model with the highest likelihood is picked as a label.

Chapter 7

Evaluation Metrics

Experiments and results using the algorithm described in the previous chapter will be presented in chapter 8. This chapter describes the evaluation metrics used to measure the performance of the algorithm and to compare it against the performance of other approaches.

7.1 Accuracy

Accuracy is the degree of conformity of a measured or calculated quantity to its actual (true) value. In the context of this thesis, accuracy is calculated as the percentage of correctly labeled trials (obtained by comparing the labels produced by the classification algorithm to the true labels) in the testing set.

7.2 Mutual Information

Given an observable process y_t consisting of a signal process x_t and an additional noise process e_t ($y_t = x_t + e_t$), it can be assumed that the signal and noise processes are distributed normally and are completely independent. Thus:

$$x_t = N(\nu_x, \sigma_x^2)$$

$$e_t = N(\mu_e, \sigma_e^2).$$

In the case of linearity and independence, the variance σ^2 of the process y_t is

$$\sigma^2 = \sigma_x^2 + \sigma_e^2$$

and the signal-to-noise-ratio (SNR) is defined as:

$$SNR = \sigma_x^2 / \sigma_e^2 = \sigma_y^2 / \sigma_e^2 - 1.$$

The entropy H of a Gaussian process is:

$$H(y) = 0.5 * \log_2(2\pi e \sigma_y^2).$$

The entropy difference between the variability of x and the variability of x given that it belongs to class c is the maximum mutual information between x and the class information c :

$$I = H(y) - H(y|c)$$

Written in terms of the signal-to-noise ratio, the maximum mutual information is:

$$I = 0.5 * \log_2(1 + SNR) = 0.5 * \log_2(\sigma_y^2 / \sigma_e^2)$$

The variance of the noise process σ_e is the average volatility within classes. The total variance σ_y^2 is the variance over all the trials.

Chapter 8

Experiments and Results

The goal is to provide an analysis system that, using the training set, can provide continuous feedback in real time, by inferring labels and their probabilities for the test set. This is why the training and testing algorithms need to assume that the data within each set is ordered, and that the order matters. An online algorithm would use the previously labeled examples to adjust the parameters of its model, and then attempt to label new examples while updating its parameters. Analyzing data in order is essential to this kind of adaptation that allows the algorithm to shift its parameters in time.

8.0.1 Model 3

Because Model 2 segments one training example using Bottom-Up Segmentation and then segments all subsequent examples so as to maximize the likelihood that they are similar to the first example, the training trial the algorithm begins with is important. Because the data is assumed to be ordered, and the subject receives feedback throughout the experiment, Model 3 uses training trials 40 – 70 instead of 1 – 70 to build a model for each imaginary movement. The training and testing procedures are identical to those of Model 2.

8.0.2 Model 5

Because the first 4 seconds in a trial are quiet, as stated in the description of the data set, Model 5 performs the same kind of analysis performed by Model 4, but it uses only the signal captured between $4s - 9s$.

8.0.3 Models 6 and 7

Examining the results of the baseline algorithm [60], it seems that the classification information for a trial is concentrated between $4s - 7s$. The Graz algorithm achieves maximum accuracy at $t = 6s$. This is why Model 6 uses only the signal captured between $4s - 7s$, and Model 7 uses the signal captured between $4s - 6s$. The analysis is identical to that of Model 5.

8.0.4 Model 9

Because Model 8 seemed to produce the highest accuracy for Channel C3, while Model 6 produced the highest accuracy for Channel C4, and the overall highest accuracy for the combination of the two channels, Model 9 explores applying Model 8 to Channel C3 and Model 6 to Channel C4 and combining their results based on the likelihoods obtained for each channel.

The models applied to the Graz task, described in the previous chapter, are summarized in the table below. Bottom-Up refers to the segmentation algorithm described in Section 6.2. DP refers to the dynamic programming algorithm of Section 6.5. LCS refers to the algorithm computing the most likely common subsequence as described in Section 6.7.8.

Model	Training	Testing	Training Examples	Signal
Model 1	Bottom-Up	DP	all	1s-9s
Model 2	DP	DP	all	1s-9s
Model 3	DP	DP	40-70	1s-9s
Model 4	Bottom-Up	Bottom-Up	all	1s-9s
Model 5	Bottom-Up	Bottom-Up	all	4s-9s
Model 6	Bottom-Up	Bottom-Up	all	4s-7s
Model 7	Bottom-Up	Bottom-Up	all	4s-6s
Model 8	LCS	LCS	all	4s-7s
Model 9	C3: LCS; C4: Bottom-Up	C3: LCS; C4: Bottom-Up	all	4s-7s

Table 8.1: Summary of Models

8.1 BCI Competition 2003 Data Set

The results of applying the models above to the Graz task presented at BCI Competition 2003 are summarized in the table below. This table lists the accuracy and maximum mutual information obtained by using each of the channels individually, the statistics obtained by multiplying the likelihoods of the two channels.

Model	% Acc C3	% Acc C4	% Acc C3+C4	MI C3	MI C4	MI C3+C4
Model 1	65	54	58	0.35	0.11	0.22
Model 2	57	57	62	0.21	0.20	0.31
Model 3	59	55	59	0.23	0.15	0.23
Model 4	57	61	60	0.21	0.30	0.28
Model 5	52	62	53	0.11	0.28	0.11
Model 6	65	75	81	0.35	0.40	0.45
Model 7	66	76	80	0.35	0.40	0.44
Model 8	67	68	79	0.36	0.37	0.43
Model 9	72	75	85	0.39	0.40	0.46

Table 8.2: Classification Accuracy and Mutual Information for Model Variations, for channels C3, C4, and their combination

The accuracy and mutual information obtained by using Bottom-Up segmentation for both training and testing, and analyzing only seconds 4 – 7 of the signal are much higher than those of the previous models. The highest accuracy (85%) and mutual information (0.46) were obtained by combining the Bottom-Up segmentation approach for the training and testing of channel C4, and the most Likely Common Subsequence for the training and testing of channel C3. These results are better than those provided by the baseline algorithm, and they are competitive with the results obtained by other participant in the BCI 2003 Competition.

To test whether training is really necessary or whether a new wave can simply be compared with the most recent examples of each type of imaginary movement, a new experiment was performed only on the test set. The examples were assumed to be ordered in time. Bottom-Up Segmentation was used for the first training example of each type, to form two models to compare against. Then, each new training example was segmented using Bottom-Up, and compared against the two models. The signal was then used as the new model for the wave corresponding to the type of movement it was an example of. The results are summarized in the table below:

Channels	% Acc
C3	44
C4	67
C3+C4	51

Table 8.3: Classification Accuracy obtained by comparing a new wave with the most recent examples of that wave for channels C3, C4, and their combination

This algorithm does not perform as well as Models 6 and 9, indicating that training is necessary for a stable BCI system.

8.2 BCI Competition 2005 Data Set

Model 6 was applied to the data set provided during BCI Competition 2005, for the signal between $t = 4s - 7s$, and compared to the baseline algorithm that provided feedback during the recording sessions. The results are summarized in the table below:

Subject	% Acc
K3	58
K6	45
L1	48

Table 8.4: Classification Accuracy obtained on 2005 Data Set for three subject, using the best channel described in [61], in order to be able to produce a comparison

Note that for this task, there are four possible classes, so the accuracy expected by chance is 0.25, not 0.5 as in the previous data set.

Subject	% Acc
K3	64
K6	36
L1	49

Table 8.5: Classification Accuracy obtained on 2005 Data Set for three subject, using the best combination of 3 channels described in [61], in order to be able to produce a comparison

8.3 Comparison to Other Algorithms

The results of the algorithm described in this thesis will be compared to the results of algorithms proposed by competitors in the BCI Competition 2003 [?]. The results of these algorithms are summarized in the table below, and the approaches are described in the following subsections.

Group	% Accuracy	Mutual Information
Graz	83	0.59
Group 1	89	0.61
Group 2	84	0.46
Group 3	83	0.45
Group 4	86	0.44
Group 5	76	0.26
Group 6	83	0.21
Group 7	68	0.09
Group 8	51	0.00

Table 8.6: Classification accuracy and mutual information for the algorithms submitted during the 2003 BCI competition

8.3.1 Adaptive Autoregressive Parameters and Linear Discriminant Analysis

The reference algorithm that the subject received feedback from used Adaptive Autoregressive Parameters combined with Linear Discriminant Analysis to produce a classification and the confidence in that classification.

An adaptive autoregressive (AAR) model describes a signal y_t in the following form [54, 60]:

$$y_t = \sum_{i=1}^p a_{i,t} y_{t-i} + E_t$$

where, in the ideal case, E_t is a purely random or white noise process with mean zero and variance σ_E^2 .

The AAR parameters are estimated using Least-Mean Square (LMS), characterized by the following update equations [60]:

$$E_t = y_t - a_{1,t-1}y_{t-1} - \dots - a_{p,t-p}y_{t-p}$$

$$a_{i,t} = a_{i,t-1} + cE_t y_{t-i} \quad i = 1 \dots p$$

$$c = UC / \text{var}\{Y\}$$

UC is an update coefficient which was chosen to be 0.004. The total power of the signal $\text{var}\{Y\}$ and the initial AAR parameters, $a_{i,0}$ were obtained by previous EEG recordings of the same subject. The model was chosen to have order $p = 6$.

The AAR parameters were combined with discriminant analysis into one output parameter $D(t) = w^T a(t) + w_0$. The weight vector w and threshold w_0 were obtained by linear discriminant analysis of previous sessions with the same subject.

The maximum mutual information of the algorithm was calculated using $D(t)$. The accuracy of this algorithm for the data set used in this thesis is 82%, and the maximum mutual information is 0.59.

8.3.2 Group 1

The best performing algorithm in the 2003 BCI competition was submitted by Schaefer et al [37].

The data were bandpass filtered using Morlet-Wavelets at the lower and higher frequency mu-rhythm, centered at 10Hz and 22Hz, respectively.

From the preprocessed training data, a multivariate normal distribution for each class (left/right) was estimated for every time instance t_i , $i = 1, \dots, T = 1152$. Furthermore, for every time instance, an upper bound for the Bayes error of the two probability models was obtained. Using this, a weight function was calculated according to the following equation:

$$w(t) = 0.5 - \text{BayesError}(t)$$

For online classification at time instance T the decision is based on all previous time instances $t \leq T$ using the expectation of the weighted probabilities:

$$p_t(\text{left}|y) = \frac{P_t(y_t|\text{left})}{P_t(y_t|\text{left}) + P_t(y_t|\text{right})}$$

$$D_T(\text{left}|y_1, \dots, y_T) = \frac{\sum_{t \leq T} w(t) p_t(\text{left}|y_t)}{\sum_{t \leq T} w(t)}.$$

To match the convention that negative (positive) values indicate left (right) trials, the result was transformed:

$$D'_T = 0.5 - D_T$$

where the magnitude reflects the confidence assigned to the actual decision.

8.3.3 Group 2

Narayana et al developed an algorithm using the Autoregressive Spectrum of the Signal and Linear Discriminant Analysis.

The AR spectrum for each window of 128 samples with a window shift of one sample was computed. The spectra were computed for both channels C3 and C4. The spectrum was divided into four frequency bands and the energy in each band was computed for both channels. The resulting band energies from channel C4 were divided by the energy of the corresponding frequency band of channel C3.

This resulted in a four dimensional feature vector. The features were classified using a Linear Discriminant Analyser (LDA), which used the Mahalanobis distance measure for classification. This resulted in the classification of the signal at every sample after an initial delay of 128 samples (or 1 second).

8.3.4 Group 3

Saffari et al developed an algorithm based on Adaptive Autoregressive parameters and neural networks.

During preprocessing, the signal was filtered using a highpass 8Hz filter. AAR parameters of order 6 were calculated for both the filtered and original C3 and C4 signals resulting in 12 AAR parameters for each time point.

Neural networks were trained on 2x11 different overlapping time regions of AAR parameters of filtered and original signals of C3 and C4 as follows: input time range was 128 samples (1 sec.) sliding 64 samples (500 m) for next network.

During testing, the results of all networks were averaged. The confidence in classification was calculated as the energy of classification signal up to the last time point.

8.3.5 Group 4

Gao et al used a Linear Discriminant Analysis based algorithm.

During preprocessing, they selecting the subject-specific frequency band of [10-12Hz] and filtered the signal with this bandpass filter. Then they computed the energy of each sample point of channels C3 and C4:

$$e(t) = \sum_{i=0}^{n-1} y(t-i)^2$$

where n is the length of a window. They constructed a feature vector for every trial and for every sampling point as:

$$E_t = [e_{C3(t)}, e_{C4(t)}]^T$$

The classification was produced using linear discriminant analysis (LDA):

$$D_t = W_T c^T * E_t + w_0$$

where D_t is a time-varying signed distance function (TSD) whose sign determines the class and whose absolute value determines the confidence in classification.

8.3.6 Group 5

Rissacher used a complexity algorithm (spectral entropy) for feature extraction and a feed-forward neural network trained with backpropagation for pattern recognition. The data considered for pattern recognition at a certain time-sample was that from the previous 1 second.

8.3.7 Group 6

Zander et al developed an algorithm using AAR parameters and ERD.

AAR parameters were calculated for channels C3 and C4 for each trial. A weight vector that reflects the difference in ERD between the given classes, was determined

using an optimization procedure. For this purpose the time course of the amplitude of the mu-rhythm was estimated by calculating the band power 9-11 Hz in the power spectrum obtained from the AAR models. This was performed for different choices of order and update coefficient of the AAR models. Some of the resulting features were combined to optimize the cross-validation performance of a linear classifier. For calculating the confidence at time t of a test trial, the trained classifier is applied to the (incomplete) trial up to and including t and the output is weighted by the ERD measure described above, and normalized by the intra-class variances (of the estimated classes).

8.3.8 Group 7

Del Ro Vera used an MLP neural network trained using backpropagation and Principal Component Analysis for feature extraction.

Signals are filtered to obtain the Mu rhythm which is associated with each signal. An ERD diagram of each filtered class set is computed in order to obtain the main features of each channel.

Principal component analysis is computed for each class and each channel, only for the interval between 4 a 5 seconds. Only the five principal components for each class and channel are used. This analysis results in 30 different patterns of 128 samples.

After that, signals are divided in windows (only the time period between 4-9 seconds is used) overlapped 127 samples from the previous (to achieve one classification for each sample). The windows obtained from the test signals are correlated with the ones obtained from the PCA analysis and only the maximum values are used (the maximum value shows the similarity between the mode and the signal).

This proceeding results in five values for each channel and each class. Each set of five values are condensed in one.

Resulting were six component vectors for each signal window, 1 for each channel and each class. These vectors were used to train an MLP backpropagation network with an inner layer with ten neurons, and the output layer with 24. This neural network was used to classify the test signals.

8.3.9 Group 8

Mbwana reduced the amount of data points in the training and test set through decimation. Features from each channel of the data sets were obtained using discriminant pursuit, each channel being analyzed separately. The predicted labels for the test set were obtained by using Support Vector Machine analysis.

8.3.10 Comparison

Model 9 performs better than the Graz baseline algorithm. It does better than most participants in the competition, with the exception of the winning algorithm proposed by Group 1.

8.3.11 BCI 2005 Baseline Algorithm

The results of the baseline algorithm for the second data set are summarized below:

Subject	% Acc
K3	57
K6	47
L1	49

Table 8.7: Classification Accuracy obtained for baseline algorithm on 2005 Data Set for three subject, using the best channel [61]

Model 6 has a similar performance to that of the baseline algorithm, which uses Adaptive Autoregressive parameter [61]. This shows that the model is easily extensible to multi-class tasks.

Subject	% Acc
K3	67
K6	39
L1	50

Table 8.8: Classification Accuracy obtained for baseline algorithm on 2005 Data Set for three subject, using the best combination of 3 channels [61]

Chapter 9

Analysis and Future Work

The algorithm implemented in this thesis provides a flexible, general-purpose method to recognize patterns in the EEG data associated with each mental activity. It is applicable to tasks requiring multi-class classification, and it provides a good probabilistic framework that determines the confidence in each classification. The algorithm can work with as little as one example of each class. It is an incremental algorithm: new examples can be added to the model very easily, by simply averaging the parameters of the existing model with those of the new example. A weighted average can allow more emphasis to be placed on recent examples, giving an adaptable algorithm.

An important conclusion is that fairly accurate classification is possible in 2 – 3 seconds. Also, reducing the patterns in the signal to examining the volatility of the signal for a specific segment is enough to provide 85% classification accuracy. The best performing algorithm is fast, allowing for real-time classification.

Exploring these models contributes to the field because it provides a simple, new method for recognizing patterns in EEG data associated with specific mental activities. This method does not exploit features of EEG signals identified by neuroscientists and it deviates from the AAR parameter and LDA analysis which seems to be prevalent in the field.

Comparing Model 9 with some of the other algorithms submitted during BCI Competition 2003 might be a slightly unfair comparison because a lot of these algorithms might take advantage of the fact that the subject is adapting to the Graz algorithm. Indeed, a lot of the proposed algorithms use AAR parameters or LDA for their analysis.

Better accuracy could be obtained using Models 6 or 9 if the subject received feedback from these algorithms, instead of the Graz algorithm. Experiments seem to indicate that subjects adapt significantly to the systems providing feedback. Model 9 achieved classification accuracy higher than that of the baseline algorithm, although it did not provide feedback to the subject, and it does not use features or classifiers similar to that of the baseline algorithm.

Bibliography

- [1] Bci competition 2003. http://ida.first.fraunhofer.de/projects/bci/competition_ii, 2003.
- [2] Bci competition 2005. http://ida.first.fraunhofer.de/projects/bci/competition_iii/, 2005.
- [3] C.W. Andersen, E.A. Stoltz, and S. Shamsunder. Multivariate autoregressive models for classification of spontaneous electroencephalographic signals during mental tasks. *IEEE Transactions on Biomedical Engineering*, 45:277–286, 1998.
- [4] F. Babiloni, F. Cincotti, L. Lazzarini, J. Millan, J. Mourino, M. Varsta, J. Heikkonen, L. Bianchi, and M.G. Marciani. Linear classification of low-resolution EEG patterns produced by imagined hand movements. *IEEE Transactions on Rehabilitation Engineering*, 8(2), 2000.
- [5] J.D. Bayliss. *A Flexible Brain-Computer Interface*. PhD thesis, University of Rochester, 2001.
- [6] J. Bhattacharya. Reduced degree long-range phase synchrony in pathological human brain. *Acta neurobiologiae experimentalis*, 61(4):309–318, 2001.
- [7] N. Birbaumer, A. Kubler, N. Ghanayim, T. Hinterberger, J. Perelmouter, J. Kaiser, I. Iversen, B. Kotchoubey, N. Neumann, and H. Flor. The thought translation device (TTD) for completely paralyzed patients. *IEEE Transactions on Rehabilitation Engineering*, 8(2), 2000.
- [8] B. Blankertz, G. Dornhege, M. Krauledat, K.-R. Muller, and G. Curio. The berlin brain-computer interface: Report from the feedback sessions. FIRST Reports.
- [9] G. Box and F.M. Jenkins. *Time Series Analysis: Forecasting and Control*. Oakland, Ca: Holden-Day, 1976.
- [10] G.R. Burkitt, R.B. Silberstein, P.J. Cadusch, and A.W. Wood. Steady-state visually evoked potentials and travelling waves. *Clinical Neurophysiology*, 111(2):246–258, 2000.
- [11] K.O. Cogger and K. Fanning. An introduction to adaptive logic networks with an application to audit risk assessment. <http://www.peakconsulting.com/>, 2006.

- [12] E.A. Curran and M.J. Strokes. Learning to control brain activity: A review of the production and control of EEG components for driving brain-computer interface (BCI) systems. *Brain and Cognition*, 51(3):326–336, 2003.
- [13] Cyberkinetics - Neurotechnology Systems. Braingate. <http://www.cyberkineticsinc.com/content/medicalproducts/braingate.jsp>, 2005.
- [14] T. DeMarse. Neural flight stabilization. <http://neural.bme.ufl.edu/downloads/index.php>, 2005.
- [15] Dobbelle. Artificial vision for the blind. <http://www.seeingwithsound.com/etumble.htm>, 2005.
- [16] E. Donchin, K.M. Spencer, and R.S. Wijesinghe. The mental prosthesis: Assessing the speed of a p300-based brain-computer interface. *IEEE Transactions on Rehabilitation Engineering*, 8:174–179, 2000.
- [17] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. New York: Wiley-Interscience, 2000.
- [18] A. Fedotchev, A.T. Bondar, and V.F. Konovalov. Stability of resonance EEG reactions to flickering light in humans. *International Journal of Psychophysiology*, 9:189–193, 1990.
- [19] G.N. Garcia, T. Ebrahimi, and J.-M. Vesin. Joint time-frequency-space classification of EEG in a brain-computer interface application. *EURASIP Journal on Applied Signal Processing*, 1(7):713–729, 2003.
- [20] G.N. Garcia, U. Hoffman, T. Ebrahimi, and J.-M. Vesin. Direct brain-computer communication through EEG signals. *IEEE EMBS Book Series on Neural Engineering*, 2004.
- [21] G.N. Garcia Molina. *Direct Brain-Computer Communication Through Scalp Recorded EEG Signals*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 2004.
- [22] X. Ge and P. Smyth. Deformable markov model templates for time-series pattern matching. Technical Report UCI-ICS 00-10, Department of Computer Science, University of California, Irvine, 2000.
- [23] Xianping Ge and Padhraic Smyth. Deformable markov model templates for time-series pattern matching. In *Knowledge Discovery and Data Mining*, pages 81–90, 2000.
- [24] C. Guger, A. Schloegl, C. Neuper, D. Walterspacher, T. Strein, and G. Pfurtscheller. Rapid prototyping of an EEG-based brain-computer interface (BCI). *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 9(1):49–58, 2001.
- [25] S. Gupta and H. Singh. Preprocessing EEG signals for direct human-system interface. In *IEEE International Joint Symposia on Intelligence and Systems*, 1996.

- [26] T. Hinterberger, A. Kubler, J. Kaiser, N. Neumann, and N. Birbaumer. A brain-computer interface (BCI) for the locked-in: Comparison of different EEG classifications for the thought translation device. *Clinical Neurophysiology*, 114, 416-425.
- [27] K. Inoue, R. Kajikawa, T. Nakamura, G. Pfurtscheller, and K. Kumamary. EEG signal analysis based on quasi-ar model - application to EEG signals during righ and left motor imagery. *SICE Annyal Conference in Sapporo*, 2004.
- [28] W. Jia, X. Zhao, H. Liu, X. Gao, S. Gao, and F. Yang. Classification of single trial EEG during motor imagery based on erd. In *Proceedings of 26th Annual International Conference of the IEEE EMBS*, 2004.
- [29] M. Kaper, P. Meinicke, U. Grossekaehofer, R. Lingner, and H. Ritter. BCI competition 2003 - data set iib: Support vector machines for the p300 speller paradigm. *IEEE Transactions on Biomedical Engineering*, 51(6), 2004.
- [30] B. Karmousi, Z. Liu, and B. He. Classification of motor imagery tasks for brain-computer interface applications by means of two equivalent dipoles analysis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 13(2), 2005.
- [31] S. Kelly, D. Burke, P. de Chazal, and R. Reilley. Parametric models and spectral analysis for classification in brain-computer interfaces. In *Proceedings of the IEEE International Conference on Digital Signal Processing*, 2002.
- [32] S.P. Kelly, E.C. Lalor, C. Finucane, G. McDarby, and R.B. Reilly. Visual spatial attention control in an independent brain-computer interface. *IEEE Transactions on Biomedical Engineering*, 52(9), 2005.
- [33] Eamonn J. Keogh, Selina Chu, David Hart, and Michael J. Pazzani. An online algorithm for segmenting time series. In *ICDM*, pages 289–296, 2001.
- [34] A. Kostov and M. Polak. Parallel man-machine training in development of EEG-based cursor control. *IEEE Transactions on Rehabilitation Engineering*, 8(2):203–204, 2000.
- [35] E.C. Lalor, S.P. Kelly, C. Finucane, R. Burke, R. Smith, R.B. Reilly, and G. McDarby. Steady state vep-based brain-computer interface control in an immersive 3d gaming environment. *EURASIP Journal on Applied Signal Processing*, 19:3156–3164, 2005.
- [36] E.C. Lalor, S.P. Kelly, C. Finucane, R. Burke, R. Smith, R.B. Reilly, and G. McDarby. Steady-state vep-based brain-computer interface control in an immersive 3d gaming environments. *EURASIP Journal on Applied Signal Processing*, 19:3156–3164, 2005.
- [37] S. Lemm, C. Schafel, and G. Curio. Bci competition 2003 - data set iii: probabilistic modeling of sensorimotor mu rhythms for classification of imaginary hand movements. *IEEE Transactions on Biomedical Engineering*, 51:1077–1080, 2004.

- [38] S. Lemm, C. Schafer, and G. Curio. BCI competition 2003 - data set iii: Probabilistic modeling of sensorimotor mu rhythms for classification of imaginary hand movements. *IEEE Transactions on Biomedical Engineering*, 51(6), 2004.
- [39] D.J. McFarland, L.M. McCane, S.V. David, and J.R. Wolpaw. Spatial filter selection for EEG-based communication. *Electroencephalography and clinical Neurophysiology*, 103:386–394, 1997.
- [40] M. Middendorf, G. McMillan, G. Calhoun, and K.S. Jone. Brain-computer interfaces based on steady-state visual-evoked response. *IEEE Transactions on Rehabilitation Engineering*, 8(2), 2000.
- [41] J.d.R. Millan and J. Mourino. Asynchronous BCI and local neural classifiers: an overview of the adaptive brain interface project. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):159–161, 2003.
- [42] J.d.R. Millan, J. Mourino, F. Babiloni, F. Cincotti, M. Varsta, and J. Heikkinen. Local neural classifier for EEG-based recognition of mental tasks. *IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 2000.
- [43] J. Muller-Gerking, G. Pfurtscheller, and H. Flyvbjerg. Designing optimal spatial filters for single-trial EEG classification in a movement task. *Electroencephalography and Clinical Neurophysiology*, 45:277–286, 1999.
- [44] G.R. Muller-Putz, R. Scherer, C. Brauneis, and G. Pfurtscheller. Steady-state visual evoked potential (ssvep)-based communication: impact of harmonic frequency components. *Journal of Neural Engineering*, 2:123–130, 2005.
- [45] NASA Intelligent Systems Division. Brain computer interface demonstration video. <http://ic.arc.nasa.gov/>, 2005.
- [46] Natures Publishing Group. Mental ping-pong could aid paraplegics. http://www.nature.com/news/2004/040823/pf/040823-18_pf.html, 2005.
- [47] Neil Squire Foundation. The brain interface project. <http://www.neilsquire.ca/section.asp?catid=123&subid=137&pageid=352>, 2006.
- [48] C. Neuper, A. Schloegl, and G. Pfurtscheller. Enhancement of left-right sensorimotor EEG differences during feedback-regulated motor imagery. *Journal of Clinical Neurophysiology*, 16(4):373–382, 1999.
- [49] C. Neuper, A. Schloegl, and G. Pfurtscheller. Enhancement of left-right sensorimotor EEG differences during feedback-regulated motor imagery. *Clinical Neurophysiology*, 16(4):373–382, 1999.
- [50] B. Obermaier, C. Neuper, C. Guger, and G. Pfurtscheller. Information transfer rate in a five-class brain computer interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 9(3):283–288, 2001.
- [51] W.D. Penny, S.J. Roberts, E.A. Curran, and M.J. Strokes. EEG-based communication: A pattern recognition approach. *IEEE Transactions on Rehabilitation Engineering*, 8(2):214–215, 2000.
- [52] G. Pfurtscheller and C. Neuper. Motor imagery and direct brain-computer communication. In *Proceedings of the IEEE*, volume 89, July 2001.

- [53] G. Pfurtscheller, C. Neuper, G.R. Muller, B. Obermaier, G. Krausz, A. Schloegl, R. Scherer, B. Graimann, C. Keinrath, D. Skliris, M. Wortz, G. Supp, and C. Schrank. Graz BCI: State of the art and clinical applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):177–180, 2003.
- [54] G. Pfurtscheller, C. Neuper, A. Schloegl, and K. Lugger. Separability of EEG signals recorded during right and left motor imagery using adaptive autoregressive parameters. *IEEE Transactions on Rehabilitation Engineering*, 6(3), 1998.
- [55] G. Pfurtscheller, Ch. Neuper, D. Flotzinger, and M. Pregenzer. EEG-based discrimination between imagination of right and left hand movement. *Electroencephalography and Clinical Neurophysiology*, 103:642–651, 1997.
- [56] G. Pfurtscheller, N. Neuper, C. Guger, W. Harkam, H. Ramoser, A. Schloegl, B. Obermaier, and M. Pregenzer. Current trends in graz brain-computer interface (BCI) research. *IEEE Transactions on Rehabilitation Engineering*, 292:211–214, 2000.
- [57] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. pages 267–296, 1990.
- [58] A. Schloegl and G. Flotzinger, D. Pfurtscheller. Adaptive autoregressive modelling used for single-trial EEG classification. *Biomedical Technik*, 42:162–167, 1997.
- [59] A. Schloegl, F. Lee, H. Bischof, and G. Pfurtscheller. Characterization of four-class motor imagery EEG data for the BCI-competition 2005. *Journal of Neural Engineering*, 2:L14–L22, 2005.
- [60] K. SchloeglA., and Lugger and G. Pfurtscheller. Using adaptive autoregressive parameters for a brain-computer-interface experiment. In *Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 19, pages 1533–1535, 1997.
- [61] A. Schloegl, F. Lee, H. Bischof, and G. Pfurtscheller. COMMUNICATION: Characterization of four-class motor imagery EEG data for the BCI-competition 2005. *Journal of Neural Engineering*, 2:L14–L22, December 2005.
- [62] S. Sutton, B. Braren, J. Zubin, and E.R. John. Evoked correlates of stimulus uncertainty. *Science*, 150:1187–1188, 1965.
- [63] G. Townsend, B. Graimann, and G. Pfurtscheller. Continuous EEG classification during motor imagery - simulation of an asynchronous BCI. *IEEE Transactions of Neural Systems and Rehabilitation Engineering*, 12(2), 2004.
- [64] L.J. Trejo, K.R. Wheeler, C.C. Jorgensen, R. Rosipal, S.T. Clanton, B. Matthews, A.D. Hibbs, R. Matthews, and M. Krupka. Multimodal neuroelectric interface developments. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11:199–204, 2003.

- [65] D.M. Tumeay, P.E. Morton, D.F. Ingle, C.W. Downey, and J.H. Schnurer. Neural network classification of EEG using chaotic preprocessing and phase-space reconstruction. In *Proceedings of the Seventeenth IEEE Annual Northeast Bio-engineering Conference*, 1991.
- [66] W.G. Walter, R. Cooper, V.J. Aldridge, W.C. McCallum, and A.L. Winter. Contingent negative variation: an electric sign of sensorimotor association and expectancy in the human brain. *Nature*, 230:380–384, 1964.
- [67] T. Wang, J. Deng, and B. He. Classification of motor imagery EEG patterns and their topographics representation. In *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, September.
- [68] Wikipedia. Hidden markov models. http://en.wikipedia.org/wiki/Hidden_Markov_Model, 2007.
- [69] J.R. Wolpaw, N. Birbaumer, D.J. McFarland, G. Pfurtscheller, and T.M. Vaughn. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113:767–791, 2002.
- [70] J.R. Wolpaw, D.J. McFarland, and T.M. Vaughan. Brain-computer interface research at the wadsworth center. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 8:222–226, 2000.