A User-Customized Self-Paced Brain Computer Interface

by

Hossein Bashashati

B.Sc., University of Tehran, 2008

M.Sc., University of Tehran, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL
STUDIES

(Electrical and Computer Engineering)

The University of British Columbia
(Vancouver)

April 2017

Abstract

Much attention has been directed towards synchronous Brain Computer Interfaces (BCIs). For these BCIs, the user can only operate the system during specific system-defined periods. Self-paced BCIs, however, allow users to operate the system at any time he/she wishes. The classification of Electroencephalography (EEG) signals in self-paced BCIs is extremely challenging, as the BCI system does not have any clue about the start time of a control task. Also, the data contains a large number of periods during which the user has no intention to control the BCI.

For sensory motor self-paced BCIs (focus of this thesis), the brain of a user goes through several well-defined internal state changes while performing a mental task. Designing classifiers that exploit such temporal correlations in EEG data can enhance the performance of BCIs. It is also important to customize these BCIs for each user, because the brain characteristics of different people are not the same.

In this thesis, we first develop a unified comparison framework to compare the performance of different classifiers in sensory motor BCIs followed by rigorous statistical tests. This study is the largest of its kind as it has been performed on 29 subjects of synchronous and self-paced BCIs. We then develop a Bayesian optimization-based strategy that automatically customizes a synchronous BCI based on the brain characteristics of each individual subject. Our results show that our automated algorithm (which relies on less sophisticated feature extraction and classification methods) yields similar or superior results compared to the best performing designs in the literature.

We then propose an algorithm that can capture the time dynamics of the EEG signal for self-paced BCI systems. We show that this algorithm yields better results compared to several well-known algorithms, over 13 self-paced BCI subjects. Finally, we propose a fully automatic, scalable algorithm that customizes a self-paced BCI system based on the brain characteristics of each user and at the same time captures the dynamics of the EEG signal. Our final algorithm is an important step towards transitioning BCIs from research environments to real-life applications, where automatic, scalable and easy to use systems are needed.

Preface

This thesis presents the research conducted by Hossein Bashashati, in collaboration with Prof. Rabab K. Ward, Dr. Ali Bashashati, Dr. Gary Birch and Dr. Amr Mohamed.

Below is the list of the scientific papers by Hossein Bashashati that were published (or submitted) through the course of his studies as a Doctor of Philosophy (PHD) candidate at University of British Columbia (UBC).

J1 - H. Bashashati, R. K. Ward, G. H. Birch and A. Bashashati, Comparing Different Classifiers in Sensory Motor Brain Computer Interfaces, PloS one. 2015 Jun 19.

J2 - H. Bashashati, R. K. Ward and A. Bashashati, User-Customized Brain Computer Interfaces Using Bayesian Optimization, Journal of neural engineering, 2016 Jan 29.

C1 - H. Bashashati, R. K. Ward and A. Bashashati, Hidden Markov Support Vector Machines for Self-Paced Brain Computer Interfaces, International Conference on Machine Learning Applications (IEEE ICMLA), 2015.

C2 - H. Bashashati, R. K. Ward and A. Bashashati, Bayesian Optimization of BCI parameters, IEEE Canadian Conference on Electrical and Computer Engineering (IEEE CCECE), 2016

C3 - H. Bashashati, R. K. Ward, A. Bashashati and A. Mohamed, Neural Network Conditional Random Fields for Self-Paced Brain Computer Interfaces, Accepted in International Conference on Machine Learning Applications (IEEE ICMLA), 2016.

Hossein Bashashati (primary author and the main contributor of the papers) proposed the algorithms, implemented them, and performed the evaluations.

Dr. Rabab K. Ward supervised the development of work, and provided technical feedback as well as editorial comments throughout writing of all the papers (J1, J2, C1, C2 and C3).

Dr. Ali Bashashati provided technical feedback as well as editorial comments throughout writing of the papers (J1, J2, C1, C2 and C3).

Dr. Gary Birch helped in manuscript writing for J1. Dr. Amr Mohamed helped in manuscript writing for C3.

A version of Chapter 2 appeared in J1. A version of Chapter 3 appeared in J2 and C2. A version of Chapter 4 appeared in C1 and C3.

Table of Contents

Abstract . . . . **ii**

Preface . . . . **iv**

Table of Contents . . . . **vi**

List of Tables . . . . **ix**

List of Figures . . . . **xiii**

Acknowledgments . . . . **xv**

1 Introduction **1**

1.1 Different Neurological Phenomena 2

1.2 Self-Paced vs. Synchronous BCIs 4

1.3 Block Diagram of a BCI 5

1.4 Properties of EEG Signal 6

1.5 Motivations and Contributions 8

2 A Unified Comparison Framework for Sensory-Motor BCIs **15**

2.1 Introduction 15

2.2 General Structure of Comparison Framework 17

2.2.1 Spatial and Frequency Filtering 18
2.2.2 Feature Extraction 19
2.2.3 Classification 20
2.2.4 Performance Measure 24
2.2.5 Model Selection 25
2.2.6 Statistical Tests 26
2.3 Datasets 28
2.4 Results and Discussion 30
2.5 Conclusion 40

3 Customizing Brain Computer Interfaces 41
3.1 Introduction 41
3.2 Bayesian Optimization 43
3.2.1 Modeling Posterior Distribution 44
3.2.2 Acquisition Function 47
3.3 Proposed Algorithm 48
3.3.1 Learning Algorithm (Objective Function) 50
3.3.2 Selecting Hyper-Parameters in BCIs 52
3.3.3 Results Aggregation 53
3.3.4 Termination of the Algorithm 55
3.3.5 Time Complexity of the Algorithm 55
3.4 Results and Discussion 56
3.4.1 Comparing with Other Hyper-Parameter Search Algorithms 57
3.4.2 Comparing the Results with Literature 63
3.4.3 Effect of Number of Optimization Iterations 66
3.5 Conclusion 68

4 Discriminative Sequence Learning Algorithms for Self-paced BCIs 71

4.1 Introduction 71

4.2 EEG Sequence Labeling with Hidden Markov Models 74

4.3 Discriminative Sequence Labeling Algorithms 75

4.3.1 Hidden Markov Support Vector Machines 76

4.3.2 Conditional Random Fields 78

4.3.3 Neural Network Conditional Random Fields 80

4.4 Datasets 82

4.5 Results 82

4.6 Conclusion 84

5 Ensemble of Discriminative Sequence Labeling Classifiers for Self-Paced BCIs 86

5.1 Introduction 86

5.2 Results 89

5.3 Conclusion 92

6 Conclusion and Future Work 93

6.1 Summary of Our Contributions 93

6.2 Future Work 98

Bibliography 100

List of Tables

Table 2.1 List of classifier parameters that were optimized using cross-validation in the training phase. 27

Table 2.2 Specification of datasets used in this thesis. 28

Table 2.3 The accuracy of classifiers for synchronous BCI operation over all subjects. The accuracies are calculated using the unseen test dataset. For each classification algorithm the first column shows the results of BP features and the second column shows the results of Morlet features. . . 32

Table 2.4 The Area Under the Curve (AUC) of classifiers for self-paced subjects. For each classification algorithm the first column shows the results of BP features and the second column shows the results of morlet features. . 34

Table 2.5 Average Rankings of the classification algorithms for both synchronous and self-paced datasets. The number in the parenthesis corresponds to the average rank of the algorithm among different subjects. For each feature extraction method the classifiers typed in bold are the recommended ones. The recommended classifiers are selected based on the results of the statistical tests. 36

Table 2.6 P-values corresponding to pairwise comparison of different classifiers. α is chosen to be 0.1. For settings 1 and 2 all hypothesis with p-value less than 0.0333 are rejected. For setting 3 and 4 all hypothesis with p-value less than 0.05 are rejected. The results are rounded up to 4 decimal places. 37

Table 2.7 Average Rankings of the classification algorithms for binary and multi-class classification in synchronous datasets. The number of subjects in binary task was 12 and the number of subjects in multi-task BCIs was 9. The number in the parenthesis corresponds to the average rank of the algorithm among different subjects. For each feature extraction method the classifiers typed in bold are the recommended ones. The recommended classifiers are selected based on the results of the statistical tests. 38

Table 2.8 P-values corresponding to pairwise comparison of different classifiers. α is chosen to be 0.1. For binary task BCIs with BP features all hypothesis with p-value less than 0.02 are rejected. For multi-task BCIs with BP features all hypothesis with p-value less than 0.0333 are rejected. For binary task BCIs with Morlet features all hypothesis with p-value less than 0.1 are rejected. For multi-task BCIs with Morlet features all hypothesis with p-value less than 0.025 are rejected. The results are rounded up to 4 decimal places. 39

Table 3.1 Accuracy of each algorithm with Morlet features. For each search algorithm we used MLR, voting (VOTE) and averaging (AVG) for aggregating the results of the classifiers. MIN is the result of the best level 1 classifier in optimization phase. Highlighted cells show the configuration for which the performance is the best. 59

Table 3.2 Accuracy of each algorithm with BP features. For each search algorithm we used MLR, voting (VOTE) and averaging (AVG) for aggregating the results of the classifiers. MIN is the result of the best level 1 classifier in optimization phase. Highlighted cells show the configuration for which the performance is the best. 60

Table 3.3 Average ranking and average accuracy of the optimization algorithms with (a) Morlet features and (b) BP features. 62

Table 3.4 P-values corresponding to pairwise comparison of different hyper-parameter search algorithms (a) Morlet features and (b) BP features. α is chosen to be 0.05. Bergmann's procedure rejects hypotheses 9 and 10 for the Morlet features. For the BP features, hypotheses 7, 8, 9 and 10 are rejected. The results are rounded up to 3 decimal places. 64

Table 3.5 Accuracy of our algorithms compared to the results reported in the literature for dataset III3b. Highlighted cells show the best performing algorithm. The text inside the parenthesis corresponds to the original label of the subjects used in the BCI competition. 64

Table 3.6 Accuracy of our algorithms compared to the results reported in the literature for dataset IV2b. Highlighted cells show the best performing algorithm. The text inside the parenthesis corresponds to the original label of the subjects used in the BCI competition. 65

Table 3.7 Accuracy of our algorithms compared to the results reported in the literature for dataset IV2a. Highlighted cells show the best performing algorithm. The text inside the parenthesis corresponds to the original label of the subjects used in the BCI competition. 65

Table 4.1 The results of comparing AUC of different algorithms on the test dataset. Highlighted cells show the algorithm for which the performance is the best. The last row shows the average rank of different classifiers across all subjects. 83

Table 5.1 The results of comparing different classifier after performing Bayesian optimization on the test dataset. MIN is the results of applying the best performing classifier in optimization phase on the test data. AVG corresponds to the results of the ensemble of classifiers. Highlighted cells show the algorithm for which the performance is the best. 90

List of Figures

Figure 1.1 (a) Self-paced BCI in which the user can control the system at any time (b) Synchronous BCI in which the user is restricted to control the system at system-defined periods. 4

Figure 1.2 A simple block diagram of an EEG-based BCI. 6

Figure 1.3 Our proposed hyper-parameter optimization in BCIs. 12

Figure 2.1 The BCI framework. Our framework has three main steps: 1. Filtering, 2. Feature Extraction and 3. Classification. The output of each step is fed to the next step. 17

Figure 3.1 An example of sequences of trials in BCI calibration phase in which each trial is 8s. In the first trial, the subject performed a left hand (L1) imagery movement followed by a No-Control (NC1) interval of 4s. In the second trial, the subject performed a right hand imagery movement (R1) followed by a 4s NC interval. 51

Figure 3.2 Accuracy of our algorithm with the BP features versus the number of iterations. The label of cross validation curves finish with '_CV' and the label of test data accuracies finish with '_Test'. 67

Figure 3.3 Accuracy of our algorithm with the Morlet features versus the number of iterations. The label of cross-validation curves finish with '_CV' and the label of test data accuracies finish with '_Test'. 68

Figure 5.1 Diagram of our optimization algorithm at iteration t. 88

Figure 5.2 Average AUC of different settings for each classifier across all subjects. The blue bars correspond to the average AUC when the default value of the hyper-parameters have been used. The red bars correspond to the average AUC after using Bayesian optimization. The green bars correspond to the average AUC when we used an ensemble of different classifiers. 91

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Rabab K. Ward. She has provided me with full support and encouragement throughout my studies. She has always provided time, helpful advice and guidance, and displayed great understanding during challenging times.

Finally, I would like to express my deepest gratitude to my parents and siblings, for their understanding, love, encouragement and many years of support.

This work was funded by the National Science of Engineering Research Council of Canada (NSERC) and Qatar National Research Fund (QNRF).

Chapter 1

Introduction

A Brain Computer Interface (BCI) aims at detecting the presence of specific patterns in a person's brain activity. These patterns relate to the user's intention to control a device [102]. If such patterns are detected in the brain waves, then the BCI issues specific signals to control the device.

Traditionally, the most popular application of BCIs has been to assist people with different kinds of motor impairments to communicate with the external world [78]. A BCI as a non-muscular communication channel allows disabled people to control different assistive devices. Besides rehabilitation applications, many new BCI applications have emerged recently [101] [17] [1]. These applications include playing video games, mental state monitoring, training and education, cognitive biometrics etc.

There are few different methods to record the brain activity of a person. Not all of these however are practical to use in real life applications. There are at least three factors that determine the practicality of BCIs. These factors include risk in use, signal temporal resolution and device portability. Invasive BCIs such as (implanted) electrocorticography-based ones record the brain activity from within the skull. Although the recorded signal has a high signal to noise ratio and is less prone to artifacts, the practicality of these methods is limited because of the need for the surgical installation of the electrodes inside the skull (risk of use) [66]. Magnetoencephalography and functional Magnetic Resonance Imaging are non-invasive but bulky, non-portable, and have very low temporal resolution.

Among the various means to measure the brain activity, the non-invasive EEG-based BCI systems can measure the changes in the brain activity within milliseconds (high temporal resolution); they are inexpensive, non-invasive and portable [75]. However, they also have several drawbacks. The signal to noise ratio of the EEG signal is low, i.e., the signals have very low amplitude (i.e. about 10 to 100 micro volts) compared to the background noise. They are also contaminated by artifacts. Therefore, detecting the intentions of a person from his/her measured EEG brain signals is a challenging task and has been at the forefront of research. Furthermore, the resolution of the EEG brain signals referred to as the spatial resolution is low and dependent on the number of electrodes that could be placed on the subject's head.

1.1 Different Neurological Phenomena

Present-day EEG-based BCIs operate based on four major electrophysiological activities of the brain [75].

Visual Evoked Potential (VEP) are typically generated in the EEG signal in response to a visual stimulus while the user is looking at the screen. An example of such a BCI is frequency-VEP in which a set of letters are flashing at different frequencies. When the user gazes at a letter, the BCI recognizes the intended letter. This type of BCI does not need any training but the drawback is that the user should stare at the screen while operating the BCI. Therefore, it is not suitable for some users such as patients at late stages of ALS disease [13].

P300 evoked potentials are generated in response to an irregular stimulus amongst several different stimuli. 300 milliseconds after the appearance of an infrequent stimulus a positive peak appears in the EEG signal of the subject. An example of such a BCI is when a grid of letters is shown to the subject. Each time one of the rows or columns of the grid flashes. The subject focuses on the letter he/she wants to communicate. When the row or the column containing the character flashes, a P300 evoked potential is generated [37].

Like VEP-based BCIs, P300 BCIs do not need user training. However, their need for external stimulus restricts their applicability. Another drawback is that after sometime of using a P300-based BCI, the user gets accustomed to the infrequent stimulus and the detection of the peak becomes difficult [84]. Also, during the whole operation of the BCI, the user should stare at the screen [40].

Slow Cortical Potentials (SCP) are slow voltage changes in the EEG generated by the subject. SCP usually lasts between 0.5 to several seconds. Subjects need training to self-regulate these SCPs, and usually the training needs up to several months of practice. Self-regulation performance varies for different subjects and some subjects might not be able to master SCP control. Besides, many factors such as the physiological and physical state of the subjects can affect the performance of the SCP-based BCIs [48].

Sensory-motor BCIs rely on the changes in sensory-motor rhythms of the brain. Sensory-motor rhythms consist of the mu, beta and gamma rhythms of the brain. The mu rhythms of the brain are generally in the range 8-12 Hz, the beta band is 13-30 Hz and the gamma is in the range 30-100 Hz.

Among the different ways to operate a BCI, those that rely on the sensory motor rhythms have been of special interest, e.g. [71] [58] [89]. Unlike VEP-based and P300-based BCIs, these kinds of BCIs do not need any stimuli from outside to control the system, and the subject can learn to generate the appropriate brain pattern to control a device voluntarily. The training process for sensory-motor based BCIs is not lengthy like SCP-based BCIs. Of particular interest is that a motor imagery activity (i.e., an imagined movement) generates patterns in the EEG that are similar to those generated by actual movements [70].

In addition, BCIs that are motor imagery-based are especially suitable for disabled people who do not have control over their limbs [59].

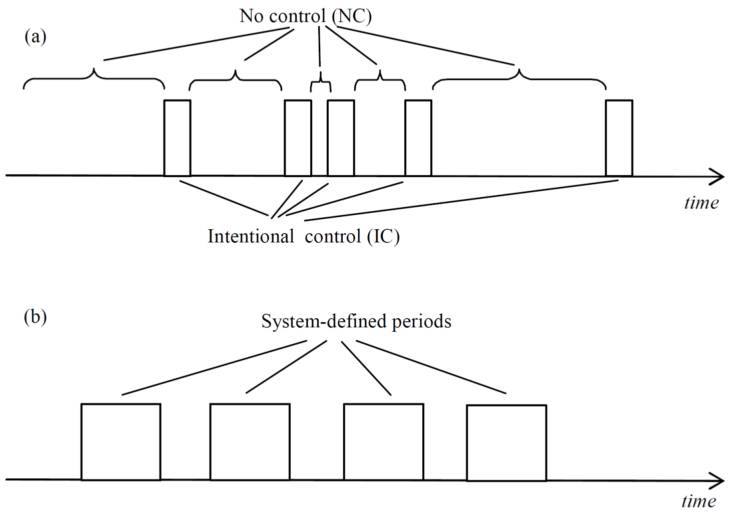

Figure 1.1: (a) Self-paced BCI in which the user can control the system at any time (b) Synchronous BCI in which the user is restricted to control the system at system-defined periods.

1.2 Self-Paced vs. Synchronous BCIs

BCI systems can be classified into two categories: synchronous and self-paced systems [69] [9]. In synchronous BCIs, subjects control the BCI output during specific short periods. That is, the user can only issue a control command when he is prompted to operate the system, therefore, the onset of the mental task is known beforehand. Figure 1.1 part (b) shows a synchronous BCI system.

Self-paced BCIs, on the other hand give the users the option of controlling the system whenever they wish to. The periods during which the user is controlling the system are called control states and those during which the user is not controlling the system are called No-Control (NC) states. During the NC states, the BCI system should not issue any control signal i.e. a false positive output should not be produced. In Figure 1.1 part (a) a self-paced

BCI is depicted.

Compared to synchronous BCI systems, designing self-paced BCIs remains extremely challenging. The challenges faced by self-paced BCIs stem from the following two properties of these BCIs:

1) In self-paced BCIs, the user can control the system at any time. Therefore, the BCI system does not have any clue about the start time of a control command. Also, the BCI should analyze and classify the EEG data at all times. This makes the design and evaluation of these BCIs more challenging.

2) During the NC periods, the user can do whatever he/she wants and can be in any mental state (except for those intended to control the BCI). Therefore the vast majority of EEG signals is formed of NC periods. This means that self-paced BCI data contains a large variety and number of periods during which the user has no intention to control the BCI. This makes it difficult to find specific patterns in the EEG during NC periods and therefore classification and controlling false positives associated to NC states become very challenging.

1.3 Block Diagram of a BCI

Figure 1.2 shows a simple block diagram of the signal processing part of an EEG-based BCI. The main two components are Feature Extraction and Classification.

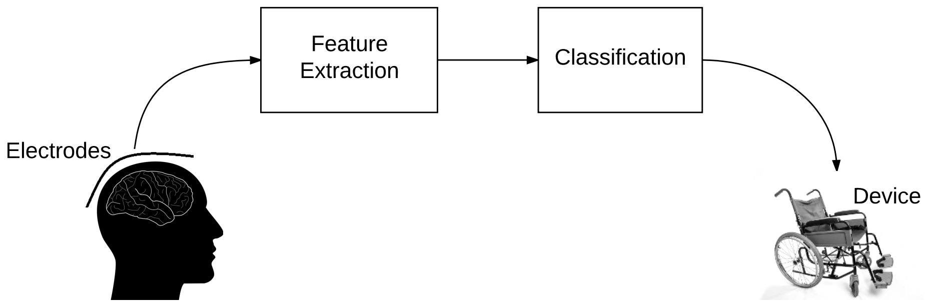

Figure 1.2: A simple block diagram of an EEG-based BCI.

The aim of the feature extraction block is to extract useful information i.e. specific patterns from the brain signal. The features capture the characteristics of the brain signals which are then used by the classification block to distinguish different mental tasks. Different mental tasks produce different patterns in the brain signals. A good feature extraction algorithm should be able to capture the information so that distinguishing between different mental tasks becomes easy.

The features should be able to capture time information, as to when certain changes in the brain signal occur. Frequency information should be considered because many mental tasks are frequency specific. Also, spatial information should be exploited because different mental tasks are related to different parts of the brain.

The aim of the classification block is to use the extracted features to detect the intention of the subject. Classifiers are calibrated using a small set of collected samples in the training sessions. The calibrated classifier is used in the test sessions on unseen data. Several different classifiers have been applied in different BCI studies. In general, simpler algorithms are preferred to more complex algorithms [8][75].

1.4 Properties of EEG Signal

The following properties of the EEG signal makes classification of the data challenging:

Data is noisy and contains outliers. The amplitude of the EEG signal is small and in the order of μVs, and the signal is usually contaminated with different types of noise (or artifacts). The noise in the EEG signal can have physiological sources such as electrooculogram and electromyogram signals or it might originate from outside sources such as power-line noise [38].

Specific variations in the EEG signal over time should be taken into account. These variations might happen in specific frequency bands. When a subject performs a sensory motor mental task, the amplitude of the signal changes in specific frequency bands. It is worth noting that these frequency bands vary from one person to another i.e. they are subject specific [102]. Therefore, besides the time information in the EEG signal, the frequency information should also be taken into consideration when designing feature extraction and classification algorithms. Equally important is that the BCI should be customized for each person.

The EEG is of high dimensions. The EEG signal is collected by several electrodes located on the surface of the scalp. Each electrode is referred to as an EEG channel. Depending on the mental task being performed, some locations on the scalp might provide more discriminative information than the other locations (for classifying the mental tasks). It is usually important to apply channel reduction algorithms to reduce the number of signals we are dealing with; this reduces the dimension of the feature space. It has been shown that the performance of the BCI is affected by the choice of the selected channels [45][5][56].

The EEG signal is non-stationary. The statistical properties of the EEG signal change over time [57]. These changes originate from different causes such as changes in the mental state of the subject, etc. Subsequently, the BCI needs to adapt according to the mental state of the subject.

BCI datasets are usually very small. Since collecting data and subject training are very time consuming processes, the available EEG datasets are usually very small. This makes it impossible to apply some powerful classifiers e.g. deep learning [62] to this data.

1.5 Motivations and Contributions

Even though much interest and progress have been made during the past two decades in BCI research, this field is still at its infancy. Many improvements are still necessary in order to encourage its widespread adoption. Current BCI systems are not accurate enough and are far from ideal for use in on-line practical settings. Particularly, for noninvasive BCIs which are by far the most widely used BCIs, the characteristics of the recorded signal (e.g. the low signal to noise ratio, susceptibility to artifacts, etc.) make the extraction of people's intentions from their brain waves a challenging task. Therefore, the accuracy of BCI systems has not yet reached the high level required for their use in day-to-day human life applications. This is especially true for self-paced BCIs, which are the more natural way to operate a BCI.

The main aim of this thesis is to design a sensory-motor based self-paced BCI. BCIs that depend on sensory-motor rhythms are convenient because no stimulus is required in the BCI control and there is no need for the constant involvement of the sensory modalities such as vision or auditory systems. Because of the challenges in designing a self-paced BCI, it is imperative to exploit the properties of the BCI data as much as possible. As discussed in section 1.4, while the subject is performing the sensory motor mental task, the variations over time in the EEG signal in specific frequency bands should be taken into consideration. These frequency bands are the sensory motor (i.e. alpha, beta and gamma) frequencies of the brain.

When a person performs a motor activity such as a limb movement, the amplitude of the sensory motor rhythms in the motor cortex of the brain changes. Generally such changes may result in a decrease or increase in the amplitude of the EEG signal in the sensory motor frequency bands of the brain [77]. The decrease in the amplitude is called Event Related Desynchronization (ERD), and the increase is referred to as Event Related Synchronization (ERS).

For the mu rhythm, ERD begins shortly before movement onset and continues for few seconds. For the beta rhythm, after a short ERD, the ERS which is the increase in the amplitude of the EEG signal occurs and lasts for a few seconds after the occurrence of the movement. In addition to alpha and beta rhythms, the gamma rhythm also exhibits amplitude changes during the onset of the movement.

The other important property is that the brain characteristics of each person is different from that of others. These brain characteristics are fed to the BCI system in the form of some parameters referred to as hyper-parameters. For instance, the frequency ranges of the alpha and beta waves of the brain are some of the hyper-parameters of the system. The optimal frequency bands of these waves that discriminate between different sensory motor mental tasks vary between different subjects. The value of these frequencies should be identified for each individual person.

Briefly, in this thesis, due to the lack of suitable benchmarks we first design and develop a general BCI comparison framework (aim 1). Using this framework we then develop a user-customized synchronous BCI to show the importance and feasibility of customizing BCI based on each subject's brain characteristics (aim 2). Then we propose new classifiers that are able to capture the variations over time of the EEG signal in self-paced BCIs (aim 3). Finally, based on the frameworks and findings from aims 1-3, we design and develop a user-customized self-paced BCI (aim of this thesis) that exploits the time and frequency features of the EEG signal.

In the following we briefly describe the aims and contributions in this thesis:

Aim 1) Design and Implementation of a General BCI Comparison Framework.

Due to the large variations in BCI technologies and the lack of a general comparison framework in BCIs, it is almost infeasible to compare existing BCI technologies. Consequently, the progress in the field has been slow, from the signal processing and machine learning point of view. There are some general purpose software systems [87] [85] that facilitate the research in BCIs. However, these systems are mostly suitable for data acquisition and for real time processing of brain signals.

In the first part of this thesis (Chapter 2), we design and develop a general framework to compare the performance of different algorithms in sensory-motor-based BCIs. The number of subjects considered in our experiments is 29 which is by far the largest study of its kind in this field. Other studies have been performed on smaller sample sizes and as a result, their findings may not be generalized for a larger subject pool.

We perform rigorous statistical tests to compare the performance of different algorithms. This enables us to examine whether or not there is significant differences between the performances of different classifiers. Other studies have not employed rigorous statistical tests for comparisons. This is mainly due to the smaller sample sizes they employed. They have relied on statistical tests that are not suitable for comparing several algorithms on multiple datasets.

We believe that our work provides a more comprehensive comparison between well-known BCI settings and will have profound impact in the progress of the BCI field.

Aim 2) Design and Implementation of a User-Customized BCI for Synchronous BCIs.

As discussed in section 1.4, one of the characteristics of the sensory-motor BCI data is that the changes in EEG signal are frequency-specific. ERD and ERS occur in specific frequency bands which are different for each subject. The majority of approaches have treated the problem of finding optimal values for these frequency bands (and other hyper-parameters of the system) separately from the methods used in the feature extraction and classification blocks. These approaches usually need considerable knowledge and expertise in EEG signal, and do not take into consideration the interconnection between preprocessing, feature extraction and the classification parts of a BCI.

An optimal learning framework for a BCI problem should aim at studying preprocessing of the data, feature extraction and classification jointly, considering the fact that the performance of the classifier depends on the choice of the features and the frequency ranges that are used to filter the raw EEG signal. In other words, we should customize the classifier and the feature extractor jointly i.e. based on each other. For instance, selecting a particular feature extraction method might cause the data samples of different brain states to be linearly separable in the feature space. As a result, selecting a linear classifier would be a better choice for discriminating between the brain states.

Another problem in the BCI literature is that the majority of the approaches in the field, have treated frequency, time, and the spatial dimensions of the BCI separately. The choice of the frequency ranges are important in the accuracy of the BCI, but so is the choice of the channels used for feature extraction. Usually the time segment from which features are extracted is also important in order to achieve maximum discrimination. The BCI system should exploit the interdependency between these dimensions to yield better performance.

In Chapter 3, we propose an automatic algorithm that jointly optimizes the values of the BCI hyper-parameters for two different types of synchronous motor-imagery-based BCIs. Our algorithm does not rely on the operator expertise and knowledge of EEG to obtain the values of the hyper-parameters. It iteratively tunes the hyper-parameter values based on the performance of the BCI.

Figure 1.3 shows a simple block diagram of our algorithm. In our approach the BCI is like a self-regulating system which improves itself based on the feedback that it receives from the classification block. The performance of the classification block is given to the hyper-parameter optimization block, and this block proposes new values for the hyper-parameters for each iteration of the algorithm. The hyper-parameter values are used to extract features from the EEG signal, and a new classifier is trained using the features extracted from the training data. Based on the cross-validation performance of the classifier, the hyper-parameter optimization block proposes another set of values, and the whole process continues until the algorithm converges.

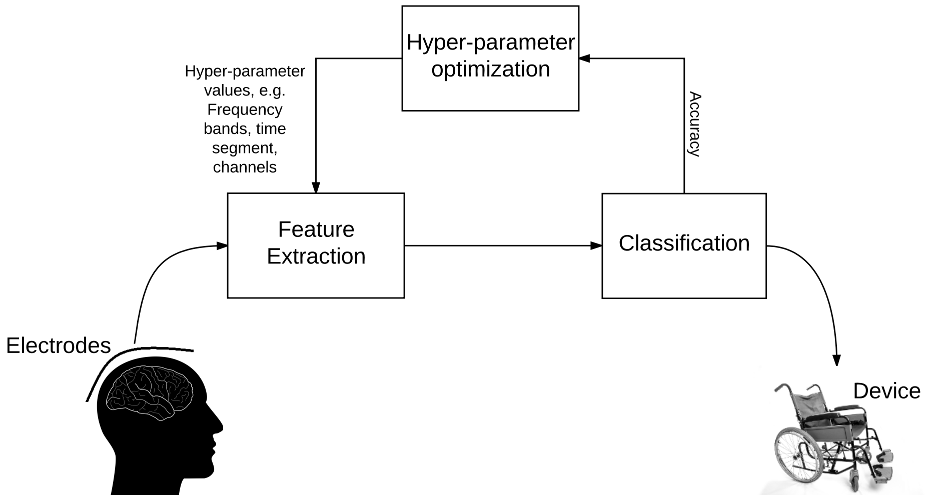
Hyper-parameter optimization
Hyper-parameter values, e.g. Frequency bands, time segment, channels
Accuracy
Feature Extraction
Classification
Electrodes
Device

Figure 1.3: Our proposed hyper-parameter optimization in BCIs.

We show the importance of hyper-parameter optimization using 21 subjects from public BCI competition datasets. We then compare our algorithm to other approaches used for finding the hyper-parameter values, and finally we compare our method to the best reported results in the literature.

Aim 3) Using Time Information in Designing a Self-Paced BCI.

Another characteristic of the BCI data is that the changes in EEG signal are time-specific. As discussed in the previous sections, the changes in the EEG signal while a user is performing the mental task are variable over time. For example, ERD and ERS occur during the movement onset and continue for few seconds after the occurrence of the movement. Exploiting this time related information is crucial in improving the performance of the BCI, Specially in self-paced BCIs where the start and the end time of the mental task is not known, and the BCI should continuously analyze and classify the EEG data.

To exploit the time information, three possible approaches can be considered:

1) Extract features from different parts (i.e. time windows) of the EEG signal, concatenate the features and give the feature vector to the classifier. This is the most popular approach in the literature [21] [43] [88] [99] [96], however this approach does not capture the temporal ordering of different features.

2) Another approach is to train different classifiers on different parts of the EEG signal, and then aggregate the results. The problem of this approach is that the correlation between different parts of the signal is not taken into account.

3) None of the above approaches properly capture the dynamics of the EEG signal. A better approach is to apply sequence labeling classifiers on the EEG data. Sequence labeling classifiers predict a sequence of labels associated with a sequence of observations. In this approach the EEG signal is divided into several consecutive segments which would result in a sequence of observations and labels. The set of these observation/label sequences builds the training set which forms the input to the classifier.

In chapter 4, we propose a sequence labeling algorithm which is a combination of neural networks and Conditional Random Fields. Our algorithm combines the power of neural networks (to extract features from the EEG signal) with the conditional random field classifier (to capture the dynamics of the EEG signal). We performed our experiments on 13 (self-paced) BCI subjects and compared the performance of our algorithm to several sequence labeling and classical classifiers. All classifiers were evaluated in an online setting, i.e. classifiers continuously classified the test data similar to real applications.

Aim 4) Design and Implementation of a User-Customized BCI for Self-Paced BCIs.

Changes in the EEG signal while a subject is performing a mental task are time and frequency-specific. In Chapter 5, we design a self-paced BCI which not only takes into account the time information for self-paced BCIs but also is customized based on the characteristics of each subject's EEG signal.

Our BCI exploits the time information by using a sequence labeling classifier. We use Bayesian optimization to tune subject specific frequencies, channels and other parameters of the self-paced BCI system. Our algorithm is a combination of the methods proposed in Chapters 3 and 4. While in Chapter 3, many different methods of customizing a synchronous BCI are studied, in chapter 5, we use an ensemble approach which led to more stable and better average performance across all subjects.

In Chapter 5, we describe our algorithm and compare our method to several well-known classifiers. We conduct our experiments using 13 subjects in self-paced BCIs. In our experiments, the aim is to discriminate the control from the no-control states of the brain. All classifiers are evaluated in a setting similar to online BCIs, i.e. classifiers continuously classify the test data by only looking at the past data. We show that tuning the hyper-parameters of a self-paced BCI system and at the same time exploiting the dynamics of the EEG signal, can considerably improve the performance of self-paced BCIs.

Chapter 2

A Unified Comparison Framework for Sensory-Motor BCIs

2.1 Introduction

Due to wide variations in BCI technologies and the lack of a general comparison framework in BCIs, it is almost infeasible to compare existing BCI technologies and speed up the progress in the field. As the first part of my thesis, I have developed a general framework to compare the performance of different algorithms in EEG-based BCIs. As this research focuses mostly on the classification part of a BCI system, the unified comparison framework that we developed was used to evaluate the performance of different classifiers on several sensory motor BCI datasets. The data used are those from BCI competitions[1], as well as one of our own datasets [20].

There have been some efforts to compare the performance of different classifiers in BCI systems [3] [63] [6] [91] [19]. However, these comparisons have been performed on a small number of subjects and for a small number of classification methods. In [68], the authors

[1]http://www.bbci.de/competition

surveyed the performance of different classifiers in BCIs that used different subjects and different features. However, the best way to compare different classifiers is to evaluate their performance in the same context, i.e., on the same set of subjects and using the same set of features with the same set of parameters.

The general trend in EEG-based BCI systems has been to utilize the Linear Discriminant Analysis (LDA) classifier for classification purposes [8] [53]. The binary LDA classifier has a very strong assumption that the conditional probability densities of the two classes have a Gaussian distribution with the same covariance function. As a result, the discriminant function is linear and may thus not be suitable for non-linear separable feature spaces. In addition, this classifier is very sensitive to outliers. On the other hand, as discussed in section 1.4 the main challenges in classification of EEG data in BCIs is that the data are non-stationary, noisy, contain outliers, are usually of high dimensions and the training set is unfortunately small [67] [103]. As a result, the classification strategies in BCI systems should be able to cope with these problems appropriately.

This part of our study provides open source comparison framework for BCI systems and is unique in three aspects. First, the number of subjects considered in this chapter are 21 subjects that used synchronous BCIs and 8 subjects that used self-paced BCIs (i.e. an overall of 29 subjects). Other studies were performed on smaller sample sizes (less than 5 subjects) and as a result, their findings may not be generalized to a larger subject pool. Secondly, we have used rigorous statistical tests to compare the performance of different algorithms. This will enable us to examine whether or not there are significant differences between the performance of different classifiers. Other studies have not employed statistical tests for comparisons, mainly because of their smaller sample sizes and have solely relied on the mean performance of the classifiers (which is not a robust measure), as a surrogate for the overall performance of a specific classifier. Thirdly, to increase the transparency, the source code of our experiments has been made openly available so that other researchers can apply it on their own data and benchmark their algorithms with standard methods.

In the following sections we first explain the general structure of the comparison framework, then we explain the datasets used and finally the results of the different algorithms are compared.

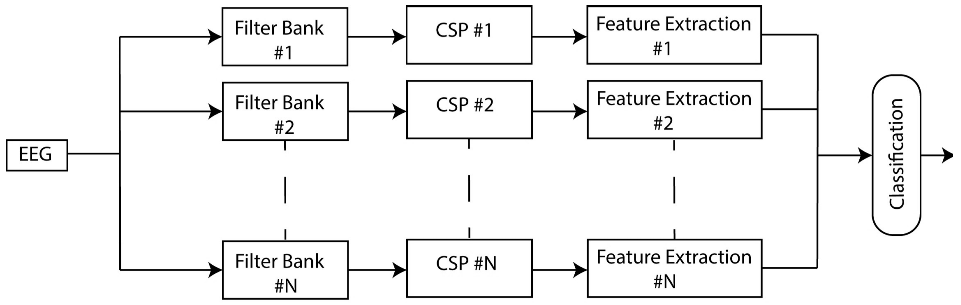

Figure 2.1: The BCI framework. Our framework has three main steps: 1. Filtering, 2. Feature Extraction and 3. Classification. The output of each step is fed to the next step.

2.2 General Structure of Comparison Framework

The overall structure of our comparison framework is shown in Figure 2.1. This framework has three main sequential processing components, Filtering, Feature Extraction and Classification. All data from the available EEG channels are fed as the input, the Filtering component performs frequency filtering and spatial filtering, and the Feature Extraction component extracts features from the filtered data. Finally, in the last step the extracted features are combined to build a dataset which is then fed to the classification component. In the following subsections, the details of each of the above mentioned components are explained.

2.2.1 Spatial and Frequency Filtering

The first step of our framework was composed of frequency filtering and spatial filtering. Frequency filtering was done using a filter bank. A filter bank is an array of band pass filters and contains n blocks. Each block corresponds to filtering the original signal in a specific sub-band. For frequency filtering, we used the fifth order Butterworth filter. This filter has a flat frequency response in the pass band. The result of applying each filter-bank block was fed to the next block in which spatial filtering is performed on the signal.

Common Spatial Patterns (CSP) [82] [18] was used for spatial filtering. CSP has been widely used in BCI systems and has yielded considerable improvements in the signal to noise ratio of the EEG signal. CSP projects multiple channels of the EEG data onto a surrogate sensor space by applying a linear transformation on the EEG data. This technique is a supervised method of combining multiple channels and has been developed for binary classification. This method maximizes the variance of signals for one class while minimizing the variance of the signals for the other. As the signals are standardized before applying CSP, the variance of the signals would be equivalent to their band power. Thereby, if we use the band power as the extracted feature in the surrogate space, the discrimination of the classes (e.g., right and left hand imagery movements) would be maximized.

Suppose that the normalized covariance matrices of both classes are given by Σ^c where $c \in \{+,-\}$, and $+$ and $-$ correspond to different mental tasks in BCI. The CSP algorithm maximizes the following equation:

$$\underset{W}{\operatorname{argmax}} \frac{W^T \Sigma^+ W}{W^T \Sigma^- W} \tag{2.1}$$

where W is the projection matrix. This equation can be solved by applying simultaneous diagonalization of the covariance matrices of both classes. By multiplying the projection matrix by the original EEG signal we can obtain the uncorrelated brain signals.

The benefit of applying CSP is that we can select a subset of filters that preserves as much information as possible and discriminates the two classes very well. However, choosing the number of filters (i.e., spatial patterns) is difficult, and is usually determined by heuristic approaches. CSP is inherently designed for 2-class BCI tasks, and we have used a one-against-one scheme to use CSP for the multi-class dataset. In a one-against-one scheme a separate CSP filter is used to discriminate each pair of mental tasks.

2.2.2 Feature Extraction

In the feature extraction step, different kinds of features can be extracted from the filtered signal. The extracted features should properly represent the information hidden in the raw EEG signal. We used two of the most common and successful feature extraction methods utilized in motor-imagery based BCI systems. Both methods calculate the band-power (BP) of the signal [25] [24] [47].

In the first approach, the band-power of a signal is extracted by directly applying to a signal, a filter that only allows the frequencies inside each filter band to pass. Assuming a perfect block filter, we can then estimate the power of the filtered signal by summing the squares of the magnitude of the filtered signal. Since the logarithm of band-power features has also been used in the literature [25], we have included both the band-power and its logarithm in our feature set. In this study, we extracted the band power features for the α [8-12]Hz and β [16-24]Hz frequency bands of the brain signals. We refer to these band-power features as the BP features in the remainder of the text.

In the second feature extraction approach, we used the Morlet wavelet to extract the band-power features. This method is one of most the successful feature extraction methods used in sensory motor BCI systems [25]. In this method, the EEG signal is decomposed using the Morlet mother wavelet and the power in each frequency band is calculated. We used the exact configuration as in [24] to extract these features; i.e., for each channel, all the frequency bands are chosen to be in the 4 to 30 Hz range. This results in 26 features for each channel. This setting for extracting BP features will result in high-dimensional feature spaces, especially for datasets with many EEG channels.

After extracting features from each block of the filter bank, the extracted features are combined to build a feature vector, which is fed to the classification component of our framework.

2.2.3 Classification

In supervised learning, given a set of training samples $D = \{(f_1, y_1), (f_2, y_2), , (f_n, y_n)\}$, the aim is to find an approximation of the unknown function $g : F \rightarrow Y$ which has generated the samples. Each f_i is a feature vector and y_i is the label of the corresponding feature vector. F is the feature matrix and Y is a vector corresponding to the label of each row (i.e. f_i) of X. In probabilistic classification, instead of approximating the function g, a posterior probability $P(c|f)$ (where c is the predicted class labels vector and f is the feature matrix) is found. Eventually the label is assigned to the class with the highest probability. This probability can be calculated using the Bayes rule. $P(c|f)$ is called the posterior, and $P(f|c)$ is the class conditional density.

$$P(c|f) = \frac{P(f|c)P(c)}{P(f)} \tag{2.2}$$

In the following, we briefly describe the classifiers we have compared using our framework.

Gaussian Discriminant Analysis

When the class conditional density is assumed to have a Gaussian distribution, then the resulting classifier is referred to as the Gaussian Discriminant Analysis [73]. This model fits a Gaussian distribution to the samples of each class. If the covariance matrices of both classes are considered to be the same, the result will be a Linear Discriminant Analysis (LDA) classifier in which the decision boundary is a linear surface. If we do not assume any constraints on the covariance function, the resulting decision boundary is a quadratic function and the corresponding classifier is called Quadratic Discriminant Analysis.

These classifiers impose very strong assumption on the underlying distribution of the data. This has the advantage that the computation of the discriminative function is very efficient. Therefore, the LDA classifier has been popular in the BCI field. It is important to mention that various algorithms [105] [42] [65] have been proposed to eliminate the shortcomings of LDA. These algorithms are more robust and some of them, such as Z-LDA, [105] can handle the case where the covariance of two classes are different.

Logistic Regression (LR)

Unlike Linear Discriminant Analysis (LDA) which makes strong assumptions about the underlying distribution of the data, logistic regression does not make any assumptions and has been preferred over LDA in machine learning applications [80].

This classifier is a discriminative learning classifier that directly estimates the parameters of the distribution $p(c|f)$ where c is the class label and f is the feature vector. This algorithm assumes the distribution $p(c|f)$ is modeled using a softmax function:

$$p(c|f) = \frac{exp(W_k^T f)}{\Sigma_j exp(W_j^T f)} \tag{2.3}$$

where W_js are the parameters to estimate. Then the maximum likelihood approach is used to directly approximate $W_j s$. As the Hessian matrix for the logistic regression model is positive definite, the error function has a unique minimum. Over-fitting can occur in logistic regression when the data is sparse and high dimensional (which is the case in BCIs). We used l_1 and l_2 regularization jointly to cope with the over-fitting problem.

Random Forests (RF)

The Random Forest classifier is an ensemble learning algorithm that is constructed by combining multiple decision trees at the training stage and produces a result that is the average of the output of all individual trees [29]. This powerful learning algorithm injects randomness into each tree in two ways. The first uses bootstrapping to sample from the original dataset (i.e. the algorithm takes M samples with replacement from the original dataset). The second selects a subset of the features to split each node of the tree. Injecting randomness in the process of building Random Forests, makes these classifiers robust and cause them to have a good performance when the data have many outliers, which is the case in BCIs [33]. Another consequence of injecting randomness in random forests is the ability to rank the different features and also to acquire a measure for feature importance.

The seminal paper of Random Forests [22] claims that increasing the number of trees does not cause the random forest to over-fit. However, [90] has found that RFs can over-fit with noisy datasets. As a consequence, along with other important parameters of RF we have tuned the number of trees (2.2.5).

Support Vector Machines (SVM)

An SVM [51] is a discriminative classification algorithm which is able to find a decision hyper-plane with the maximum distance (margin) to the nearest data points (Support Vectors) of each class. As a result, this method has a high generalization power. The decision function of an SVM is fully specified by a subset of the training data points, which leads to a sparse solution for SVM. The cost function of an SVM is a convex function that leads to an optimal solution for the optimization task.

The mathematical formulation of an SVM gives us the ability to use what is known as the kernel trick, to map the original finite dimensional space into a destination space with much higher dimensions. The use of the kernel trick is beneficial particularly when the data points cannot be separated with a hyper-plane in the original feature space. This may lead to an easier separation of the data points in the destination space. In this study, we have only used SVMs with a radial basis function (RBF) kernel. As stated in [55], under some conditions, linear kernel SVM can be approximated with an RBF kernel. Even though the data points might be mapped into a very high dimensional or even an infinite space, the complexity of computing a kernel matrix can be far smaller. Hence, the SVM algorithm circumvents the curse of dimensionality by relying on the data points only through the kernel. SVMs are inherently designed for two-class classification tasks. Many different methods have been proposed to adapt SVMs to multi-class problems [52]. We used a one-against-one scheme to adapt SVMs for multi-class problems. Given a multi-class problem with M classes, in a one-against-one scheme, we train $\binom{M}{2}$ classifiers to discriminate each pair of classes. At the test time, a voting scheme is used to predict the label of a new unseen sample.

Boosting Algorithm

A Boosting classifier [41] is a learning algorithm based on the observation that building a highly accurate rule for classification is a really difficult task. This algorithm works by building many rough rules of thumb. These rules are called weak classifiers. Each of these weak classifiers is trained on a subset of the data points. These weak classifiers are then combined into a single prediction rule which would be much more accurate than individual weak classifiers.

There are many variants of the boosting algorithm. The differences amongst these algorithms stem from two issues. The first is concerned with how to select the subset of the data points for each weak learning algorithm. The second is related to how to combine the output of a weak classifier into a single prediction rule. In this study, we used a variant of boosting called the Adaboost algorithm. In this algorithm, the subset of data points that are used for training a weak learning are the ones that are misclassified by the previous weak learner. For generating a single prediction rule, Adaboost takes a weighted majority vote of the prediction of the weak learners.

Multi Layer Perceptron (MLP)

The last classifier used in this study is a feed forward neural network [14] with one hidden layer. It has been shown that an MLP with enough number of neurons in the hidden layer can approximate a wide variety of functions[50]. Despite the flexibility and capability of this algorithm to approximate any nonlinear function, this algorithm can easily overfit and the cost function to optimize is a non-convex function. The cost function we used is the negative log-likelihood function with both l_1 and l_2 regularization to avoid overfitting.

2.2.4 Performance Measure

For the synchronous BCI datasets, we first obtained the labels of the different classes of motor imagery tasks, then the classification accuracy was used as the measure to compare the performance of different algorithms.

For self-paced datasets, the true positive rate (TPR) and the false positive rate (FPR) are the most popular measures in the field. It is common to fix the FPR to a small value (e.g. 1 percent) and compare the performance of TPRs of different methods. However, this method does not exploit all the information given by the output of the classifiers because all the classifiers applied in this research produce probability estimates (i.e. confidence) of a sample belonging to each class. Since, the output of a classifier is not just the label of each sample, a good performance measure should exploit the added information. The receiver operating characteristic (ROC) [39] curve is the best way to compare these classifiers as it illustrates the performance of the classifiers for varied discrimination thresholds. ROC is well-suited for self-paced BCIs because ROC is suitable for evaluating the performance when we have imbalanced data or unequal misclassification costs. To compare the performances of the methods, we use the Area Under the ROC Curve (AUC). AUC represents the probability that a randomly selected positive sample gets a higher rank than a randomly chosen negative sample. AUC varies between 0 and 1 and in general a higher AUC is better.

2.2.5 Model Selection

Model selection consists of finding the appropriate set of parameters for each learning algorithm. We adjusted the value of a set of parameters (referred to as ”BCI parameters”) consisting of parameters of the sensory motor EEG signal and the classifier-specific parameters using a grid-based approach.

For each individual subject, a set of candidate parameters including both the BCI parameters and the classifier parameters were tuned for each algorithm, and 5 times 5-fold cross validation was performed to find the best value of the parameters. Then the parameter values with the best mean performance were used to train a final classifier on the training set for that subject. The final classifier was used to evaluate the performance on independent test dataset for the same subject. The values of all these parameters (i.e. the BCI parameters and the classifier parameters) were adjusted jointly.

The BCI parameters used in the synchronous BCI dataset consisted of selecting the time segment from which the features are extracted and the number of CSP filters. To adjust the values of these parameters, the system should be customized based on the brain characteristics of each individual subject. To get an acceptable performance with the synchronous datasets, the usual practice is to discard some parts of the movement period in each trial. The choice of the time segment from which features are extracted, has a major role in the performance of the learning system. In datasets with many channels we have also used CSP to combine different channels. As a result, the number of CSP filters was included as a parameter in model selection.

In self-paced BCIs, the classification problem is a sequential learning problem. To be able to apply static classifiers to this problem, we used sliding windows with overlap. The size of the window, the overlap of two consecutive windows, and the number of CSP filters formed the set of BCI parameters in the self-paced datasets.

The set of classifier-specific parameters that we adjusted for each classifier are given in Table 2.1.

2.2.6 Statistical Tests

Instead of using empirical approaches that have been commonly used in the BCI field to compare the performance of different algorithms, the Friedman statistical test is used in this study. The Friedman test [32] [44] is a non-parametric statistical test, which ranks different classifiers for each subject separately. Then it averages the ranks over all subjects. The null hypothesis assumes that all algorithms have the same performance so they have the same rank. In other words, it assumes the difference between different algorithms is random. The test statistic has an approximate Chi-square distribution, and if the p-value is low enough to reject the null hypothesis, we can conclude that the difference between the algorithms is not random.

Table 2.1: List of classifier parameters that were optimized using cross-validation in the training phase.

Random Forest	Number of trees
	Maximum number features evaluated to split each node
	Maximum depth of each tree
	Minimum number of samples in each leaf
SVM	C
	Gamma
GLM	Regularization Type
	Regularizer Coefficient
Boosting	Number of trees
	Maximum number features evaluated to split each node
	Maximum depth of each tree
	Learning rate
MLP	Number of neurons in hidden layer
	L1 coefficient
	L2 coefficient
	Learning rate
GDA	None

If the null hypothesis is rejected, another statistical test is performed to identify which algorithms are the source of difference. We then conduct the Holm's test as the post-hoc statistical test. The classifier with the best rank is selected as the control classifier, then a pairwise comparison of all the other classifiers and the control classifier is performed. In this statistical test, the null hypothesis states that the control classifier and the other classifier have the same mean rank. We have divided the algorithms based on the outcome of the Holm's test into two categories: the recommended and not recommended. The recommended classifiers include the control classifier, i.e., the best one and any other classifier that we are not able to prove it has a worse performance than the best classifier. All the others were deemed to belong to the not recommended category.

Table 2.2: Specification of datasets used in this thesis.

Dataset	Type	Number of Subjects	Task	Number of Channels
BCICIII3b	Synchronous	3	left hand vs. right hand	2
BCICIV2b	Synchronous	9	left hand vs. right hand	3
BCICIV2a	Synchronous	9	left hand vs. right hand vs. both feet vs. tongue	22
BCICIV1	self-paced	4	left hand, right hand and foot vs. No-Control	59
SM2	self-paced	4	right index finger vs. No-Control	10

2.3 Datasets

Five sensory motor BCI datasets consisting of 29 subjects were used to evaluate different methodologies studied in this chapter. These are the datasets I [15], IIa [26] and IIb [64] of the BCI competition IV, and dataset IIIa [16] from BCI competition III and SM2 from [20].

Table 2.2 shows the specifications of each dataset. While SM2 and BCICIV1 datasets were used to evaluate different BCI designs in the self-paced paradigm, the remaining datasets were used for synchronous BCI systems evaluation. For each subject in each dataset we have two sets of data: calibration and evaluation data. The calibration data is used to train (calibrate) the BCI system. The evaluation data is used for the evaluation of the BCI system.

Dataset I from competition IV (BCICIV1) was recorded from four subjects performing motor imagery tasks (left hand, right hand or foot imagery). Each subject participated in two sessions of brain signal recording. The first session was used for training the BCI system, and the second session of signal recording was used for the evaluation of the BCI system. This dataset consisted of 59 EEG channels (corresponding to 59 sensors). In the calibration session, each subject was assigned to perform two of the three classes of motor imagery tasks: left hand, right hand, or foot imagery movements. There were 200 trials of imagery movements that were balanced between two classes. Each trial was 8 seconds long, the length of the motor imagery intervals in the evaluation session varied from 1.5 to 8s. The NC intervals were also between 1.5 and 8s. Getting a very good performance on this 3-class self-paced dataset is very challenging, therefore we converted the problem into a binary classification in which the aim is to decide whether an output is a control state or a No-Control state.

The SM2 dataset was collected from four subjects attempting to activate a switch. At random intervals (mean = 7 seconds), a cue was displayed for the subjects. The subjects attempted to activate the switch by moving their right index finger after the cue appeared. In this dataset, the EEG was recorded from 10 channels positioned over the supplementary motor area and the primary motor cortex (i.e. FC1-4, FCz, C1-4, Cz). For each subject, an average of 10 sessions of EEG recording was performed for six days. In each session, the period between any two trials varied and the subjects performed actual movement. For each subject a fixed set of trials (mean = 500 trials) was used for training the BCI and the rest for evaluating the BCI. A detailed description of this dataset can be found in [20].

Dataset IIa from the BCI competition IV (BCICIV2a) was recorded for nine subjects performing 4-class motor imagery (left hand and right hand, both feet and tongue imagery movements) tasks. The data consisted of 19 channels along the scalp. The data was recorded in two sessions and each session comprised of 288 trials. The first session was used to train (calibrate) the BCI and the second session was used to evaluate the BCI system.

Dataset IIb from BCI competition IV (BCICIV2b) was recorded for nine subjects performing 2-class motor imagery (left hand and right hand) tasks. The data consisted of three channels (i.e. C3, CZ and C4) along the motor cortex of the scalp. For each subject, five sessions were recorded. The first three sessions were used for training the BCI system, and the remaining two sessions were used for testing the system.

Dataset IIIb from BCI competition III (BCICIII3b) was recorded from three subjects performing 2-class motor imagery (left hand and right hand) tasks. The data had two channels (i.e. C3, C4) from the motor cortex area of the brain. For each subject, three sessions of EEG recordings was provided.

2.4 Results and Discussion

We applied different combinations of feature extraction, classification and model selection methods to five datasets. Tables 2.3 and 2.4 show the performance for synchronous and self-paced datasets, respectively, with the best performing feature extraction/classification combinations highlighted in blue. Qualitative comparison of the different feature extraction/classification combinations suggests the following: 1) for synchronous BCI systems, logistic regression classification outperforms the other classifiers (Table 2.3). This is regardless of the feature extraction methodology used. 2) For self-paced BCIs, (Table 2.4), both logistic regression and MLP classifiers yield better performances. In addition, in both self-paced and synchronous BCIs, Tables 1 and 2 show that for the subjects with the higher number of EEG channels, the Band-Power (BP) outperforms the Morlet features mainly due to the application of CSP on the channels. For datasets having a lower number of EEG channels, the use of Morlet features outperforms that of BP features.

The Friedman statistical test was performed to compare the performance of the different classification methods. This was done for both the BP and Morlet feature extraction methods for both types of BCIs (synchronous and self-paced), resulting in four different comparison settings. In Settings 1 and 2, the BP and Morlet features with different classifiers were applied on the synchronous datasets, respectively, and in settings 3 and 4, the BP and Morlet features with different classifiers were applied on the self-paced datasets, respectively.

Table 2.3: The accuracy of classifiers for synchronous BCI operation over all subjects. The accuracies are calculated using the unseen test dataset. For each classification algorithm the first column shows the results of BP features and the second column shows the results of Morlet features.

Classifier / **Subjects**	**Boosting**		**Logistic**		**Random Forest**		**SVM**		**LDA**		**QDA**		**MLP**	
	BP	Morlet	BP	Morlet	BP	Morlet	BP	Morlet	BP	Morlet	BP	Morlet	BP	Morlet
Subject1 (O3)	75.47	80.5	80.5	82.39	74.84	79.25	81.76	77.36	81.13	74.21	79.25	60.38	78.62	83.65
Subject2 (S4)	71.11	77.96	70.19	83.89	68.52	79.26	71.11	83.52	70	81.11	70.37	72.41	70.37	82.22
Subject3 (X11)	72.22	78.15	74.26	78.15	71.48	77.78	72.78	77.78	74.07	76.48	74.81	72.22	74.81	76.67
Subject4 (100)	66.23	61.84	60.96	68.86	63.6	65.35	64.04	64.91	61.4	75.88	63.16	58.77	63.6	69.3
Subject5 (200)	53.47	51.84	56.33	58.37	54.29	54.29	56.73	59.18	56.33	55.92	54.69	55.51	56.33	58.37
Subject6 (300)	56.52	54.35	56.09	53.48	51.74	51.74	51.74	45.22	56.09	49.13	54.35	46.52	57.83	51.74
Subject7 (400)	89.25	90.88	94.79	94.79	91.21	92.83	95.11	94.14	93.81	94.46	95.77	83.39	92.18	94.46
Subject8 (500)	61.9	86.45	67.77	91.58	63.37	85.71	68.13	85.71	65.57	87.18	67.03	76.56	67.4	85.71
Subject9 (600)	74.1	79.68	75.7	82.87	74.1	80.48	65.74	80.88	76.1	82.07	75.3	60.96	76.89	83.27
Subject10 (700)	54.31	70.69	53.02	72.84	49.14	76.29	52.59	70.69	59.05	71.55	57.33	64.66	51.29	71.12
Subject11 (800)	91.74	83.91	92.17	83.48	90	86.52	92.17	80.87	92.17	80.87	86.52	67.83	90.87	80
Subject12 (900)	78.37	82.45	77.55	86.12	75.92	82.86	76.73	86.12	77.96	76.33	77.96	68.57	77.14	83.67
Subject13 (1)	79	71.53	79	60.85	81.85	73.31	81.14	61.57	74.38	50.89	52.67	26.33	81.85	58.01
Subject14 (2)	51.59	53.36	61.13	54.42	53.36	51.24	57.24	57.6	62.9	32.86	38.87	25.8	58.3	54.77
Subject15 (3)	78.75	83.15	86.45	86.81	78.02	82.78	84.25	78.02	80.22	50.55	41.03	27.47	85.35	83.15
Subject16 (4)	71.49	32.02	73.68	41.23	73.68	45.18	71.05	36.84	60.96	35.53	36.84	30.26	72.81	34.65
Subject17 (5)	56.16	33.33	60.14	40.58	58.33	35.14	56.88	41.67	50	28.26	34.06	28.26	60.14	38.77
Subject18 (6)	51.63	24.65	56.74	26.98	52.09	26.05	57.21	25.58	54.42	27.91	33.95	26.98	59.07	26.05
Subject19 (7)	84.48	64.98	87.36	56.32	81.95	71.48	87.36	64.26	70.4	32.49	25.27	31.41	87	60.29
Subject20 (8)	77.12	54.98	80.81	61.62	79.34	63.47	81.55	62.36	75.65	37.27	46.49	31	80.44	61.62
Subject21 (9)	78.03	46.97	83.71	39.39	82.95	56.44	84.09	45.08	73.86	41.29	34.47	25	84.85	44.32

The reason we compared the performance of different classifiers when a single feature extraction methodology was used was that the nature of feature spaces was different for each setting and the results could thus be misleading. For example, when we use the BP features, the feature space is of low dimensions compared to the settings where the Morlet feature is used. Therefore, different classification methodologies could only be compared when they are applied on the same feature space. Since the classification problem in synchronous and self-paced BCIs was different, it only make sense to compare the classifiers on self-paced and synchronous datasets separately.

Table 2.4: The Area Under the Curve (AUC) of classifiers for self-paced subjects. For each classification algorithm the first column shows the results of BP features and the second column shows the results of morlet features.

Classifier / **Subjects**	**Boosting**		**Logistic**		**Random Forest**		**SVM**		**LDA**		**QDA**		**MLP**	
	BP	Morlet	BP	Morlet	BP	Morlet	BP	Morlet	BP	Morlet	BP	Morlet	BP	Morlet
Subject22 (22)	0.46	0.56	0.64	0.58	0.49	0.49	0.41	0.48	0.64	0.57	0.62	0.55	0.63	0.56
Subject23 (23)	0.54	0.61	0.67	0.68	0.53	0.55	0.5	0.59	0.64	0.68	0.63	0.53	0.66	0.7
Subject24 (24)	0.6	0.58	0.66	0.58	0.58	0.51	0.6	0.52	0.65	0.58	0.64	0.54	0.63	0.57
Subject25 (25)	0.36	0.31	0.78	0.77	0.6	0.69	0.47	0.6	0.79	0.77	0.68	0.66	0.79	0.73
Subject26 (a)	0.59	0.53	0.66	0.55	0.64	0.53	0.56	0.53	0.65	0.53	0.61	0.5	0.66	0.57
Subject27 (b)	0.79	0.77	0.82	0.83	0.78	0.81	0.72	0.76	0.66	0.8	0.72	0.73	0.82	0.83
Subject28 (f)	0.49	0.54	0.51	0.53	0.51	0.53	0.49	0.51	0.5	0.53	0.48	0.51	0.49	0.52
Subject29 (g)	0.52	0.5	0.53	0.58	0.52	0.51	0.53	0.51	0.53	0.55	0.58	0.52	0.53	0.58

The comparison between different classifiers was performed as follows: (a) we ranked each classification algorithm based on its performance on the test data. For example, for subject1 (O3), when the Morlet features were used, the MLP was the best classifier, LR was the second best and so on. We ranked the algorithms over each subject separately, then we averaged the ranks, (b) the Friedman test was then applied on the resulting ranked results. (c) if the null hypothesis was rejected by the Friedman statistical test, we performed a second set of statistical tests. In particular, we used the Holm's test and compared all classifiers to the control classifier. The control classifier is the one with the best average rank on the test data.

The average rank of the different classifiers that are based on the number of wins for the synchronous and the self-paced datasets respectively are given in Table 2.5. These results show that depending on the feature sets and the dataset type (synchronous versus self-paced), both linear classifiers such as LR and LDA and non-linear ones (e.g., MLP, RF) could be among the top performing classifiers. Another interesting observation is that in all settings, the average rank of the LR classifier is better than the LDA's. This is due the fact that the BCI data is noisy and usually has many outliers. As discussed in the previous sections, unlike LR, LDA is sensitive to outliers and is therefore not robust. Comparing the ensemble classifiers shows that the RF classifier has a better average rank compared to Ada-Boosting (i.e. BST); this is again due the sensitivity of BST to outliers.

Table 2.5: Average Rankings of the classification algorithms for both synchronous and self-paced datasets. The number in the parenthesis corresponds to the average rank of the algorithm among different subjects. For each feature extraction method the classifiers typed in bold are the recommended ones. The recommended classifiers are selected based on the results of the statistical tests.

Data	**Synchronous**		**Self-paced**	
Feature	**BP** (Setting 1)	**Morlet** (Setting 2)	**BP** (Setting 3)	**Morlet** (Setting 4)
1	**MLP**(2.92)	**LR**(2.45)	**LR**(1.81)	**LR**(1.87)
2	**LR**(2.97)	**RF**(3.3)	**MLP**(2.75)	**MLP**(2.56)
3	**SVM**(3.11)	**MLP**(3.42)	**LDA**(3.06)	**LDA**(2.81)
4	**LDA**(4.09)	**SVM**(3.57)	QDA(4.18)	BST(4.25)
5	BST(4.54)	BST(4.16)	RF(4.87)	RF(4.87)
6	QDA(5.11)	LDA(4.45)	SVM(5.5)	SVM(5.81)
7	RF(5.21)	QDA(6.61)	BST(5.81)	QDA(5.81)

In Setting 1, as shown in Table 2.5, MLP was the best performing classifier, i.e., when the BP features were used on synchronous datasets. Since the null hypothesis was rejected by the Friedman statistical test (p-value=1.4e-4), the Holm's post-hoc was performed and all classifiers were compared to Multi-Layer Perceptron (control classifier).

The p-values corresponding to pairwise comparison of classifiers are shown in Table 2.6. For $\alpha = 0.1$, all hypothesis with p-value less than 0.0333 were rejected. According to the Holm's test results, there is no significant difference between the performance of MLP, LR, SVM and LDA. Thereby, these classifiers are the recommended ones for the BP features in synchronous data. The other classifiers, i.e., BST, QDA and RF had a poor performance for this type of features.

Table 2.6: P-values corresponding to pairwise comparison of different classifiers. α is chosen to be 0.1. For settings 1 and 2 all hypothesis with p-value less than 0.0333 are rejected. For setting 3 and 4 all hypothesis with p-value less than 0.05 are rejected. The results are rounded up to 4 decimal places.

Setting 1(Synchrounous, BP)		**Setting 2(Synchrounous, Morlet)**		**Setting 3(Self-paced, BP)**		**Setting 4(Self-paced, Morlet)**	
hypothesis	$P-value$	hypothesis	$P-value$	hypothesis	$P-value$	hypothesis	$P-value$
RF vs. MLP	0.0006	QDA vs. LR	0.0	SVM vs. LR	0.0002	SVM vs. LR	0.0002
QDA vs. MLP	0.0010	LDA vs. LR	0.0026	BST vs. LR	0.0006	QDA vs. LR	0.0002
BST vs. MLP	0.0151	BST vs. LR	0.0101	RF vs. LR	0.0045	RF vs. LR	0.0054
LDA vs. MLP	0.0801	SVM vs. LR	0.0932	QDA vs. LR	0.0278	BST vs. LR	0.0278
SVM vs. MLP	0.7750	MLP vs. LR	0.1431	LDA vs. LR	0.2471	LDA vs. LR	0.3854
LR vs. MLP	0.9430	RF vs. LR	0.1985	MLP vs. LR	0.3854	MLP vs. LR	0.5244

In Setting 2, the best performing classifier was LR (Table 2.5). The p-value for the Friedman test was 1.7e-8; therefore, the difference between the classifiers is not random. The Holm's test suggests that RF, MLP, and SVM are as good as the LR classifier (Table 2.6 setting 2).

In Settings 3 and 4, the LR classifier performed better than others (Table 2.5). In Setting 3 the Friedman's test p-value was 7.16e-4, and in Setting 4 Friedman's test p-value was 1.8e-4. The Holm's test results suggested that there was no significant difference between MLP, LDA and LR (Table 2.6 setting 3 and 4). All hypotheses with p-value less than 0.05 were rejected.

Among the classifiers used in this study, RF, BST and MLP are inherently designed to handle multi-class classification and the others (i.e., SVM, LR, LDA and QDA) are used in a one against others setting to handle multi-task problems. Therefore, in addition to the four settings discussed above, we have also performed two other statistical tests. The aim was to determine which classifiers would yield the best results in binary-task BCIs and which one(s) would yield the best results in multi-task BCIs.

Table 2.7: Average Rankings of the classification algorithms for binary and multi-class classification in synchronous datasets. The number of subjects in binary task was 12 and the number of subjects in multi-task BCIs was 9. The number in the parenthesis corresponds to the average rank of the algorithm among different subjects. For each feature extraction method the classifiers typed in bold are the recommended ones. The recommended classifiers are selected based on the results of the statistical tests.

Feature	**BP**		**Morlet**	
Data	**Binary**	**Multiclass**	**Binary**	**Multiclass**
1	**SVM**(3.33)	**MLP**(2.11)	**LR**(1.87)	**RF**(2.50)
2	**LDA**(3.41)	**LR**(2.22)	**MLP**(3.20)	**SVM**(3.00)
3	**MLP**(3.54)	**SVM**(2.83)	RF(3.91)	**LR**(3.22)
4	**LR**(3.54)	**RF**(3.88)	LDA(3.91)	**MLP**(3.72)
5	**QDA**(3.70)	BST(4.94)	SVM(4.0)	**BST**(3.94)
6	**BST**(4.25)	LDA(5.0)	BST(4.33)	LDA(5.16)
7	RF(6.20)	QDA(7.0)	QDA(6.74)	QDA(6.44)

From the total of 21 subjects in the synchronous BCIs datasets, 12 had performed binary tasks and 9 had performed multi-task control of BCIs. Therefore, we performed separate statistical tests for binary and for multi-task datasets. The average rank of different classifiers for the binary and for the multi-task datasets are given in Table 2.7. This Table shows that in binary-task BCIs for both BP and Morlet features the best performing classifier is an inherently binary classifier (i.e., SVM in binary-task BCIs with BP features and LR in binary-task BCIs with Morlet features). Furthermore, in multi-task BCIs for both kinds of features the best performing classifier is an inherently multi-class classifier (i.e. MLP in multi-task BCIs with BP features and RF in multi-task BCIs with Morlet features).

Table 2.8: P-values corresponding to pairwise comparison of different classifiers. α is chosen to be 0.1. For binary task BCIs with BP features all hypothesis with p-value less than 0.02 are rejected. For multi-task BCIs with BP features all hypothesis with p-value less than 0.0333 are rejected. For binary task BCIs with Morlet features all hypothesis with p-value less than 0.1 are rejected. For multi-task BCIs with Morlet features all hypothesis with p-value less than 0.025 are rejected. The results are rounded up to 4 decimal places.

(Binary, BP)		**(Multi-task, BP)**		**(Binary, Morlet)**		**(Multi-task, Morlet)**	
hypothesis	$P-value$	hypothesis	$P-value$	hypothesis	$P-value$	hypothesis	$P-value$
RF vs. SVM	0.0011	QDA vs. MLP	0.0	QDA vs. LR	0.0	QDA vs. RF	0.0001
BST vs. SVM	0.2986	LDA vs. MLP	0.0045	BST vs. LR	0053	LDA vs. RF	0.0088
QDA vs. SVM	0.6706	BST vs. MLP	0.0053	SVM vs. LR	0.0159	BST vs. RF	0.1560
LR vs. SVM	0.8132	RF vs. MLP	0.0808	RF vs. LR	0.0206	MLP vs. RF	0.2300
MLP vs. SVM	0.8132	SVM vs. MLP	0.4781	LDA vs. LR	0.0206	LR vs. RF	0.4781
LDA vs. SVM	0.9247	LR vs. MLP	0.9131	MLP vs. LR	0.1305	SVM vs. RF	0.6234

For each group of subjects, we performed the Friedman test and the Holm's post-hoc test. This is done to statistically compare the performance of other classifiers with respect to the best performing classifier in each case. The p-values corresponding to pairwise comparison of classifiers are given in Table 2.8. Table 2.8 suggests that for the BP features, MLP, LR and SVM are recommended for both the binary and multi-class BCIs. According to Table 2.5, these classifiers are also recommended in Setting 1 (corresponding to 21 subjects performing synchronous BCIs with BP feature extraction method). Table 2.8 also suggests that, for Morlet features, LR and MLP classifiers are recommended for both the binary and multi-class BCIs. According to Table 2.5, these classifiers are among the recommended ones in Setting 2 (corresponding to 21 subjects performing synchronous BCIs with Morlet feature extraction method). The results suggest that among the classifiers that are designed to handle multi-task classification, RF and MLP are recommended for multi-task BCIs. The other observation is that even in multi-task BCIs, some inherently binary classifiers are among the recommended classifiers. The results of Table 2.5 are, however, more reliable as the number of subjects in Settings 1 and 2 is almost twice the number of subjects in Table 2.7.

2.5 Conclusion

In this chapter, we built a general open source framework to compare the performance of different algorithms in BCIs. Using this framework, we performed a comprehensive comparison between 14 different BCI designs (two feature extraction methods and seven classification methods for each feature extraction methodology) over 29 BCI subjects performing sensory motor tasks in synchronous and self-paced BCI paradigms. We backed our results with rigorous statistical tests.

Our results show that the Logistic Regression (LR) and Multi-Layer Perceptron (MLP) classifiers are among the best performing classifiers and are recommended for all different designs. LR is a linear classifier like Linear Discriminant Analysis (LDA) while MLP is a very powerful nonlinear classifier. Both LR and MLP classifiers are prone to over-fitting; however, in both cases we have included regularization terms to avoid over-fitting. The observation that LR was among the best classifiers suggested that the feature space of our task was somewhat linearly separable.

Finally, we should emphasize that classification is just one step in our framework, and to get acceptable performance other steps are also important. Pre-processing of the data, feature extraction, and feature selection all change the distribution of the data in the feature space and have a major role in getting good results. Therefore, a BCI system should be viewed as a unit consisting of different blocks in which all the block settings and parameters should be adjusted jointly for each individual subject.

Chapter 3

Customizing Brain Computer Interfaces

3.1 Introduction

As discussed in chapter 1, different subjects have different brain characteristics, and a BCI system should be customized for each individual subject. For example, the frequency bands in which the alpha and beta activities occur differs from one individual to another. In sensory motor BCIs, the event related desynchronization (ERD) and the event related synchronization (ERS) occur in these frequency bands [79].

Generally the range of the alpha and beta brain waves are $\approx [8-12]Hz$ and $\approx [16-24]Hz$ respectively. However, the value of these brain waves vary from one person to another [77]. Finding these frequency bands would thus play a significant role in obtaining a desirable performance. In synchronous BCIs, the selection of the EEG time segment (from which the features are extracted), for each person has a major role in determining the accuracy of a learning system. Also, in BCI systems, the choice as to which channels to use is very important. These subject-specific brain characteristics form some of the hyper-parameters that are fed to a BCI system. The value of these hyper-parameters are determined before the feature extraction and classification processes.

The problem of finding the appropriate values for the hyper-parameters is known as hyper-parameter optimization. This problem is of great importance in any machine learning task. Two of the most common approaches for finding good values for hyper-parameters are grid search and manual search. To apply a grid search method on a continuous space, we usually discretize the space. In the grid search approach, when the dimensionality of the search space, or the granularity of the discretization of continuous hyper-parameters increases, the computation time to carry out the search process increases vastly (curse of dimensionality). Manual search, has the drawback that its results are not reproducible. This is because researchers usually do not report the hyper-parameter values used in their methods, and this makes the reproduction of their results difficult. Both these methods are thus not applicable in high-dimensional search spaces. Therefore, there is a need for smart methods that find good solutions efficiently i.e. using a small number of function evaluations.

In the BCI field, the values of these hyper-parameters have been mostly selected manually i.e. selected by EEG experts. There exists some prior work in the field that have partially addressed the hyper-parameter optimization task. Some studies only adjust the frequency bands and the time interval [93] [18] [28] [94] [31] [11], and some others address the channel selection problem [45] [104] [5] [61]. These methods are however not scalable, they also do not optimize all the hyper-parameters jointly. Furthermore, in these methods the process of selecting the subject specific hyper-parameters is completely independent from the classification and feature extraction processes. That is, the hyper-parameter optimization task has no knowledge of the nature of the feature extraction and classification algorithms.

In this work, we take a totally different approach from those in the literature. An ideal learning framework for a BCI problem should aim at studying the hyper-parameter optimization, the feature extraction and the classification processes jointly. In other words, the hyper-parameters of the BCI system should be optimized based on the selected classifier and feature extractor methods.

In this chapter, we first show the importance of finding the optimal values for the hyper-parameters of BCIs. Then, we propose a learning framework that is capable of performing subject-specific customization for BCIs. we use Bayesian optimization to tune the hyper-parameters of the BCI system. The proposed framework is computationally in-expensive and can be used to customize any number of hyper-parameters in a BCI, and the experiments have led to significant improvements in motor imagery based BCIs.

3.2 Bayesian Optimization

Bayesian optimization (BO) is capable of optimizing functions that do not have closed form mathematical expressions and are computationally expensive to evaluate [23]. This approach has several advantages over other global optimization algorithms [54] and has thus gained much attention for hyper-parameter optimization in machine learning. Bayesian optimization is very efficient in terms of the number of function evaluations. Its efficiency originates from its ability to incorporate prior knowledge about the optimization task. Especially in the case of optimizing the hyper-parameters, the candidate points in the neighborhood of a certain point x have almost the same function value (i.e. the function being optimized is smooth). In Bayesian optimization, we can incorporate this domain knowledge about the system through a Kernel function. Bayesian optimization uses all the information gained from previous function evaluations to find a new candidate point. This is done by utilizing the global information captured in the probability distribution fitted to the data to propose a new candidate.

Bayesian optimization optimizes an objective function $g(x)$ over some bounded set X. The objective function does not have a closed form expression and its derivative is unknown. A Bayesian optimization algorithm sequentially constructs a probabilistic model for $g(x)$ and then uses this model to select a candidate for the optimization task. In our case, the objective function is the accuracy of the learning framework used.

Let us assume that at time $t-1$, the sequence $D_{1:t-1} = \{(x_1, g(x_1)), (x_2, g(x_2)), ..., (x_{t-1}, g(x_{t-1}))\}$ have been observed. The Bayesian optimization algorithm creates a posterior distribution over the objective function g given the observations i.e. $P(g \mid D_{1:t-1})$. This posterior function expresses the algorithm's belief about the objective function g, and can be seen as a way of estimating the objective function.

Based on this posterior function, the Bayesian optimization algorithm proposes another candidate to be evaluated in the next iteration. In particular, it creates a new utility function based on the posterior distribution $P(g \mid D_{1:t-1})$. This function is called the acquisition function. Unlike the original optimization task, finding the maximum of the acquisition function is not difficult, however this function is still nonconvex. To optimize this function, a gradient descent algorithm or a derivative free optimization algorithm can be used. A local maximum of the acquisition function is the new candidate (i.e. x_t). This whole process is continued for T iterations.

For the Bayesian optimization algorithm, there are two main components that should be determined in advance. The first component is how to model the posterior $(P(g \mid D_{1:t}))$ distribution over the objective function. Such modeling should be powerful enough to be able to model complex posterior distributions of the objective function. This is discussed in Section 3.2.1. The second component is the choice of the acquisition function. This is discussed in Section 3.2.2.

3.2.1 Modeling Posterior Distribution

To model the posterior distribution two of the most powerful methods for regression are used. The first method utilizes Gaussian Processes (GP) [83] which is a very convenient method for modeling non-linear regression tasks (Section 3.2.1). GPs assume the correlation between samples in all parts of the space has the same structure, and they fail to model cases in which the dataset is piece-wise continuous. The second approach utilizes Random Forests (RF) for the regression task (Section 3.2.1). In [29] Criminisi et al. have compared the performance of RF and GP regression methods. They stated that GP and RF have different behaviors when large gaps exist in the training data. In such cases RFs can model the uncertainty in the gaps appropriately. They also concluded that RFs can model multi-modal distributions of data in the gaps while GPs are intrinsically uni-modal Gaussian distribution.

Gaussian Processes (GP)

Gaussian Processes (GPs) have been shown to be well-suited to estimate the posterior distribution over an objective function [72]. GPs are one of the most popular families of stochastic processes for modeling regression problems. One characteristic of GPs is their ability to model the uncertainty of predictions. A GP is a non-parametric learning algorithm which represents an infinite dimensional Gaussian distribution over functions. A good property of GP is that solving the prediction problem associated with GPs is straightforward and there is a closed form solution for that.

As is the case of an ordinary Gaussian distribution, Gaussian processes can be fully determined by their first and second order moments. As a result, to model a regression problem with a GP, the mean and the variance i.e. kernel function of the GP should be determined. The family of functions represented by a GP is determined by the choice of the mean and kernel function of the prior distribution. Usually we assume the mean of the prior is zero and only the kernel function defines the functions represented by a GP. Such a kernel function controls the smoothness of the functions represented by a GP. This is specially useful in hyper-parameter optimization, where we can design a smooth posterior over the hyper-parameter space by choosing an appropriate kernel.

To model the posterior distribution with a GP, we assume that the sequence of hyper-parameter candidates up to time $t-1$ is $x_{1:t-1} = [x_1, x_2, ..., x_{t-1}]^T$, and the corresponding values of the objective function for this sequence are $g_{1:t-1} = [g(x_1), g(x_2), ..., g(x_{t-1})]^T$. At time t, the joint density distribution of $g_{1:t-1}$ and g_t (i.e. $g(x_t)$) is Gaussian, i.e.,

$$g_{1:t} \sim N\left(0, \begin{bmatrix} K & k_{t,1:t-1} \\ k_{t,1:t-1}^T & k_{t,t} \end{bmatrix}\right), \tag{3.1}$$

where K is the kernel matrix and is the result of applying the kernel function ($k_{i,j} = k(x_i, x_j)$) on all pairs of previous observations $x_{1:t-1}$:

$$K = \begin{bmatrix} k_{1,1} & ... & k_{1,t-1} \\ . & . & . \\ . & . & . \\ . & . & . \\ k_{t-1,1} & ... & k_{t-1,t-1} \end{bmatrix}. \tag{3.2}$$

In Equation 3.1, $k_{t,1:t-1}$ is $[k_{t,1}, k_{t,2}, ..., k_{t,t-1}]^T$ and $k_{t,t} = k(x_t, x_t)$. Then the posterior distribution is calculated using equation 3.3:

$$P\big(g_t | g_{1:t-1}, x_t\big) = N\big(\mu_t, \sigma_t^2\big), \tag{3.3}$$

where

$$\mu_t = k_{t,1:t-1}^T K^{-1} g_{1:t-1}, \tag{3.4}$$

$$\sigma_t^2 = k_{t,t} - k_{t,1:t-1}^T K^{-1} k_{t,1:t-1}. \tag{3.5}$$

Random Forests (RF)

If we use random forests to model the posterior distribution, at each iteration t of the Bayesian optimization algorithm, a random forest is trained using $D_{1:t-1}$. Assuming M regression trees are trained on the data, then:

$$p\big(g_t|g_{1:t-1},x_t\big) = \frac{1}{M}\sum_{m=1}^{M} p_m\big(g_t|g_{1:t-1},x_t\big) \tag{3.6}$$

where $p_m\big(g_t|g_{1:t-1},x_t\big)$ is the probability density function over the output variable produced by the m^{th} tree.

3.2.2 Acquisition Function

The second component of a Bayesian optimization algorithm is the acquisition function. The role of the acquisition function is to direct the search process to the new candidate points. The new points should be selected so that they have higher probability i.e., yield higher values of the objective function. A good acquisition function should be able to balance the trade-off between exploration and exploitation. In other words, the algorithm should search globally in the beginning of the optimization process. As the search proceeds, the posterior over the objective function becomes more accurate, and the algorithm should become more exploitative.

One simple acquisition function is to select a candidate that has the maximum probability of improving the objective function value over the current best optimal solution. However, this method does not explore the search space properly, and the points that are infinitesimally greater than the current best value will have a higher chance to be selected. A better alternative is to maximize the Expected Improvement (EI). EI considers both the probability of improvement and the amount of improvement.

$$x_t = \underset{x}{\operatorname{argmax}} \mathbb{E}(\max(0, g(x) - g^+) \,|\, D_{1:t-1}), \tag{3.7}$$

where g^+ is the maximum observed value until iteration $t-1$. The maximization inside the expectation in equation 3.7 states that we only evaluate those points which our model believes would result in improvement over the current best point.

Under the assumption of a Gaussian posterior distribution, EI can be evaluated analytically. Therefore, the expectation for a possible candidate x_t in Equation 3.7 can be evaluated, as following:

$$\begin{cases} (\mu_t - g^+)\Phi(Z_t) + \sigma_t \phi(Z_t) & \sigma_t > 0 \\ 0 & \sigma_t = 0 \end{cases}, \tag{3.8}$$

and

$$Z_t = \frac{\mu_t - g^+}{\sigma_t}, \tag{3.9}$$

where $\phi(.)$ and $\Phi(.)$ are the probability density function and the cumulative distribution function of a standard Gaussian distribution. The candidate with the maximum EI value in equation 3.8 is selected as the candidate for the t^{th} iteration of Bayesian optimization. For both Gaussian process and random forest posteriors, we have used equation 3.8.

3.3 Proposed Algorithm

For any BCI system, the goal is to achieve the best possible accuracy. The traditional approach is to model the BCI design task as a machine learning problem and solve the problem accordingly. In this section however, we solve the BCI task from an optimization point of view. We define an objective function (Section 3.3.1), and we then use a Bayesian optimization algorithm (Section 3.3.2) to optimize this function. Our objective function is the performance of the learning system, and we optimize this function over the hyper-parameter space. Subsequently, our algorithm builds an ensemble of classifiers (Section 3.3.3) using the results of the optimization phase.

The pseudo code of our algorithm is given in Algorithm 1. At each iteration (t) of our algorithm, the optimizer suggests a new candidate (x_t) in the hyper-parameter space, then a new classifier (l_t) is built based on the values of the new candidate. Based on the performance of l_t at iteration t, another candidate is suggested for iteration $(t+1)$ and this process continues for T iterations. After T iterations, we have T classifiers, each trained on a different subset of the hyper-parameter space. The outputs of the classifiers are then combined into a single prediction rule which is much more accurate than the individual classifiers.

Algorithm 1: The pseudo-code of the proposed algorithm

```
1  Select an $x_1$ from the hyper-parameter space uniformly at random;
2  Create a feature matrix $f_1$ from the brain signals using $x_1$;
3  Train a new Classifier ($l_1$) using feature matrix $f_1$, and calculate the its performance
     $g_1$;
4  Create the posterior distribution;
5  for  $t$:=2 to $T$ do
6      Find candidate ($x_t$) from the hyper-parameter space (equation 3.7);
7      Create a feature matrix $f_t$ from the brain signals using $x_t$;
8      Train a new Classifier ($l_t$) using feature matrix $f_t$, and calculate the its
         performance $g_t$;
9      Update the posterior distribution $p(g_t|g_{1,t-1},x_t)$ (equations 3.3 for GP and 3.6
         for RF);
10 end
11 Combine $l_1 ... l_T$ to build an ensemble classifier;
```

3.3.1 Learning Algorithm (Objective Function)

The objective function $g : H \to A$ is a function that maps the hyper-parameter space (H) to the set of possible performances (A) of the learning system. Using Bayesian optimization we decide which hyper-parameters yield the best performance. Below, we explain our learning system in more details.

We used the learning framework described in section 2.2 as our learning system. For frequency filtering we used the fifth order Butterworth filter, and for spatial filtering we used CSP (section 2.2.1). For feature extraction we used BP and Morlet features as described in section 2.2.2. In our experiments the number of BP features extracted from each channel, were either 2 or 4 (Section 3.3.1). For the classification step, we used Logistic Regression. Comparing classifiers in sensory motor BCIs in chapter 2, Logistic Regression was among the best performing classifiers. To select the appropriate values for the parameters of the classifier, 5×5-fold cross validation is performed. The parameters with the best mean performance were used to train the classifier.

The accuracy in the classification is used as the performance measure for evaluating different algorithms. As the number of samples are the same for different imagery movements, this measure is appropriate to use in this study. Therefore, the output domain of the objective function is taken as the cross validation accuracy of all possible classifiers. In other words, our algorithm uses 5×5-fold cross-validation to assess each classifier.

Input Domain of the Objective Function (Hyper-Parameter Space)

In this section, we explain the domain i.e. the hyper-parameter space of the objective function. We have considered three groups of parameters to form our hyper-parameter space. We find the values of these hyper-parameters simultaneously for each subject:

(i) the time interval from which features are extracted: In order to explain this hyper-parameter, a sequence of trials in a movement sensory BCI dataset is illustrated in Figure

3.1. Each trial consists of a movement interval (left or right hand movement) and a No-Control (NC) interval. During a movement interval, the subject is requested to perform a movement imagery task, and in an NC interval the subject should stop controlling the system.

As depicted in Figure 3.1, the movement and NC interval are consecutive. The presence of these transition states (i.e. switching from a NC state to movement state or vice versa) in each brain state segment makes classification more difficult. To alleviate this problem, a chunk of the signal is discarded from the beginning and the end of the trials. The features are extracted from the remaining signal.

Since, this BCI hyper-parameter has a real value, we have discretized the search space in our experiments.

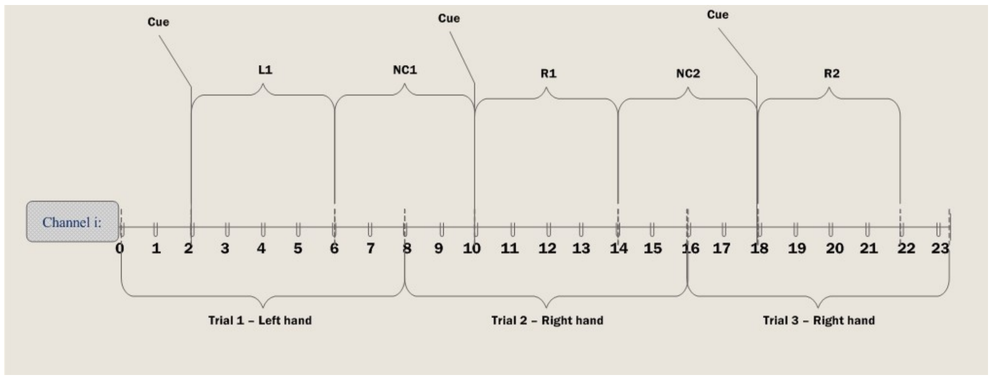

Figure 3.1: An example of sequences of trials in BCI calibration phase in which each trial is 8s. In the first trial, the subject performed a left hand (L1) imagery movement followed by a No-Control (NC1) interval of 4s. In the second trial, the subject performed a right hand imagery movement (R1) followed by a 4s NC interval.

(ii) EEG Channels: For datasets containing data from many channels (dataset IV2a), selecting a small set of appropriate channels forms another hyper-parameter of the system. Our algorithm selects the number of CSP filters (N = 2, 4, 6) or has the option of not applying CSP (i.e. uses all the channels without applying any spatial filtering).

(iii) Frequency bands: Another important hyper-parameter in sensory motor BCIs, is the choice of the frequency bands used for feature extraction. As discussed in Section 3.1, ERD and ERS occur in the upper alpha and the lower beta rhythms. This hyper-parameter only applies to BP features. As mentioned in Section 2.2.2, for Morlet features, the frequency ranges were selected based on [24].

For BP features, in each iteration of the optimization algorithm, the algorithm selects between two options. The first option is to apply a filter-bank with 2 separate blocks corresponding to the alpha and the beta frequencies of the brain on each channel, and then, extract band-power of the filtered signal. The second option is to apply a filter with a large bandwidth in the range $\approx [4-35]$Hz that includes both alpha and beta brain waves of each channel.

The frequency band hyper-parameter is real valued, therefore we have discretized the space.

3.3.2 Selecting Hyper-Parameters in BCIs

Choice of the Bayesian Optimization Algorithm

Two different Bayesian optimization algorithms were used here. The first one uses Gaussian process posterior with Matern 5/2 Kernel [92]. This covariance function results in functions that are twice differentiable, an assumption used to perform quasi-newton optimization. The behavior of the prior function is governed by the choice of its parameters. The most common approach to derive appropriate values for the parameters is to use a point estimate (e.g. Maximum Likelihood) of these values. However in [92] a fully Bayesian approach is used to obtain a marginalized acquisition function. The acquisition function is the expected improvement (EI), as it can be written in a closed form.

As mentioned in Section 3.2.1, a Gaussian distribution is uni-modal, and it is not appropriate for use in modeling the data when the underlying distribution is multi-modal. To handle such cases, we have also applied a second type of Bayesian optimization which uses Random Forest (RF) posterior. The acquisition function used for RF posterior is the expected improvement (EI).

Random search

In addition to the Bayesian optimization algorithm, we have also used random search for finding the hyper-parameter values. Bergstra et al. have shown empirically and theoretically that random search can perform as good as grid-search [10] in less computational time.

3.3.3 Results Aggregation

After performing optimization on the objective function, we obtain T classifiers which are trained using the T candidate points proposed by the search algorithm (i.e. Bayesian optimization or random search). We can use the classifier with the minimum cross validation error as our final classifier. However, instead of selecting a single classifier to predict the class label of a given new sample, we exploit all the information gained from training several classifiers on the data, by aggregating all the results. The above hyper-parameter optimization leads to classifiers which are trained on different parts of the data. We call these classifiers the level one classifiers. Each level one classifier can be considered as an expert on a specific part of the dataset. We then combine all these classifiers into one final meta-level classifier using three different methodologies as explained below:

(i) Voting: Given a new sample, the output of each classifier (i.e. the predicted class label) is considered as the vote of that classifier. Then the label with the majority of votes is selected as the final label for a given sample. This method is most useful for non-probabilistic classifiers.

(ii) Averaging: In probabilistic classifiers, the classifier output represents its belief about the class label of a new sample. Then based on the classifiers belief, we decide the label of the new sample. This refinement of the voting method for probabilistic classifiers, uses the belief of each classifier for final prediction. In this method we take the average of the probabilities returned from each level one classifier. In our case the number of classifiers is the same as the number of iterations of the hyper-parameter optimizer (i.e. number of candidates we have examined).

$$P(C_{final} = k|f) \propto \frac{1}{T}\sum_{i=1}^{T} P(c_i = k|f) \qquad\qquad k \in 1, 2, ..., K \tag{3.10}$$

Where T is the number of iterations (i.e the number of level one classifiers), K is the number of classes and f is the new sample. C_{final} represents the final classifier's label.

(iii) Stacking: In this method, a learning algorithm is employed to combine the classifiers. A meta level learning algorithm decides the label of a new given sample.

One of the most successful stacking methods is the multi-response linear regression (MLR) algorithm [36] [95]. In this meta learning algorithm, a classification problem with K output classes is converted into K regression problems. Given a new sample, the belief of the meta level classifier about each class label is a linear combination of the class probability of all T classifiers. Therefore, the linear regression for class k is given by

$$P(C_{final} = k|f) = \sum_{i=1}^{T} \alpha_{ki} P(c_i = k|f), \tag{3.11}$$

where $P(c_i = k|f)$ is the class probability of the i^{th} classifier, $P(C_{final} = k|f)$ corresponds to the class probability of the final (meta) classifier and the parameters (α_{ki}) of the linear models are learned using least squares.

Given a new sample f, we first plug f in all of the T classifiers, then we calculate the class probabilities $P(c_i = k|f)$ for each k. Finally, we plug the class probabilities corresponding to the class k in each of the K linear models. The class k with the maximum value of $P(C_{final} = k|f)$ is selected as the final label of the given sample (f).

3.3.4 Termination of the Algorithm

In our Bayesian optimization algorithm, the algorithm stops when the termination condition is reached, otherwise the algorithm continues for T iterations. In fact, T is the maximum possible number of iterations for our algorithm. The termination condition basically checks if there has been an improvement in the cross-validation accuracy in the last τ iterations of the algorithm. To check if there is no improvement, we use the following equation:

$$\frac{MAX - min}{m_1} \leq \varepsilon \tag{3.12}$$

In equation 3.12, MAX is the maximum cross validation accuracy in the last τ iterations of the algorithm, min is the minimum cross validation accuracy in the last τ iterations of the algorithm, and m_1 is the initial cross validation accuracy at time $t = 1$. ε is the threshold to stop the algorithm.

3.3.5 Time Complexity of the Algorithm

There are three factors that govern the running time of the Bayeisan optimization algorithm. 1) the sum of the running times of feature extraction and classification, 2) the running time of each iteration of the Bayesian optimization, and 3) the number of iterations of the algorithm.

1) The sum of the running times of classification and feature extraction: In our case, the running time of this part is small. The running time of extracting Morlet and BP features is low. The running time of Logistic Regression is also very low. The Hessian of the cost function in Logistic Regression can easily and quickly be calculated. Using second order optimization algorithms makes the convergence of the algorithm very fast.

2) Running time of each iteration of Bayesian optimization: The running time of each iteration of the Bayesian optimization is governed by the complexity of building the pos-

terior distribution, and in finding the optimal value of the acquisition function. The complexity of Gaussian process is $O(T^3)$ (assuming the parameters of the covariance function are known). T is the number of Bayesian optimization iterations. The complexity of generating a Random Forest is $O(M(mT logT))$, M is the number of trees, m is the number of hyper-parameters selected to split each node of the tree. In our case M was equal to 50. In Bayesian optimization, the size of T is very small so the complexity of one iteration of Bayesian optimization is almost negligible.

3) The number of Bayesian optimization iterations: Depending on the dimensionality of the hyper-parameter space, the number of iterations required by Bayesian optimization (to converge) increases. The overall running time of the Bayesian optimization is the number of iterations multiplied by the sum of the running time of feature extraction and classification, and the running time of each step of Bayesian optimization.

3.4 Results and Discussion

Three sensory motor BCI datasets consisting of 21 subjects were used to evaluate the methodology proposed in this chapter. These are datasets IIa [26] and IIb [64] of BCI competition IV, and dataset IIIa [16] of BCI competition III. All datasets are for synchronous BCIs. We have used the training data of these datasets for our training phase and tested our method on their test data (section 2.3).

To show the importance of finding the appropriate values for hyper-parameters, we have first compared the performance of different search (optimization) algorithms in Section 3.4.1. Then in Section 3.4.2, we have compared our results to those reported in the literature. In both cases, we have conducted separate statistical tests. To gain more insights about the effects of the number of optimization iterations we have performed a set of experiments which are described in Section 3.4.3.

3.4.1 Comparing with Other Hyper-Parameter Search Algorithms

The purpose of this section is to compare our fully automated algorithm with other search methods (e.g. random search, manual search), and demonstrate the importance of hyper-parameter optimization.

The most common approach to search for the hyper-parameter values is the grid-search. Since, the BCI hyper-parameter space has real valued hyper-parameters (e.g. frequency bands for alpha and beta bands), we have discretized the search space (section 3.3.2). Unfortunately, even with discretization, the search space would still contain hundreds or thousands of candidate points to examine. This makes it infeasible to find the optimal values using the grid search. Therefore, we compare our method with the manual search method. In the manual search method, a set of hyper-parameter values that were specified by the user are tested.

We tested 3 different automated optimization algorithms: 1) Bayesian optimization with Gaussian Process (BO-GP) posterior, 2) Bayesian optimization with Random Forest (BO-RF) posterior and 3) random search. The initial point used to start the Bayesian optimization was selected randomly. For each of the 3 search algorithms, we aggregated the results of the trained classifiers using the three methods discussed in section 3.3.3; MLR, voting and averaging. We also reported the results of the best classifier. This classifier was trained using the candidate hyper-parameter set that had minimum cross-validation error.

To extract the Band Power (BP) features, we optimized 1) the part of the movement interval from which the features are extracted, 2) the frequency bands used to extract features, and 3) the number of CSP filters for dataset (IV2a) with 22 channels. To extract the Morlet features, we optimized the same parameters excluding the frequency bands, because we used the same frequency bands as [24] for the Morlet features.

In addition to the results obtained by our proposed fully automatic algorithm and the manual search method, we have also included the results of using the default values of the hyper-parameters. In our results, we refer to this as the ”No-Search” method. In this method, the features were extracted from the whole segment of the movement interval, and no channel selection was performed. Also, in the case of BP features, the frequency bands were in the range $[8-12]Hz$ and $[16-24]Hz$ which correspond to the standard alpha and beta frequencies of the brain.

The results of applying all these search algorithms when the Morlet features are used are given in Table 3.1, and those for the BP features are given in Table 3.2. For datasets III3b and IV2b, the maximum number of iterations was 40 (i.e. $T = 40$), the maximum number of iterations for dataset IV2a was 60, and for all algorithms ε (equation 3.12) was e^{-4}. We have repeated our automatic search algorithms 10 times to reduce the effects of the random seed.

The Qualitative comparison of the different algorithms in Tables 3.1 and 3.2 suggests the following results: 1) for almost all subjects, Bayesian optimization significantly improves the results compared to the manual search method. There are however some cases in which Random Search has the best performance. 2) For all subjects, the results of applying Bayesian optimization is significantly better than the results of using the default values of the hyper-parameters (i.e. No-Search columns in Tables 3.1 and 3.2). 3) For the Morlet features (Table 3.1), Bayesian optimization with Random Forest posterior outperforms other methods. 4) For BP features (Table 3.2), Bayesian optimization with Gaussian process posterior outperforms other methods.

As discussed in 2.2.6, according to Demšar et al. [32], when comparing several algorithms across several datasets, comparing the average ranks of different algorithms is more reliable than comparing average accuracies. Comparing the average ranks of different algorithms shows that for the Morlet features, the best performing algorithm is BO-RF (AVG), and for the BP features, the best algorithm is BO-GP (AVG).

Table 3.1: Accuracy of each algorithm with Morlet features. For each search algorithm we used MLR, voting (VOTE) and averaging (AVG) for aggregating the results of the classifiers. MIN is the result of the best level 1 classifier in optimization phase. Highlighted cells show the configuration for which the performance is the best.

	No	**Manual**	**BO-GP**				**BO-RF**				**Random Search**			
	Search	**Search**	MLR	VOTE	AVG	MIN	MLR	VOTE	AVG	MIN	MLR	VOTE	AVG	MIN
subject 1	73.58	82.39	86.10±0.91	83.77±0.55	83.84±0.80	84.28±0.00	85.60±0.77	82.70±0.42	83.65±0.69	84.28±0.00	84.28±0.63	83.14±1.01	81.07±0.71	83.96±0.94
subject 2	83.88	83.89	84.19±0.09	83.30±0.26	83.35±0.23	83.15±0.00	84.20±0.12	83.11±0.23	83.28±0.35	83.15±0.00	83.81±0.42	82.17±0.58	82.80±0.29	82.39±1.20
subject 3	77.4	78.15	86.20±0.22	83.89±0.23	84.98±0.33	80.74±0.00	86.46±0.31	83.37±0.44	84.70±0.45	80.74±0.00	86.46±0.55	83.74±0.51	84.98±0.54	80.70±0.11
subject 4	68.85	68.86	73.33±0.26	70.44±0.59	70.70±0.70	70.18±0.00	73.68±0.48	70.83±0.35	71.40±0.75	69.69±1.45	73.51±1.24	70.13±1.15	71.80±0.79	70.35±2.15
subject 5	57.95	58.37	56.82±1.34	58.73±0.94	58.37±0.45	58.78±0.00	56.53±1.57	61.27±1.36	60.57±1.21	57.92±1.23	57.14±1.49	58.69±0.94	58.41±0.76	57.22±1.19
subject 6	55.21	53.48	53.26±1.74	54.96±0.71	55.74±0.91	56.87±0.26	53.48±0.00	58.52±0.40	58.09±0.29	56.96±0.00	53.48±1.63	54.87±1.06	55.70±1.31	55.35±3.03
subject 7	96.09	94.79	96.06±0.27	96.64±0.21	96.35±0.24	97.62±0.15	95.44±0.00	97.04±0.18	97.13±0.20	97.69±0.10	96.19±0.90	95.96±0.49	96.48±0.41	97.04±0.76
subject 8	86.44	91.58	88.86±0.72	91.61±0.45	91.50±0.32	88.39±0.17	89.19±0.34	90.22±0.33	91.32±0.23	88.46±0.18	89.41±1.09	90.84±0.91	90.33±0.85	86.92±3.03
subject 9	84.86	82.87	85.62±0.28	84.86±0.31	85.10±0.32	85.66±0.00	85.86±0.74	86.18±0.86	85.94±0.36	85.46±1.47	85.26±1.39	85.10±0.69	85.46±1.03	84.86±1.61
subject 10	73.27	72.84	72.11±0.51	75.34±0.42	74.57±0.61	72.41±0.00	72.41±0.70	77.37±0.59	76.98±0.65	72.59±0.52	74.83±1.52	76.98±0.93	75.86±0.96	72.41±1.49
subject 11	85.21	83.48	89.09±0.23	88.43±0.21	88.65±0.13	84.78±0.00	89.09±0.41	91.35±0.71	91.96±0.68	85.22±1.51	89.30±0.83	91.17±0.87	90.91±1.17	85.30±0.80
subject 12	85.3	86.12	81.96±0.31	85.92±0.46	87.06±0.41	85.71±0.00	79.71±1.16	84.61±0.73	84.53±0.76	85.63±0.24	81.55±1.81	84.78±1.13	84.82±0.87	84.82±1.25
subject 13	54.51	77.77	63.23±4.52	84.86±0.62	85.00±0.66	83.68±0.00	68.85±9.76	81.53±1.44	82.12±1.20	83.68±0.00	68.06±8.33	82.08±1.46	82.29±1.31	82.26±2.69
subject 14	34.72	52.77	35.49±1.35	44.51±0.43	44.03±0.97	46.53±0.00	30.07±2.80	40.94±1.48	44.86±2.15	42.43±3.10	33.72±1.64	40.76±1.17	42.33±0.91	46.08±3.40
subject 15	59.72	86.11	82.85±0.78	86.18±0.51	86.60±0.59	85.28±0.23	82.26±1.28	85.66±0.97	86.60±0.88	84.97±0.31	81.32±3.44	83.85±1.21	84.79±0.69	85.00±1.57
subject 16	44.44	47.91	43.65±4.79	60.49±0.73	61.98±1.31	59.38±0.00	41.04±6.30	64.86±1.06	66.28±1.44	58.82±1.06	48.23±8.20	61.91±1.61	62.33±1.95	58.19±2.28
subject 17	32.638	38.54	43.65±0.16	43.06±1.24	43.82±0.95	42.71±0.00	38.72±2.44	46.08±1.18	48.72±1.54	43.78±2.39	33.19±4.63	44.72±1.88	46.04±2.24	43.09±2.34
subject 18	36.8	42.7	47.29±1.61	44.90±0.68	47.50±1.16	41.94±1.60	44.10±0.38	53.61±1.57	53.30±1.55	44.79±0.00	41.70±2.81	47.74±1.93	51.32±1.46	42.53±1.30
subject 19	58.68	58.33	64.55±1.46	67.60±0.35	69.41±0.96	64.93±0.00	67.15±3.09	70.07±1.36	72.64±1.78	64.03±1.36	66.53±4.33	72.05±1.67	75.03±1.15	64.48±2.04
subject 20	43.75	79.16	83.51±0.70	82.57±0.58	83.75±0.67	83.40±0.14	79.76±6.44	80.66±1.97	82.33±2.18	79.55±3.18	72.50±10.80	82.50±1.84	83.23±1.70	80.66±1.53
subject 21	49.65	77.08	75.00±0.00	78.65±1.23	79.41±0.93	79.51±0.00	73.85±3.44	76.18±0.78	76.35±0.74	79.51±0.00	72.36±4.65	76.01±1.43	76.74±1.26	76.84±3.31

Table 3.2: Accuracy of each algorithm with BP features. For each search algorithm we used MLR, voting (VOTE) and averaging (AVG) for aggregating the results of the classifiers. MIN is the result of the best level 1 classifier in optimization phase. Highlighted cells show the configuration for which the performance is the best.

	No	**Manual**	**BO-GP**				**BO-RF**				**Random Search**			
	Search	**Search**	MLR	VOTE	AVG	MIN	MLR	VOTE	AVG	MIN	MLR	VOTE	AVG	MIN
subject 1	69.81	80.5	83.08±0.82	83.21±0.63	83.33±1.02	84.03±0.50	74.09±1.46	74.59±0.64	75.41±0.66	74.21±0.89	81.19±1.88	80.44±1.24	81.51±1.23	80.88±1.47
subject 2	69.62	70.19	77.59±0.37	75.87±0.34	75.85±0.36	77.78±0.00	65.65±1.49	60.61±0.63	61.33±0.92	65.11±1.52	73.74±1.09	68.24±1.04	69.43±1.48	72.07±2.22
subject 3	74.07	74.26	78.65±0.51	78.46±0.37	78.70±0.51	77.78±0.97	61.56±4.15	59.63±1.10	60.30±1.13	63.28±1.36	77.39±0.84	73.89±0.77	74.85±0.72	77.33±1.54
subject 4	62.28	60.96	67.81±0.76	67.02±1.14	67.19±1.03	66.75±0.94	50.09±3.49	52.02±1.11	53.03±2.02	54.47±2.87	65.66±1.81	61.32±0.92	62.54±0.79	66.23±1.30
subject 5	57.55	56.33	59.35±1.24	60.73±1.35	61.80±1.23	60.78±1.54	50.61±2.45	53.27±0.58	55.22±1.28	52.78±2.42	61.63±1.49	59.88±1.10	62.45±0.77	60.86±1.59
subject 6	53.91	56.09	55.91±0.85	58.61±1.73	59.35±1.43	57.13±0.52	54.30±2.60	52.48±1.15	52.87±1.39	53.87±1.71	55.78±3.41	56.74±2.24	56.09±1.08	57.78±1.91
subject 7	92.18	94.79	94.82±0.57	92.87±0.63	93.39±0.39	92.44±1.07	78.44±3.81	66.38±3.47	71.63±3.62	78.14±4.54	95.08±0.98	93.84±0.79	94.20±0.65	92.25±1.41
subject 8	63.73	67.77	72.78±0.98	70.95±1.25	71.50±1.02	71.79±0.00	55.57±2.59	56.85±1.20	57.14±1.87	57.77±2.79	71.61±1.94	64.80±0.93	65.64±1.60	70.59±3.32
subject 9	72.11	75.7	83.55±0.99	79.40±1.17	79.52±0.80	80.52±2.24	65.66±3.15	60.88±1.47	62.47±1.29	65.46±2.78	82.07±1.83	75.78±1.10	77.73±1.55	80.12±2.02
subject 10	49.56	53.02	76.21±1.50	73.23±1.16	74.57±1.31	73.71±2.75	49.14±0.00	49.14±0.00	49.78±0.65	50.47±2.76	75.17±0.99	64.91±1.14	68.88±1.07	73.58±2.39
subject 11	92.17	92.18	91.96±0.40	91.57±0.62	91.61±0.62	91.00±1.08	53.48±5.22	51.61±3.62	51.48±7.56	52.87±11.22	90.22±1.09	90.43±0.75	90.83±0.36	90.43±1.73
subject 12	77.55	77.55	85.67±0.69	84.41±0.48	83.76±0.70	83.76±1.35	52.94±4.11	52.16±1.51	53.63±2.27	56.57±3.88	81.84±1.92	78.49±1.27	77.96±0.98	80.82±3.10
subject 13	53.47	72.56	74.86±2.42	80.56±1.20	79.65±1.19	76.98±2.60	64.10±3.88	64.72±2.13	63.78±2.31	67.78±1.12	69.76±4.19	78.58±2.14	79.13±1.10	75.45±2.74
subject 14	40.27	52.08	41.46±2.24	45.24±1.55	47.33±2.02	40.49±3.25	31.67±1.50	33.30±0.88	33.58±0.82	32.78±2.52	45.14±4.03	45.31±1.88	46.56±1.29	43.85±2.66
subject 15	62.15	71.18	70.35±2.27	72.85±0.89	73.30±0.77	69.44±1.98	73.96±1.31	75.31±0.94	75.73±1.12	75.21±1.10	68.96±2.23	71.15±0.78	71.98±0.91	68.61±2.22
subject 16	45.13	60.76	58.26±1.85	59.72±0.60	61.25±0.91	57.40±1.77	57.88±1.58	58.47±2.02	59.83±1.90	57.26±2.35	58.92±3.34	64.69±0.84	63.92±1.15	57.85±2.84
subject 17	35.76	31.59	32.19±1.15	37.57±1.67	35.56±1.42	30.97±0.21	36.88±1.76	36.49±2.39	36.25±2.87	32.99±2.50	36.01±2.13	40.24±1.07	40.49±1.73	34.69±1.70
subject 18	41.66	40.62	48.61±2.02	54.44±1.61	55.45±1.22	43.23±0.17	44.38±2.62	47.74±2.82	48.12±2.37	41.49±1.38	47.12±3.97	46.63±1.76	49.72±2.30	45.52±3.07
subject 19	56.59	60.06	75.42±2.63	77.53±0.84	77.64±0.70	73.99±2.28	70.83±3.69	71.46±3.52	70.90±2.21	68.40±4.73	67.57±2.58	75.28±0.77	74.69±1.37	67.85±4.82
subject 20	56.25	78.47	75.62±1.55	79.72±0.61	80.52±1.01	75.52±1.08	74.65±3.28	77.88±2.89	80.03±2.50	76.28±3.14	78.47±2.52	81.15±1.18	81.98±1.47	76.18±2.44
subject 21	61.45	66.66	75.03±2.57	80.76±1.43	81.15±1.28	72.26±2.92	76.18±2.34	77.01±1.18	77.43±0.89	76.18±1.09	73.85±4.11	78.44±2.03	79.51±1.64	72.85±3.67

Statistical Tests

we use the Friedman test to statistically validate the obtained results. The null hypothesis assumes that all algorithms have the same performance i.e., they have the same rank. If the p-value is low enough to reject the null hypothesis, it is concluded that the difference between the algorithms is not random. If the null hypothesis is rejected, we use Bergmann's test [44] as the post-hoc statistical test. For this test, we carried out a pairwise comparison of all the five algorithms (i.e. No-Search, Manual Search, BO-GP, BO-RF and Random Search). Since our goal is to show the importance of hyper-parameter optimization and the power of our proposed method (in finding good candidate hyper-parameter values), we have compared MIN column of the automatic methods to Manual Search and No-Search scenarios. MIN column corresponds to the performance of the BCI in the test phase, when the corresponding hyper-parameter value results in the best cross-validation accuracy in the optimization phase.

In the Bergmann's test, each null hypothesis assumes that two different algorithms have the same performance. In pairwise comparison of algorithms, we have $\binom{5}{2} = 10$ hypotheses to test, and each of them can be true or false. These hypotheses are logically related i.e. some combinations of true and false hypothesis cannot hold at the same time. Bergmann's test considers the logical relation between the different hypotheses. This algorithm takes all the sets of hypotheses Ψ that can be true at the same time. Then it computes a set A, and every hypothesis H which does not belong to A is rejected.

$$A = \cup\{S : S \in \Psi \text{ and for each hypothesis } H_i \in S, P_i > \alpha/|S|\}, \tag{3.13}$$

where P_i is the p-value corresponding to hypothesis H_i, and α is the significance level.

To perform the statistical tests for each feature extraction method, we first ranked the five different search algorithms based on their performance. The ranking for each subject

Table 3.3: Average ranking and average accuracy of the optimization algorithms with (a) Morlet features and (b) BP features.

(a) Morlet Features

Method	Average Ranking	Mean Accuracy (%)
No-Search	4.21	63.95
Manual Search	3.19	71.29
BO-GP (MIN)	2.14	73.14
BO-RF (MIN)	2.30	72.83
Random Search (MIN)	3.14	72.40

(b) BP Features

Method	Average Ranking	Mean Accuracy (%)
No-Search	3.97	61.30
Manual Search	2.83	66.35
BO-GP (MIN)	2.0	69.42
BO-RF (MIN)	4.0	59.68
Random Search (MIN)	2.19	68.85

was done separately, then we averaged the rank of different algorithms over the subjects. For both the Morlet and the BP features, the average ranks of each algorithm are shown in Table 3.3.

For the Morlet features, after performing the Friedman's test, the null hypothesis was rejected (p-value = 1.2E-4), indicating that there is a significant difference between the performance of different algorithms. Bergmann's post-hoc test (Table 3.4) rejected the following hypotheses: No-Search vs. BO-GP (MIN) and No-Search vs. BO-RF (MIN). For the BP features, the null hypothesis was also rejected (p-value = 3.7E-6) when we applied Friedman statistical test. Therefore, there are significant differences between the algorithms, and the differences between their performances are not random. The Bergmann's procedure rejected the following hypotheses: No-Search vs. BO-GP (MIN), No-Search vs. Random Search (MIN), BO-GP (MIN) vs. BO-RF (MIN) and BO-RF (MIN) vs. Random Search (MIN) (Table 3.4).

Based on the results of all the above statistical tests, we can conclude that for the Morlet features, Bayesian optimization with Random Forest or Gaussian process posterior are the best suited methods for hyper-parameter optimization. Also, for the BP features, Bayesian optimization with Gaussian process posterior and Random Search algorithms are the best choices for hyper-parameter optimization.

The results of the performed statistical tests support the idea that searching for suitable values for the hyper-parameters can significantly improve the performance of the algorithm. In addition to the results of the statistical tests, we have also included in Table 3.3 the average accuracy of different algorithms across all subjects. As shown in Table 3.3, for both feature extraction methods, Bayesian optimization can considerably improve the average accuracy. Comparing the average accuracies across all subjects shows that Bayesian optimization can boost the accuracy from 63.95% to 73.14% for the Morlet features. For the BP features, the average accuracy across all subjects increases from 61.30% to 69.42% if we use Bayesian optimization.

3.4.2 Comparing the Results with Literature

As discussed in the previous section, for the Morlet features, Bayesian optimization with Random Forest and averaging (BO-RF (AVG)) yields the best results, and for the BP features, Bayesian optimization with Gaussian process posterior and averaging (BO-GP (AVG)) yields the best results. Below, we compare the results in the literature with these two algorithms (i.e. BO-RF (AVG) and BO-GP (AVG)).

Table 3.5 shows the results of comparing our algorithms to those reported in the literature for dataset III3b. For datasets IV2b and IV2a, the results are given in Tables 3.6 and 3.7 respectively. In all Tables, the last row shows the average rank of different algorithms. To compare the results of our algorithm to those reported in the literature, we have used the results of their best algorithms, reported in each of [25], [24], [106], [4] and [7].

To compare the different algorithms, we have performed the Friedman's statistical test

Table 3.4: P-values corresponding to pairwise comparison of different hyper-parameter search algorithms (a) Morlet features and (b) BP features. α is chosen to be 0.05. Bergmann's procedure rejects hypotheses 9 and 10 for the Morlet features. For the BP features, hypotheses 7, 8, 9 and 10 are rejected. The results are rounded up to 3 decimal places.

(a) Morlet Features

i	Hypothesis	p-value
10	No-Search vs. BO-GP (MIN)	0.000
9	No-Search vs. BO-RF (MIN)	0.000
8	No-Search vs. Random Search (MIN)	0.028
7	Manual Search vs. BO-GP (MIN)	0.031
6	No-Search vs. Manual Search	0.035
5	BO-GP (MIN) vs. Random Search (MIN)	0.040
4	Manual Search vs. BO-RF (MIN)	0.071
3	BO-RF (MIN) vs. Random Search (MIN)	0.087
2	BO-GP (MIN) vs. BO-RF (MIN)	0.732
1	Manual Search vs. Random Search (MIN)	0.922

(b) BP Features

i	Hypothesis	p-value
10	BO-GP (MIN) vs. BO-RF (MIN)	0.000
9	No-Search vs. BO-GP (MIN)	0.000
8	BO-RF (MIN) vs. Random Search	0.000
7	No-Search vs. Random Search (MIN)	0.000
6	Manual Search vs. BO-RF (MIN)	0.016
5	No-Search vs. Manual Search	0.019
4	Manual Search vs. BO-GP (MIN)	0.087
3	Manual Search vs. Random Search (MIN)	0.187
2	BO-GP (MIN) vs. Random Search (MIN)	0.696
1	No-Search vs. BO-RF (MIN)	0.961

Table 3.5: Accuracy of our algorithms compared to the results reported in the literature for dataset III3b. Highlighted cells show the best performing algorithm. The text inside the parenthesis corresponds to the original label of the subjects used in the BCI competition.

	BO-GP(AVG) (BP)	BO-RF(AVG) (Morlet)	results of [25]	results of [24]	results of [106]
subject 1 (O3)	83.33	83.65	77.1	80.7	89.3
subject 2 (S4)	75.85	83.28	81.5	81.7	72.96
subject 3 (X11)	78.7	84.7	80.4	80.9	74.26
Average	79.29	83.88	79.67	81.1	78.84
Average Rank	3.67	1.33	3.67	2.67	3.67

Table 3.6: Accuracy of our algorithms compared to the results reported in the literature for dataset IV2b. Highlighted cells show the best performing algorithm. The text inside the parenthesis corresponds to the original label of the subjects used in the BCI competition.

	BO-GP(AVG) (BP)	BO-RF(AVG) (Morlet)	results of [4]	results of [25]	results of [24]
subject 4 (100)	67.19	71.4	70	77.5	74.4
subject 5 (200)	61.8	60.57	60.5	56.4	56.1
subject 6 (300)	59.35	58.09	61	51.9	52.2
subject 7 (400)	93.39	97.13	97.5	93.4	95.6
subject 8 (500)	71.5	91.32	93	96.9	95.9
subject 9 (600)	79.52	85.94	80.5	87.8	89.1
subject 10 (700)	74.57	76.98	78	80.6	72.2
subject 11 (800)	91.61	91.96	92.5	80	89.1
subject 12 (900)	83.76	84.53	87	78.8	82.8
Average	75.85	79.77	80.0	78.14	78.6
Average Rank	3.8	2.6	2.1	3.2	3.3

Table 3.7: Accuracy of our algorithms compared to the results reported in the literature for dataset IV2a. Highlighted cells show the best performing algorithm. The text inside the parenthesis corresponds to the original label of the subjects used in the BCI competition.

	BO-GP(AVG) (BP)	BO-RF(AVG) (Morlet)	results of [4]	results of [7]
subject 13 (1)	79.65	82.12	76	80.5
subject 14 (2)	47.33	44.86	56.5	53.5
subject 15 (3)	73.3	86.6	81.25	79
subject 16 (4)	61.25	66.28	61	62.5
subject 17 (5)	35.56	48.72	55	44.5
subject 18 (6)	55.45	53.3	45.25	50.5
subject 19 (7)	77.64	72.64	82.75	76.75
subject 20 (8)	80.52	82.33	81.25	78.25
subject 21(9)	81.15	76.35	70.75	82
Average	65.76	68.13	67.75	67.5
Average Rank	2.9	2.0	2.5	2.6

for each dataset separately. The Last row in Tables 3.5, 3.6 and 3.7 show the average ranking of different algorithms in datasets III3b, IV2b and IV2a, respectively. In all datasets, the Friedman's statistical test did not reject the null hypothesis. The p-values for the Friedman's test in datasets III3b, IV2b and IV2a were 0.280, 0.138 and 0.471, respectively. These results suggest that there is no statistical difference between the best performing results in the literature and our results.

To be able to draw conclusions about the performance of different algorithms, we have also compared the mean accuracy of different methods. In datasets III3b and IV2a, the average accuracies of our algorithms are 2.78% and 0.38% better than the results reported in the literature, respectively. In dataset IV2b, the mean accuracy of [4] is 0.23% better than our algorithms. It is also important to mention that in both datasets III3b and IV2a, the average rank of our algorithm is considerably better than the others.

While our proposed algorithm yielded similar or better performance compared to the literature, it is important to note that the best reported results in the literature were achieved based on manual fine-tuning of the hyper-parameters of the BCI system and based on more sophisticated feature extraction and classification methods. Our results show that our automated parameter optimization of a BCI system with less sophisticated feature extraction and classification methods yields similar/superior results compared to best performing designs in the literature.

3.4.3 Effect of Number of Optimization Iterations

To gain more insights about the effects of the number of optimization iterations (i.e., the number of classifiers) on the BCI performance results, we have conducted an experiment to observe the cross-validation accuracy and test set accuracy with different numbers of optimization iterations ranging from T=1 to T=80 iterations.

For subject 1 of dataset IV2a, Figure 3.2 shows the cross-validation accuracy and the

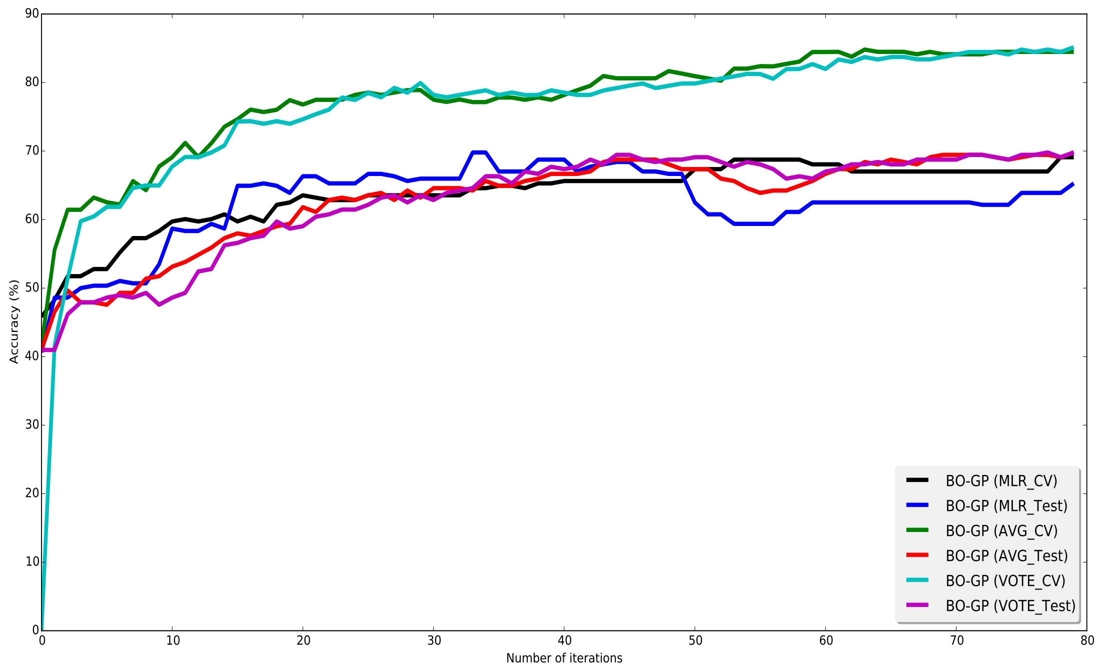

Figure 3.2: Accuracy of our algorithm with the BP features versus the number of iterations. The label of cross validation curves finish with '_CV' and the label of test data accuracies finish with '_Test'.

test set accuracy of our algorithm as a function of the number of optimization iterations with BP features. Figure 3.3 repeats the same but with the Morlet features.

As shown in these figures, as the number of iterations increases the cross-validation accuracy eventually reaches a plateau. The termination condition in our algorithm checks when the cross-validation accuracy stops improving and then stops the optimization process. In the case of MLR algorithm, if we do not stop the optimization process, we start to see some signs of over-training i.e. the performance on the test set declines compared to the cross-validation accuracy. This happens after about 25 iterations in the Morlet features and 50 iterations in the BP features. We did not observe any signs of over-fitting with other methods that combine the classifiers (i.e., AVG and VOTE). However, to decrease the risk of over-training, we have used the termination condition described in Section 3.3.4 rather than using a fixed number of iterations which is prone to over-fitting.

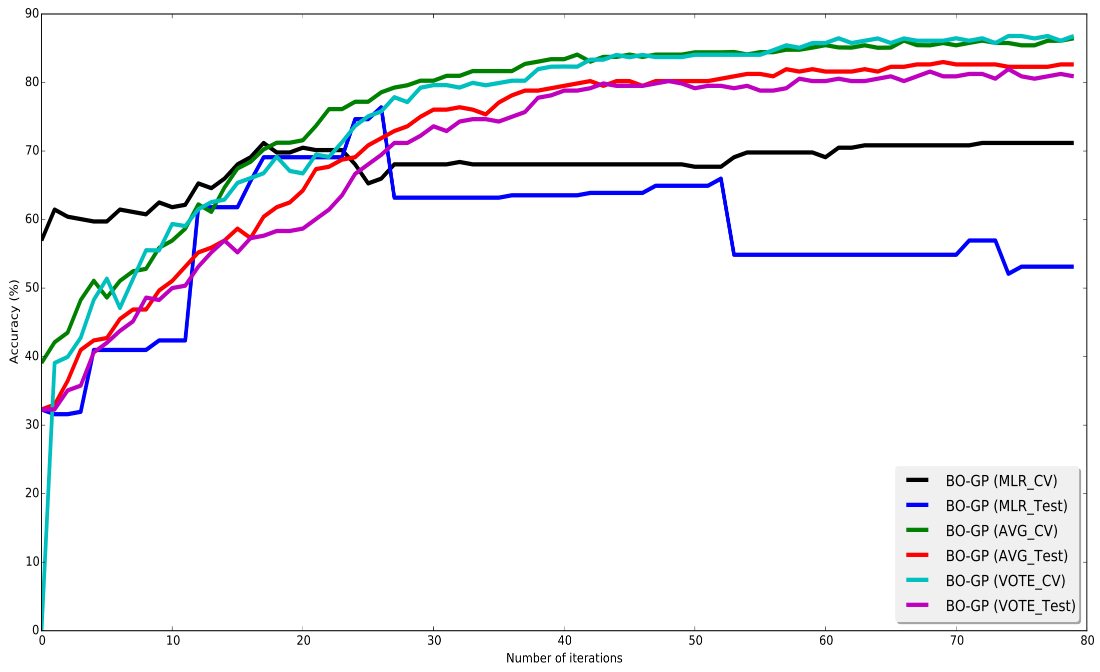

Figure 3.3: Accuracy of our algorithm with the Morlet features versus the number of iterations. The label of cross-validation curves finish with '_CV' and the label of test data accuracies finish with '_Test'.

We performed the same analysis for all the other subjects (in all datasets) and observed the same results as subject 1 of dataset IV2a.

3.5 Conclusion

Customizing the parameters of a BCI system is a necessary step to capture each subject's unique brain signal characteristics. Such characteristics are captured by a collection of parameters of the different blocks of a BCI system, and form the hyper-parameters of a BCI system. These hyper-parameters are either tuned manually based on the operator experience (which is very cumbersome and not optimal); or in a semi-automated approach.

In this chapter, we proposed a fully automated optimization-based approach for tuning the hyper-parameters of a BCI system for each subject with the goal of achieving a user-customized BCI. To be more specific, we demonstrated the utility of Bayesian optimization with Gaussian process and random forest posteriors to tune the parameters of a motor-imagery BCI. The BCI was based on the band-power and the Morlet wavelet features and Logistic Regression classification followed by ensemble classification. In our framework, we treated each of the classifiers (built upon each set of hyper-parameters) as a member of the ensemble of classifiers. This is to harness as much information as possible since each hyper-parameter set captures specific characteristics of the brain signals.

The results of comparison of the performance of different algorithms on 21 subjects in the previous section, showed that, when the BP features were used, Bayesian optimization with GP posterior and averaging (BO-GP (AVG)) outperform other methods. When the Morlet features were used, the proposed Bayesian optimization with RF posterior and averaging (BO-RF (AVG)) yielded better performance. In addition to providing a framework for automated tuning of the parameters of a BCI system for each subject, our findings confirm that customizing a BCI for each subject has a major role in improving the performance of the system.

More specifically, using our proposed approach, the mean accuracies improved by 8.12% and 9.19% when the band power and the Morlet features were used in the BCI systems respectively. These results highlight the importance of parameter optimization and utility of our approach in finding optimal parameters of a BCI system. We have also backed our results with proper statistical tests.

We have compared the performance of the optimized BCIs with those reported in the literature and have shown that hyper-parameter optimization leads to similar or better performance compared to the literature. To be more specific, in dataset III3b, our results were better than the ones reported in the literature. The average accuracy before optimization was 71.16% for the Morlet features. After applying our proposed algorithm, the average accuracy is boosted to 83.88%. The best reported method in the literature has an average accuracy of 81.1%. In dataset IV2b, the average accuracy across all subjects using the hyper-parameter values usually used in the literature was 77.02%; our algorithm boosted it to 79.77%. The average accuracy of the best reported results in the literature was 80%. For dataset IV2a, the average accuracy for the Morlet features before optimization was 46.1% and after optimization it became 68.13%. The competition winners average accuracy across all subjects was 67.75%.

We demonstrated the utility of our proposed optimization-based approach to customize the parameters of two BCI systems (based on the BP and the Morlet features) that perform motor-imagery tasks. The proposed framework however can be applied to any BCI system that is based on other neurological phenomena and to any combination of other feature extraction and classification schemas.

Our results show that applying a good hyper-parameter selection algorithm can boost a simple baseline algorithm's performance to enable it to compete with the state of the art BCIs. It is also important to mention that our algorithm can also be applied to any BCI design irrespective of the number of parameters that need to be optimized. Our algorithm is easy to apply, fully automatic, inexpensive and does not need expertise in EEG. In addition, unlike similar works in the field, our algorithm is scalable and can tune any number of hyper-parameters in a BCI system.

Chapter 4

Discriminative Sequence Learning Algorithms for Self-paced BCIs

4.1 Introduction

As we described in chapter 1, BCI systems can be classified into two categories: synchronous and self-paced systems [9]. In synchronous BCIs, subjects control the BCI output during short periods specified by the system, while self-paced BCIs, give users the option of controlling the system whenever they wish to. For the latter type of BCIs, the periods during which the user is controlling the system are called Control states and those during which the user is not controlling the system are called No-Control (NC) states.

Designing a self-paced BCI which is ultimately the more natural way to operate BCIs is extremely challenging compared to synchronous BCI systems. The task of classifying brain signals in a self-paced BCI is formulated as a sequential supervised learning problem [34]. In sequential learning, a sequence of observations and their corresponding labels are known, and the goal is to construct a classifier that predicts the sequence of the labels of a new sequence of observations.

The most popular way to obtain observation sequences for sequential supervised learning is to use a sliding window over the signal (these windows might overlap). This approach takes consecutive input windows of the EEG brain signal, each of length *w* milliseconds, extracts features from each window and assigns a label to each window. The assigned label corresponds to the intention of the user, as to whether or not he/she wants to operate the device. The sequence of the extracted feature vectors and their corresponding labels are used to train a classier. For a new sequence, the trained classifier estimates the label of each window (i.e. the intention of the user).

The advantage of using the sliding window approach is that it converts the sequential supervised learning problem into a standard classification problem. Thus any classical supervised learning method could be used to solve the problem. The majority of the publications in the field have used this approach to build different self-paced BCIs [8]. The disadvantage of the sliding window approach is that the sequential correlation in the labels of consecutive EEG windows is not exploited. The observations and the labels of nearby windows are usually related to each other, and it is thus important to use the information about sequential correlation between adjacent EEG windows.

In [77] the authors show that the brain goes through several well-defined internal state changes while the subject is carrying out a motor imagery mental task. Utilizing an algorithm that can model/exploit these state transitions can enhance the performance of the BCI. The class of sequence labeling algorithms has been used to exploit the temporal structure of the EEG data. Among these algorithms the most popular is the Hidden Markov Model (HMM) [81] which is a generative classifier. A generative classifier models the joint probability of observations and label sequences. Although the HMM classifier has been successful in synchronous BCIs, in [27] the authors concluded that the sliding window approach (i.e. using classical classifiers) is superior to HMM in self-paced BCIs. Due to intractability issues, HMMs assume the observations are independent given the states.

This makes it difficult to incorporate the knowledge about the structure of the EEG data into the model by extracting informative overlapping observations.

Another type of sequence labeling classifiers (besides HMMs) are the discriminative classifiers. These classifiers directly maximize the conditional likelihood of the label sequence given the observations. These algorithms have yielded very promising results in the natural language processing [74], and activity recognition [100] fields (which have very similar nature to the task of self-paced classification of BCIs). The advantage of these models is that they give the user the freedom to extract many informative and overlapping features from the observation sequence [1]. These features might be extracted from previous windows of the brain signal and may be correlated.

In [46] and [86], the authors applied discriminative sequence labeling algorithms to self-paced BCIs to classify different motor imagery tasks. We, on the other hand, use discriminative sequence labeling algorithms to discriminate between NC and control states. As discussed above, during the NC periods the subject's brain can be in any state therefore it is extremely difficult to find specific patterns in the signal. This makes the discrimination between NC and control states more difficult compared to the discrimination of different movement imagery tasks.

In this chapter, we propose a new discriminative sequence labeling classifier to solve the self-paced BCI problem. Our method combines the power of one of the most popular discriminative sequence labeling classifier, i.e. Conditional Random Fields (CRF) [60] to exploit the correlation in consecutive EEG windows. It also utilizes a neural network to extract high-level features from the observations. We call our method Neural Network Conditional Random Fields (NNCRF). Using NNCRF the observation vector is passed through a multi-layer neural network. The neural network applies several layers of nonlinear transformation on the observations vector. Then, the output of the neural network is given to a

[1]These observations correspond to the features extracted from the raw EEG signal.

CRF classifier. It is also important to note that the parameters of the CRF and the neural network parts are learned jointly.

4.2 EEG Sequence Labeling with Hidden Markov Models

Hidden Markov Models directly model the joint distribution of observations and labels [81]. The non-stationary nature of EEG signals makes it difficult for the HMM to model the observations. As this method is a generative learning algorithm, defining the emission distribution is difficult when the feature space is of high dimensions. In such cases the number of parameters is too large and this might lead to over-fitting. Therefore it is common to use simplifying assumptions, e.g. conditional independence of the elements of the observation vectors given the labels. This however makes this technique less powerful. The emission function we used here is the mixture of Gaussian distributions.

To apply HMMs on self-paced BCIs, we trained different HMMs for the different mental tasks. We trained two different HMMs, one was trained using NC data and the other one using the samples of the movement (Control) task. Then the likelihood of a new given sequence was calculated using the forward-back algorithm and the classification was performed by comparing the likelihoods of the different HMMs. In this case, each HMM focuses on learning the structure of the mental task that it is trained on, instead of learning to discriminate between different tasks.

The emission function used for HMM is the mixture of Gaussian distributions. The parameters of the HMM include the transition probabilities, and the parameters of the mixture of Gaussian distributions. The parameters are learned by maximizing the likelihood of the training dataset. Baum-Weltch algorithm is used to learn the parameters of the HMM model [12].

4.3 Discriminative Sequence Labeling Algorithms

For sequence labeling algorithms, each data sample in the training set consists of a sequence of observations[2], and its corresponding label sequence. Assuming the training set is $\{x_i, y_i\}_{i=1}^{N}$ where N is the number of training samples. For a training sample there are K windows i.e. x_i is the sequence of observation vectors from K consecutive windows and y_i is the corresponding sequence of K labels. As in the sliding window approach, every w milliseconds of the EEG creates a window. x_i is created by concatenating the vectors of observations of each of K consecutive windows i.e. $x_i = [x_{i1}, ..., x_{ik}, ..., x_{iK}]$ and x_{ik} corresponds to the vector of observations in the k_{th} window in the sequence number i [3]. Likewise, y_i is created by concatenating the labels of each of these K consecutive windows i.e. $y_i = [y_{i1}, ..., y_{ik}, ..., y_{iK}]$ and y_{ik} corresponds to the label of the k_{th} window in the sequence number i.

Discriminative sequence labeling classifiers directly maximize the conditional likelihood of the label sequence given the observations. These models make it possible to extract many informative and overlapping features from the observation sequence. Furthermore, these models have the ability to represent long range dependencies between observations. The conditional nature of these models gives them the freedom to relax the independence assumptions made by HMM.

The power of these methods comes from their ability to design a set of features based on the structure of the data. Each feature is a function of the joint observations and label pair. In classical classification the feature vector is built based on the observations (X) only. However, here the idea is to extract features from the joint observations (X) and label (Y) spaces. In this way, each feature function Φ measures the compatibility of x (x_i) and y (y_i) [4]. Although more complex types of features can be used, in all the following discriminative

[2]These observations correspond to the features extracted from the raw EEG signal.

[3]In our case, x_{ik} corresponds to the bandpower of the window in a specific frequency band.

[4]For the sake of clarity, we drop the index i in the remainder of the text

sequence labeling classifiers the feature functions are inspired by a first order HMM.

In the following, we breifly describe three different discriminative sequence labeling classifiers, the Hidden Markov Support Vector Machines (HMSVM), the Conditional Random Fields (CRF) and the Neural Networks Conditional Random Fields (NNCRF).

4.3.1 Hidden Markov Support Vector Machines

Sequential supervised learning can be modeled as a special case of structured learning in which both observations and labels are sequences. In structured learning problems, the output of the classifier has a particular structure. It is possible to solve a structural learning task using any multi-class classifier. However, in this case the internal structure of the output is not exploited and the number of classes might be very large which makes using these algorithms almost impossible. Structural Support Vector Machine (SSVM) [98] is a generalization of multi-class SVMs which allows us to build classifiers for structured data (such as sequence of observations and labels). SSVM has the advantage of being a large margin classifier like SVM.

A Hidden Markov Support Vector Machine [2] is a special case of SSVM in which the features are designed to capture the sequential nature of the data. The set of feature vectors we use, capture the dependency between consecutive labels in a sequence (y_k and y_{k-1}), and measure the relation between the observation in the k_{th} window (x_k) and its corresponding label y_k in a sequence.

The first set of features captures the dependency between two consecutive labels in a sequence:

$$\begin{aligned}\phi^{\sigma_1\sigma_2}_{k(k-1)} &= \phi(y_k, y_{k-1}) \\ &= I(y_k = \sigma_1 \wedge y_{k-1} = \sigma_2),\ \sigma_1, \sigma_2 \in \Sigma,\end{aligned} \tag{4.1}$$

$I(.)$ denotes the indicator function which has the value 1 when the input is $true$ and 0 otherwise. Σ corresponds to the set of possible values of the label of each window i.e.

movement imagery or NC state. The vector $\phi_{k(k-1)}$ is formed by stacking $\phi_{k(k-1)}^{\sigma_1\sigma_2}$ elements for all possible values of σ_1 and σ_2.

The second type of features measures the relation between the observation token x_k and its corresponding label y_k in the sequence:

$$\phi_k^{\sigma} = \phi(x_k, y_k) = I(y_k = \sigma)x_k, \sigma \in \Sigma, \tag{4.2}$$

where x_k represents the observations from the k_{th} window of the sequence. The vector ϕ_k is formed by stacking ϕ_k^{σ} for all possible values of σ.

By concatenating the two type of feature vectors, we obtain a single vector of all features for the k^{th} window. Therefore, the whole set of features for the k^{th} window ($\phi(x,y,k)$) is equal to $[\phi_k, \phi_{k(k-1)}]$. To create the final set of features $\Phi(x,y)$ for the sequence, we sum the feature vectors over the sequence:

$$\Phi(x,y) = \sum_{k=1}^{K} \phi(x,y,k). \tag{4.3}$$

Therefore, $\Phi(x,y)$ for type one features corresponds to the number of transitions between different labels. For type two features, it is the sum of the band-power of the consecutive windows in a sequence for each possible label. These types of features are very similar to the maximum likelihood solution of an HMM.

As in [2], we have used a linear function of the observation/label pair for classification of the brain signals. Given a new unseen sequence x_{N+1} the goal is to predict its corresponding sequence of labels. For inference i.e. assigning a sequence of labels to x_{N+1} we maximize the following equation:

$$y_{predicted} = \underset{y}{\operatorname{argmax}}\, W^T \Phi(x_{N+1}, y), \tag{4.4}$$

$W^T\Phi(x,y)$ is a linear function defined on the joint space of observation/Labels sequences pair. Like HMMs, the solution of Equation 4.4 is found using the Viterbi algorithm.

For the training phase i.e. finding an optimal value of the vector W, SSVM solves the following minimization problem:

$$\begin{aligned}
&\min_{W,\xi} \frac{1}{2}||W||^2 + C\sum_{i=1}^{N} \xi_i, \\
&\text{s.t. } \forall j \in [1,N], \forall y \in Y, \\
&W^T\Phi(x^j,y^j) - W^T\Phi(x^j,y) \geq \Delta(y^j,y) - \xi_j,
\end{aligned} \tag{4.5}$$

where $\Delta(y^j,y)$ is the loss function that measures the penalty of estimating a wrong label for the sequence. This maximization problem is a convex problem however, it involves a large number of linear inequality constraints. These constraints make the distance between the true label of a sequence and all other possible values of the sequence label maximum [97].

4.3.2 Conditional Random Fields

The Conditional Random Field (CRF) is another discriminative classifier which is well suited for sequential supervised learning. In general the linear chain Conditional Random Fields classifier models the posterior distribution of the form

$$\Pr(Y|X) = \frac{\exp(\sum_{j=1}^{J} \lambda_j \sum_{k=1}^{K} \Phi_j(y_{k-1},y_k,x,k))}{Z(X)}, \tag{4.6}$$

where K is the sequence length, J is the number of features extracted from the joint observation labels pair, and Φ_j is the feature function. λ_js are the parameters of the model which are learned based on the training data. $Z(x)$ is the normalization factor (partition function):

$$Z(X) = \sum_{y'_1}\sum_{y'_2}\dots\sum_{y'_K}\exp(\sum_{j=1}^{J}\lambda_j\sum_{k=1}^{K}\Phi_j(y'_{k-1},y'_k,x,k)) \tag{4.7}$$

To calculate the partition function we use forward-backward algorithm which is a dynamic programming algorithm.

The set of feature functions (ϕ_j) should be specified before using CRF. The first feature we used, is defined as

$$\phi_0^{\sigma} = I(y_k = \sigma),\ \sigma \in \Sigma, \tag{4.8}$$

where I represents the indicator function, and Σ corresponds to the set of possible label values (i.e. NC states and movement states).

The second set of features $\phi_{k,1:L}$ captures the relation between the observation vector and the k_{th} label in the sequence. We are assuming the observation vector is of dimension L. Each element $\phi_{k,l}^{\sigma}$ of the vector $\phi_{k,1:L}^{\sigma}$ is defined as

$$\phi_{k,l}^{\sigma} = x_{k,l}I(y_k = \sigma),\ \sigma \in \Sigma, \tag{4.9}$$

where $x_{k,l}$ is the l_{th} dimension of the observation vector.

The third type of features is defined as

$$\phi_{k(k-1)}^{\sigma_1,\sigma_2} = I(y_k = \sigma_1 \wedge y_{k-1} = \sigma_2),\ \sigma_1,\sigma_2 \in \Sigma, \tag{4.10}$$

where $\phi_{k(k-1)}^{\sigma_1,\sigma_2}$ captures the relation between the k_{th} and $(k-1)_{th}$ labels in the sequence.

Using the above mentioned set of features (equations 4.8, 4.9 and 4.10) the posterior function will be of the form

$$\Pr(y_{1:K}|x_{1:K}) = \frac{\exp\left(\sum_{k=1}^{K}(b_{y_k} + W_{y_k,1:L}^T x_k) + \sum_{k=2}^{K} V_{y_{k-1},y_k}\right)}{Z(X)}, \tag{4.11}$$

where b_{y_k}, $W_{y_k,1:L}$ and V_{y_{k-1},y_k} are the parameters of the posterior distribution and should be learned using the training dataset. The set of parameters b_{y_k}, $W_{y_k,1:L}$ and $V_{y_k,y_{k-1}}$ captures the importance of the first, second and third type of features respectively.

To learn the optimal values of the parameters, we minimize the L_2 regularized negative log likelihood

$$\min_{\lambda} \sum_{i=1}^{N} -Log \Pr(y_i|x_i) + C\lambda^T\lambda, \tag{4.12}$$

where the vector λ consists of the parameters of the model (i.e. b_{y_k}, $W_{y_k,1:L}$ and V_{y_{k-1},y_k}), and C is the regularization coefficient. To find the optimal value of the parameters, we use the stochastic gradient descent algorithm. As the loss function is convex, the gradient descent algorithm converges to the global optimum of the loss function.

For the inference i.e. assigning a sequence of labels to x_{N+1}, we assign the most probable sequence as the predicted labels i.e.

$$y_{predicted} = \operatorname*{argmax}_{y} P(y|x_{N+1}). \tag{4.13}$$

We used a Viterbi like algorithm to find the most probable sequence of labels.

4.3.3 Neural Network Conditional Random Fields

The linearity of the exponent term in CRF classifiers makes them have less expressive power compared to the models that exploit kernels. A good approach that makes these algorithms more powerful is to combine feed-forward neural network with CRF. Neural network transforms the observation vector into high level features which are then used as the input to the CRF. As a result, the exponent term in CRF (i.e. the exponent term in equation 4.11) becomes non-linear because of the several layers of non-linear activation functions that have been applied on the observation vector [35] [76].

Neural Network Conditional Random Field (NNCRF) can be viewed as a standard linear chain CRF that uses a high level representation of the observations. Therefore, the posterior function of the NNCRF has the same form as that in equation 4.11 except that the x_k terms are replaced with the $h^M(x_k)$ term which represents the output of a feed-forward neural networks with M layers.

$$\Pr(Y|X) = \frac{\exp(\sum_{k=1}^{K} b_{y_k} + W^T_{y_k,1:L} h^{(M)}(x_k)_{y_k} + \sum_{k=2}^{K} V_{y_{k-1},y_k})}{Z(X)}. \tag{4.14}$$

The output of the $(M)_{th}$ layer ($h^{(M)}$) is

$$h^{(M)}(x_k) = tanh(b^{(M-1)} + W^{(M-1)} h^{M-1}(x_k)), \tag{4.15}$$

where $b^{(M-1)}$ and $W^{(M-1)}$ are the weights of the $(M-1)_{th}$ layer, $h^{M-1}(x_k)$ is the output of the $(M-1)_{th}$ layer of the neural network, and $h^0(x_k) = x_k$. To avoid over-fitting, the weights of the neural networks are the same for all observations x_k in a sequence i.e. the values of the weights do not depend on k.

Initialization of a neural networks is very crucial for this algorithm to converge to a good local optima. For initialization of the neural network, the value of each parameter is a sample taken from a uniform distribution $U[-\gamma, +\gamma]$ where

$$\gamma = \frac{\sqrt{6}}{\sqrt{size(M) + size(M-1)}}, \tag{4.16}$$

where $size(M)$ is the number of hidden units in the M_{th} layer of the neural networks.

The loss function is the same as the loss function of the standard CRF (Equation 4.12). We use the stochastic gradient descent algorithm to jointly optimize the values of the parameters of the neural networks and of the CRF. The learning rate of the stochastic gradient descent algorithm is adjusted using the bold driver approach. For inference the same algorithm as in standard linear chain CRF is used.

4.4 Datasets

To perform the experiments two self-paced sensory motor BCI datasets have been used. The first dataset, SM2, was collected from 4 subjects attempting to activate a switch by performing a right index finger movement (as explained in section 2.3).

The second dataset, BCICIV2a, is the dataset IIa from the BCI competition IV (Section 2.3) which is recorded from 9 subjects performing 4-class motor imagery (left hand and right hand, both feet and tongue imagery movements) tasks. We have treated this dataset as a self-paced BCI dataset. In other words, to evaluate the performance of the classifiers on this dataset the time of transition from previous mental task to the new one (the time cue was displayed) has not been used. We have also converted the problem into a binary classification task i.e. separating movement imagery from NC states. All 4-classes of motor-imagery are considered as movement (control states) and the periods in which the subject did not control the system are considered as the No-Control class.

4.5 Results

We compared the performance of NNCRF with the standard CRF classifier, Hidden Markov Support Vector Machines (HMSVM), Hidden Markov Models (HMM) and two popular classical classifiers, Logistic Regression and Support Vector Machines (SVM).

For frequency filtering, we applied a filter-bank with 2 blocks in ranges [8-12]Hz and [16-24]Hz corresponding to the typical ranges of the alpha and beta brain waves. For spatial filtering, we used CSP with 2, 4 and 6 filters. The values of these hyper-parameters along with each classifier's parameters are adjusted jointly using 5-fold cross-validation. Then the parameters with the best mean cross-validation accuracy were used to train a classifier on the training set. To evaluate the performance of the classifiers on the test dataset, we used the Area Under the Curve (AUC) measure.

In our experiments, the band power of the EEG signal is used as the extracted features (observations) for the classification phase. The window length was equal to the sampling rate of the dataset and we used the last two seconds of the data to perform the classification in the test phase.

Table 4.1 shows the results of comparing different classifiers. We also included the average rank of each algorithm in Table 4.1. The average ranks of different classifiers are calculated by ranking them based on their AUC for each subject separately, and then averaging the ranks over all subjects.

Table 4.1: The results of comparing AUC of different algorithms on the test dataset. Highlighted cells show the algorithm for which the performance is the best. The last row shows the average rank of different classifiers across all subjects.

Subject	LR	SVM	HMM	HMSVM	CRF	NNCRF
1	0.617	0.472	0.470	0.567	0.554	0.565
2	0.605	0.535	0.577	0.602	0.640	0.681
3	0.666	0.659	0.646	0.643	0.690	0.691
4	0.673	0.618	0.556	0.593	0.594	0.540
5	0.559	0.560	0.511	0.510	0.473	0.493
6	0.679	0.653	0.484	0.629	0.692	0.701
7	0.725	0.710	0.605	0.683	0.672	0.666
8	0.709	0.706	0.509	0.542	0.629	0.718
9	0.616	0.606	0.527	0.565	0.591	0.588
22	0.767	0.723	0.711	0.708	0.789	0.840
23	0.724	0.728	0.762	0.706	0.828	0.798
24	0.610	0.604	0.548	0.580	0.615	0.638
25	0.585	0.574	0.604	0.588	0.669	0.707
Average Rank	2.54	3.62	5	4.54	2.85	2.46

As shown in Table 4.1, the average rank of NNCRF is better than those of other algorithms, the second best classifier is the LR classifier. Another observation is that HMM has the worst performance, which suggests that HMM has no advantage over classical classifiers (this observation has also been reported in [27]). It is worth mentioning that for some subjects none of the discriminative sequence labeling classifiers have a good performance, this means that these classifiers are not able to capture the temporal structure of the signal in these subjects.

4.6 Conclusion

In this chapter, we proposed a discriminative sequence labeling algorithm (classifier), to capture the dynamics of the EEG signal. We evaluated the performance of our algorithm (which we denote as NNCRF) on two self-paced BCI datasets and showed that it is superior, compared to classical classifiers and sequence labeling classifiers. NNCRF is a combination of a neural network and a CRF classifier. We demonstrated that CRF and NNCRF can capture the temporal properties of the EEG signal and improve the accuracy of the BCI in most of the subjects. In some subjects however, the temporal structure of the EEG data is difficult to capture by these classifiers. The neural network part of NNCRF converts the original observation vector into a new representation which is then fed to the CRF part. The non-linear transformation (by the neural network) of the observation vector helps the CRF part to discriminate between the different control and NC easily.

Overall, the reason for the poor performance of classical approaches is that they do not exploit the inherent dynamics in the EEG signal. As for the HMM, although this algorithm models the temporal correlations in an EEG signal, its poor performance stems from the fact that it focuses on learning the mental task without learning to discriminate between the different mental tasks. On the other hand, discriminative sequence labeling classifiers do not only model the temporal properties of each mental task, they also model the transition from one mental task to another (e.g. transition from movement to NC state).

Chapter 5

Ensemble of Discriminative Sequence Labeling Classifiers for Self-Paced BCIs

5.1 Introduction

The aim of this chapter is to design a self-paced BCI that can be automatically customized for each user. In Chapter 3, we showed the importance of customizing synchronous BCIs based on the brain characteristics of each user. We then proposed an approach for customizing two different kinds of synchronous BCIs based on band-power features and Morlet features with logistic legression classifier. Here, we adapt the same approach to customize self-paced BCIs (proposed in chapter 4) based on the brain characteristics of each user. So, our algorithm not only takes the correlation between consecutive EEG windows into account, but also is customized based on the characteristics of each subject's EEG signal.

Customizing a BCI for each subject involves optimizing a cost function over a hyper-parameter space. The cost function used here is the cross-validation accuracy of the BCI on the training data, and the hyper-parameters are the parameters of the BCI system which are selected before feature extraction and classification. We use the algorithm we proposed in chapter 3 which is an iterative algorithm that optimizes the values of the hyper-parameters. After the optimization is finished, we can either use the best performing classifier that our classifier selected in the optimization phase or we can combine all the trained classifiers into one final classifier.

In general, for our final classifier we seek a model with low bias and high variance as our final classifier. When the training data is small but the classifier has high variance, then by averaging the results of individual classifiers we reduce the variance of the final classifier while preserving the low bias of a single model. Especially in the case of neural networks which can get stuck in a local optimum and has a high variance, creating an ensemble of neural networks which are trained on different parts of the hyper-parameter space, may result in a better approximation of the best possible classifier.

In self-paced BCI, the hyper-parameter space of every classifier consists of the frequency ranges over which the EEG signal is filtered, the selected channels (from which the features are extracted), and the window length (w). For the first (i.e. frequency ranges) and second (i.e. channels) types of hyper-parameters, we use the same settings as in section 3.3.1. The third hyper-parameter is the length w of the sliding window. For all the classifiers, we have used the past two seconds of the EEG signal to build the observation vector from. For classical classifiers, we only look at the past w milliseconds of the data i.e. only the last window. For sequence labeling classifiers, we build a chain of consecutive windows as our observation vector. The length of the chain is therefore $\frac{2}{|w|} \times 1000$. For the proposed NNCRF classifier which we proposed in chapter 4, we have an additional hyper-parameter which is the number of neurons in the hidden layer of the neural network. We optimize all these hyper-parameters jointly.

Using Bayesian optimization, we search through the hyper-parameter space to find the best hypothesis (classifier) that makes a good prediction. Bayesian optimization iteratively proposes different points (values) in the hyper-parameter space. Each of these points are

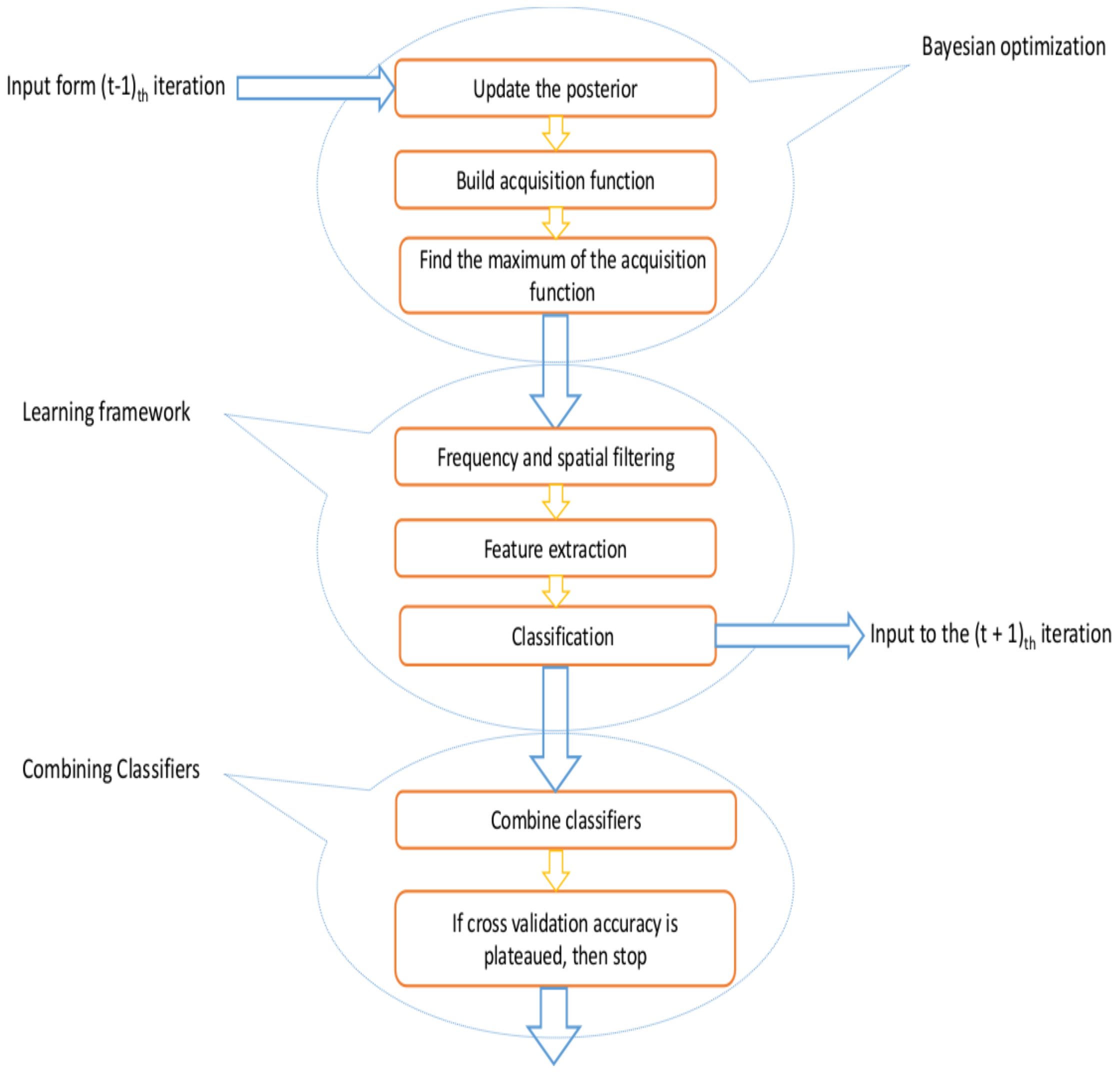
Bayesian optimization
Input form $(t\text{-}1)_{th}$ iteration
Update the posterior
Build acquisition function
Find the maximum of the acquisition function
Learning framework
Frequency and spatial filtering
Feature extraction
Classification
Input to the $(t+1)_{th}$ iteration
Combining Classifiers
Combine classifiers
If cross validation accuracy is plateaued, then stop

Figure 5.1: Diagram of our optimization algorithm at iteration t.

used to train a classifiers. After the optimization is finished, we can either select the best performing classifier obtained in the optimization phase as our final classifier or we can combine all trained classifiers into one final classifier.

To make our algorithm clear, we have added the diagram (Figure 5.1) of our algorithm that elaborates one iteration of our algorithm. In each iteration of our algorithm, we first update the posterior distribution. Then based on the posterior distribution, we build the acquisition function and find the maximum point of the acquisition function.

The maximum point (proposed candidate by the Bayesian optimization) is used to perform filtering on the EEG. The features are then extracted and a classifier is trained on the training samples. We then, calculate the accuracy of the classifier, this will be used in the next iteration of our algorithm. In each iteration of our algorithm, we combine the classifiers built in the previous iterations, and examine the stopping condition to decide whether to stop the algorithm or continue to the next iteration.

5.2 Results

The set of classifiers and the feature extraction we used are the same as those of chapter 4. The datasets are also the same and are explained in 4.4. For the experiments, we compared the ensemble of the classifiers generated in the optimization phase.

Table 5.1 shows the results of comparing different classifiers when we used ensemble of classifiers. The columns with the MIN label correspond to the results of applying the best performing classifier obtained in the optimization phase on the independent test dataset. The Bayesian optimization algorithm was run for at most 50 iterations or until cross-validation accuracy plateaued. We have repeated our experiments five times to reduce the effect of the random seed. The average number of iterations of the Bayesian optimization algorithm for all different classifiers was 27. In Table 5.1, the columns with AVG label correspond to the results of using all the generated classifiers in the optimization phase. To create the ensemble of classifiers, we averaged the results of all individual classifiers.

Table 5.1: The results of comparing different classifier after performing Bayesian optimization on the test dataset. MIN is the results of applying the best performing classifier in optimization phase on the test data. AVG corresponds to the results of the ensemble of classifiers. Highlighted cells show the algorithm for which the performance is the best.

	LR		SVM		HMM		HSVM		CRF		NNCRF	
Subject	MIN	AVG	MIN	AVG	MIN	AVG	MIN	AVG	MIN	AVG	MIN	AVG
1	0.677±0.001	0.708±0.003	0.612±0.027	0.684±0.008	0.571±0.001	0.572±0.003	0.626±0.008	0.682±0.004	0.646±0.015	0.651±0.006	0.663±0.0	0.653±0.004
2	0.653±0.0	0.663±0.004	0.655±0.004	0.668±0.002	0.597±0.0	0.647±0.014	0.592±0.008	0.647±0.003	0.643±0.003	0.66±0.004	0.641±0.012	0.658±0.004
3	0.689±0.001	0.709±0.003	0.686±0.005	0.721±0.004	0.705±0.0	0.688±0.003	0.663±0.0	0.724±0.001	0.724±0.011	0.735±0.002	0.735±0.003	0.743±0.004
4	0.743±0.0	0.769±0.002	0.737±0.002	0.758±0.003	0.554±0.0	0.571±0.004	0.695±0.0	0.749±0.004	0.748±0.0	0.725±0.006	0.736±0.003	0.722±0.003
5	0.595±0.006	0.629±0.003	0.561±0.001	0.622±0.007	0.518±0.004	0.509±0.003	0.605±0.0	0.655±0.003	0.637±0.002	0.618±0.006	0.632±0.016	0.632±0.004
6	0.657±0.001	0.692±0.004	0.678±0.012	0.729±0.003	0.512±0.024	0.63±0.015	0.62±0.002	0.718±0.002	0.691±0.021	0.731±0.005	0.711±0.016	0.73±0.006
7	0.738±0.0	0.762±0.002	0.729±0.008	0.778±0.002	0.64±0.009	0.71±0.01	0.737±0.0	0.802±0.002	0.769±0.0	0.751±0.003	0.769±0.0	0.755±0.004
8	0.716±0.002	0.729±0.002	0.716±0.004	0.736±0.002	0.547±0.126	0.633±0.034	0.505±0.021	0.658±0.013	0.648±0.018	0.697±0.011	0.723±0.004	0.779±0.004
9	0.588±0.016	0.626±0.006	0.57±0.012	0.638±0.007	0.565±0.046	0.598±0.013	0.681±0.006	0.687±0.009	0.702±0.006	0.669±0.005	0.709±0.0	0.686±0.003
22	0.795±0.004	0.826±0.004	0.786±0.0	0.818±0.002	0.704±0.016	0.732±0.006	0.741±0.001	0.827±0.004	0.815±0.02	0.871±0.005	0.847±0.005	0.881±0.004
23	0.716±0.0	0.733±0.004	0.752±0.007	0.763±0.004	0.641±0.012	0.637±0.01	0.701±0.009	0.777±0.003	0.813±0.009	0.831±0.004	0.817±0.007	0.833±0.006
24	0.594±0.007	0.597±0.002	0.577±0.003	0.6±0.003	0.556±0.013	0.554±0.006	0.58±0.011	0.617±0.003	0.639±0.009	0.626±0.005	0.633±0.004	0.631±0.003
25	0.607±0.0	0.605±0.002	0.61±0.03	0.642±0.008	0.521±0.007	0.518±0.004	0.626±0.009	0.667±0.007	0.699±0.028	0.682±0.011	0.734±0.003	0.723±0.013
AVERAGE Rank	7.65	5.23	8.50	4.23	11.23	10.65	9.54	4.15	4.92	4.65	3.88	3.35

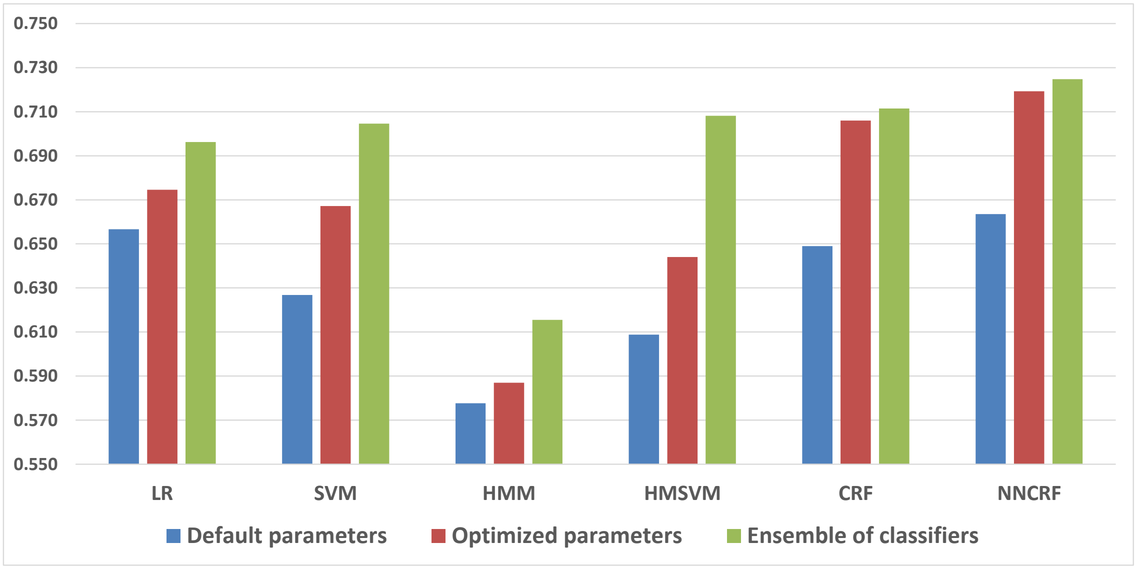

Figure 5.2: Average AUC of different settings for each classifier across all subjects. The blue bars correspond to the average AUC when the default value of the hyper-parameters have been used. The red bars correspond to the average AUC after using Bayesian optimization. The green bars correspond to the average AUC when we used an ensemble of different classifiers.

Qualitative comparison of different algorithms in Tables 4.1 and 5.1 suggests the following: 1) optimizing the values of the hyper-parameters improves the performance of any classifier considerably (comparing MIN columns of Table 5.1 with Table 4.1), 2) in almost all subjects and for almost all classifiers, using an ensemble of classifiers improves the performance, 3) HMM is the worst performing classifier, 4) NNCRF outperforms other algorithms in terms of average rank of the algorithm, and 5) interestingly, for some subjects none of the discriminative sequence labeling classifiers have yielded a good performance, this means that these classifiers are not able to capture the temporal structure of the signal in these subjects.

Figure 5.2 shows the average AUC of different algorithms across all subjects. The blue bars correspond to the average AUC when the default value of the hyper-parameters is used. The red bars correspond to the average AUC after using Bayesian optimization. The green bars correspond to the average AUC when we use an ensemble of different classifiers.

Figure 5.2 shows that for all classifiers using Bayesian optimization considerably improves the average AUC of classifiers across all subjects, and using an ensemble of the trained classifiers further improves the results. The best performing algorithm is the one that uses an ensemble of NNCRF classifiers.

5.3 Conclusion

To further improve the performance of our algorithms used in chapter 4, we customized the hyper-parameters of the BCI (using the algorithm proposed in 3). We evaluated different algorithms on the same subjects that we used in chapter 4. We showed that customizing the BCI hyper-parameters improves the performance of every classifier used in this study. After customizing the parameters of every classifier used, the best performing classifier was NNCRF, while CRF was the second best classifier.

We also showed that using an ensemble of classifiers that have been trained on different parts of the BCI hyper-parameter space can further improve the performance measure (AUC). We used Bayesian optimization to find the different values of the BCI hyper-parameters. Selecting different values for the hyper-parameters exposes the classifier to different parts of the BCI hyper-parameter space. We believe that this diversification is the key to the superiority of using the ensemble of classifiers compared to the single classifier approach. The performance of the classifiers is very sensitive to the choice of the hyper-parameters of the BCI and using an ensemble of classifiers can decrease the variance of the classifiers. The best performing algorithm was the ensemble of NNCRF classifiers.

Chapter 6

Conclusion and Future Work

6.1 Summary of Our Contributions

This thesis addresses the design of Brain Computer Interface (BCI) systems, in general, but specifically self-paced BCIs. In designing a self-paced BCI system that is based on sensory motor rhythms, it is important to exploit the properties of the brain signal. We should take into consideration the variations over time of the EEG signal in specific frequency bands. It is common to filter the data in these specific frequency bands during specific time segments and exploit the information embedded in the signal so as to gain maximum discrimination between different mental tasks. These frequency bands vary from one person to another. The optimal value of these frequency bands should be adjusted for each individual subject.

Also, the spatial information as to which channels should be used for a specific mental task has a major role in the performance of the BCI. All such information about the characteristics of the EEG signal is fed to the BCI system as a set of hyper-parameters. Thus the selection of the value of these parameters is very important as they determine the success and accuracy of the BCI. The value of all BCI hyper-parameters should be customized based on the brain characteristics of each subject. Furthermore, in sensory motor self-paced BCIs, while the subject is performing a mental task, his/her brain goes through several well-defined internal state changes. Therefore, it is important to use a classifier that is able to capture the dynamics of the EEG signal (i.e. takes the time course of the EEG signal into account).

In this thesis we study 1) how to automatically adjust the values of self-paced BCI parameters, and 2) how to exploit the time-frequency properties and user-specific characteristics of the EEG. Our final algorithm exploits the brain characteristics of the user in self-paced BCIs. Our algorithm has the following advantages, it is easy to apply to any BCI, it is fully automatic, it does not need EEG expertise, scalable in the number of hyper-parameters and computationally inexpensive. Moreover, in contrast to the general trend in BCIs our algorithm uses the sequential information in the EEG signal that further improves the performance of a self-paced BCI system.

We believe that our proposed system is an important step towards transitioning BCIs from research devices to in-home communication assistive devices. In the following, we will describe a summary of our contributions in each chapter of this thesis.

Chapter 2

In this chapter of the thesis, we developed an open source framework for comparing different BCI systems that is unique in two aspects. First, we performed a comprehensive comparison between 14 different sensory-motor-based BCI designs over 29 subjects. This is the first study that includes large number of designs and subjects. Secondly, we performed rigorous statistical tests to statistically validate the results. The strength of this study is that the source code and data of our experiments are openly available so that other researchers can (a) apply it on their own data and benchmark their data with standard methods, and (b) extend the framework to include other machine learning and signal processing methodologies.

Unlike most publications in the BCI field which recommend the linear discriminant analysis classifier as the best classifier, our findings show that for each feature extraction method many classifiers should be tried, and then the best classifier should be selected based on the cross-validation results of the classifiers. In general, we found that there is not a best classifier or best feature extraction method that outperforms all others. For each subject, the combination of the classifier, the features and model parameters should all be tuned together, and finally the method with the best performance on the training data set should be selected as the final model for testing on unseen data.

Chapter 3

In the third chapter of this thesis, we proposed an algorithm to automatically customize a BCI based on the brain characteristics of each user. The study was performed on 21 subjects in synchronous BCIs from the BCI competitions.

In our first set of experiments, we showed the importance of the joint tuning of hyper-parameters in BCIs. We showed that tuning the values of the BCI parameters using Bayesian optimization can considerably improve the mean accuracy across all 21 subjects. In the second part of the experiments, we compared our algorithms to several other approaches for tuning the values of the hyper-parameters. We also backed our results with the proper statistical tests.

Moreover, we compared the results of the proposed algorithm to the results reported in the literature on the same subjects. We have shown that our algorithm has at least similar performance, but usually outperforms the results reported in the literature.

A big advantage of our proposed method is that it is fully automatic. While our proposed algorithm yielded similar or better performance compared to the literature, it is important to note that the best reported results in the literature were achieved based on lengthy extensive manual fine-tuning of the parameters of the BCI system and based on more sophisticated feature extraction and classification methods. Our results show that our automated parameter optimization of a BCI system that relies on less sophisticated feature extraction and classification methods yields similar/superior results compared to the best performing designs in the literature.

Overall the results show the importance of selecting the appropriate values for the hyper-parameters of the system. Furthermore, these results show that applying a good hyper-parameter selection algorithm can boost a simple baseline algorithm's performance so it can compete with the state of the art algorithms and results.

Chapter 4

In chapter 4, we proposed an algorithm that can capture the time dynamics inherent in the EEG signal. The study was performed on 13 self-paced BCI subjects. The aim was to evaluate the performance of different algorithms in discriminating between a control task (movement or imagery movement) and no-control states. The main challenges in self-paced BCIs are that the start and end times of the control task and the No-Control (NC) state are not known, and it is extremely difficult to find the specific patterns in the NC states, this is because the subject can be in any mental state during the NC periods.

In chapter 4, we proposed a discriminative sequence labeling algorithm (classifier) to capture the dynamics of the EEG signal. The advantage of such classifiers is that they not only model the temporal properties of each mental task, but also model the transition from one metal task to another (e.g. transition from a movement to an NC state).

We evaluated the performance of our algorithm (which we denoted as NNCRF) and showed that it is superior, compared to classical classifiers and the sequence labeling classifiers. The proposed NNCRF is a combination of a neural network and a conditional random field classifier. The neural network part of NNCRF converts the original observation vector into a new representation which is then fed to the CRF part. The non-linear transformation (by the neural network) of the observation vector helps the CRF part to better discriminate between a control task and NC states easily. We demonstrated that the CRF and NNCRF can capture the temporal properties of the EEG signal and improve the accuracy of the BCI in most subjects.

Chapter 5

In chapter 5, we proposed an algorithm that customizes a self-paced BCI to each user i.e. exploits the brain characteristics of each subject. Our algorithm not only fine-tunes the (hyper-)parameters of the self-paced BCI, but also exploits to the time information specific to each user. The study was performed on 13 self-paced BCI subjects.

In this chapter, we showed that customizing the self-paced BCI hyper-parameters improves the performance of every classifier used in the study. We also showed that using an ensemble of classifiers (where every classifier has been trained on a different part of the BCI hyper-parameter space) can further improve the performance measure of the BCI. We used the algorithm proposed in chapter 3 to find the different values of the BCI hyper-parameters. Selecting different values for the hyper-parameters exposes the classifier to different parts of the BCI hyper-parameter space. We believe that this diversification is the key to the superiority of using an ensemble of classifiers compared to the single classifier approach. The performance of a classifiers is very sensitive to the choice of the hyper-parameters of the BCI and using an ensemble of classifier can decrease the variance of the classifiers.

The algorithm proposed in chapter 5, exploits the time and frequency characteristics of the specific user's brain in self-paced BCIs. We believe that our proposed system is an important step towards transitioning BCIs from research environment to real-life applications, this is because it is fully automatic, scalable and easy to use. Overall our results show the importance of selecting good values for the hyper-parameters, and at the same time exploiting the time information in self-paced BCI systems.

6.2 Future Work

Some possible future directions of this work are briefly discussed below:

In this study, we used linear chain discriminative sequence labeling classifiers for self-paced BCIs, however, using other more complex classifiers that can better capture the time correlation between different parts of the EEG signal could be beneficial. Another possible future direction is to use more advance feature functions for discriminative sequence labeling classifiers to incorporate more knowledge about the EEG. The type of the feature functions used in this study were all inspired by Hidden Markov Models. So they are all local in nature, i.e. each feature function only depends on the current or the previous label in the sequence. It is possible to use global features such as the ones that capture higher orders of dependency of the transition between consecutive labels, or feature functions that capture dependency between the EEG signal and labels of the EEG signal from distant past. These types of feature functions have the ability to capture more complex structures in the data and increase the power of classifier.

One of the major problems in BCIs is that the datasets are very small. Collecting more data will enable us to use deep learning algorithms [62]. Specially in our proposed NNCRF algorithm, having larger datasets would enable us to utilize a deep neural network which will eventually gives the algorithm the ability to extract better and more informative features from the EEG signal. The current trend in BCIs is to hand-craft feature extractors from EEG signal that result in high level features from EEG signal. On the other hand, by using deep learning algorithms feature extraction process becomes a part of the automated system which can lead to easier classification of the BCI data. In addition to NNCRF, another class of deep learning algorithms which can be used here, are recurrent neural networks. These algorithms have been very successful in different sequence labeling applications that are very similar in nature to the self-paced classification of BCIs [49] .

Non-stationarity of the EEG data makes real-life applications of (sensory-motor) BCIs difficult to use. These variations in the EEG signal properties might deteriorate the BCI performance. One possible approach to overcome this problem is to use semi-supervised learning algorithms. These algorithms have the ability to use a small set of labeled data with a large amount of unlabeled data. Specially, in deep learning methods unlabeled data can be used to pre-train the neural network and the labeled data can be used to perform supervised fine-tuning of the network. The semi-supervised sequence learning [30] algorithms can be especially useful for self-paced BCIs. These classifiers can be trained using a small amount of training data. While the subject is operating the BCI, the classifier can improve itself based on the new coming EEG data.

Bibliography

[1] Sarah N. Abdulkader, Ayman Atia, and Mostafa-Sami M. Mostafa. Brain computer interfacing: Applications and challenges. *Egyptian Informatics Journal*, 16(2):213 – 230, 2015. → pages 1

[2] Yasemin Altun, Ioannis Tsochantaridis, Thomas Hofmann, et al. Hidden markov support vector machines. In *ICML*, volume 3, pages 3–10, 2003. → pages 76, 77

[3] Omar AlZoubi, Irena Koprinska, and Rafael A Calvo. Classification of brain-computer interface data. In *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, pages 123–131. Australian Computer Society, Inc., 2008. → pages 15

[4] Kai Keng Ang, Zheng Yang Chin, Chuanchu Wang, Cuntai Guan, and Haihong Zhang. Filter bank common spatial pattern algorithm on bci competition iv datasets 2a and 2b. *Frontiers in Neuroscience*, 6, 2012. → pages 63, 65, 66

[5] Mahnaz Arvaneh, Cuntai Guan, Kai Keng Ang, and Chai Quek. Optimizing the channel selection and classification accuracy in eeg-based bci. *Biomedical Engineering, IEEE Transactions on*, 58(6):1865–1873, 2011. → pages 7, 42

[6] Onder Aydemir and Temel Kayikcioglu. Comparing common machine learning classifiers in low-dimensional feature vectors for brain computer interface applications. → pages 15

[7] Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Multiclass brain–computer interface classification by riemannian geometry. *Biomedical Engineering, IEEE Transactions on*, 59(4):920–928, 2012. → pages 63, 65

[8] Ali Bashashati, Mehrdad Fatourechi, Rabab K Ward, and Gary E Birch. A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals. *Journal of Neural engineering*, 4(2):R32, 2007. → pages 6, 16, 72

[9] Ali Bashashati, Rabab K Ward, and Gary E Birch. Towards development of a 3-state self-paced brain-computer interface. *Computational intelligence and neuroscience*, 2007, 2007. → pages 4, 71

[10] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012. → pages 53

[11] Martin Billinger, Vera Kaiser, Christa Neuper, and Clemens Brunner. *Automatic frequency band selection for BCIs with ERDS difference maps*. → pages 42

[12] Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. 1998. → pages 74

[13] G. Bin, X. Gao, Y. Wang, B. Hong, and S. Gao. Vep-based brain-computer interfaces: time, frequency, and code modulations [research frontier]. *IEEE Computational Intelligence Magazine*, 4(4):22–26, November 2009. → pages 2

[14] Christopher M Bishop. *Pattern recognition and machine learning*, volume 1. springer New York, 2006. → pages 24

[15] Benjamin Blankertz, Guido Dornhege, Matthias Krauledat, Klaus-Robert Müller, and Gabriel Curio. The non-invasive berlin brain–computer interface: fast acquisition of effective performance in untrained subjects. *NeuroImage*, 37(2):539–550, 2007. → pages 28

[16] Benjamin Blankertz, K Muller, Dean J Krusienski, Gerwin Schalk, Jonathan R Wolpaw, Alois Schlogl, Gert Pfurtscheller, Jd R Millan, M Schroder, and Niels Birbaumer. The bci competition iii: Validating alternative approaches to actual bci problems. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 14(2):153–159, 2006. → pages 28, 56

[17] Benjamin Blankertz, Michael Tangermann, Carmen Vidaurre, Siamac Fazli, Claudia Sannelli, Stefan Haufe, Cecilia Maeder, Lenny E Ramsey, Irene Sturm, Gabriel Curio, and Klaus R Mueller. The berlin brain-computer interface: Non-medical uses of bci technology. *Frontiers in Neuroscience*, 4(198), 2010. → pages 1

[18] Benjamin Blankertz, Ryota Tomioka, Steven Lemm, Motoaki Kawanabe, and K-R Muller. Optimizing spatial filters for robust eeg single-trial analysis. *Signal Processing Magazine, IEEE*, 25(1):41–56, 2008. → pages 18, 42

[19] Reza Boostani, Bernhard Graimann, MH Moradi, and Gert Pfurtscheller. A comparison approach toward finding the best feature and classifier in cue-based bci. *Medical & biological engineering & computing*, 45(4):403–412, 2007. → pages 15

[20] Jaimie F Borisoff, Steven G Mason, Ali Bashashati, and Gary E Birch. Brain-computer interface design for asynchronous control applications: improvements to the lf-asd asynchronous brain switch. *Biomedical Engineering, IEEE Transactions on*, 51(6):985–992, 2004. → pages 15, 28, 29

[21] Andreas Trllund Boye, Ulrik Qvist Kristiansen, Martin Billinger, Omar Feix do Nascimento, and Dario Farina. Identification of movement-related cortical potentials with optimized spatial filtering and principal component analysis. *Biomedical Signal Processing and Control*, 3(4):300 – 304, 2008. → pages 13

[22] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. → pages 22

[23] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010. → pages 43

[24] Nicolas Brodu, Fabien Lotte, and Anatole Lcuyer. Exploring two novel features for eeg-based brain–computer interfaces: Multifractal cumulants and predictive complexity. *Neurocomputing*, 79:87–94, 2012. → pages 19, 52, 57, 63, 64, 65

[25] Nicolas Brodu, Fabien Lotte, and Anatole Lécuyer. Comparative study of band-power extraction techniques for motor imagery classification. In *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium on*, pages 1–6. IEEE, 2011. → pages 19, 63, 64, 65

[26] C Brunner, R Leeb, G Müller-Putz, A Schlögl, and G Pfurtscheller. Bci competition 2008–graz data set a. → pages 28, 56

[27] Silvia Chiappa, Nicolas Donckers, Samy Bengio, and Frédéric Vrins. Hmm and iohmm modeling of eeg rhythms for asynchronous bci systems. In *ESANN*, 2004. → pages 72, 84

[28] Zheng Yang Chin, Kai Keng Ang, Chuanchu Wang, Cuntai Guan, and Haihong Zhang. Multi-class filter bank common spatial pattern for four-class motor imagery bci. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 571–574. IEEE, 2009. → pages 42

[29] A Criminisi, J Shotton, and E Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. → pages 22, 45

[30] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015. → pages 99

[31] Michele Dalponte, Francesca Bovolo, and Lorenzo Bruzzone. Automatic selection of frequency and time intervals for classification of eeg signals. *Electronics Letters*, 43(25):1406–1408, 2007. → pages 42

[32] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006. → pages 26, 58

[33] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000. → pages 22

[34] Thomas G Dietterich. Machine learning for sequential data: A review. In *Structural, syntactic, and statistical pattern recognition*, pages 15–30. Springer, 2002. → pages 71

[35] Trinh Do, Thierry Arti, et al. Neural conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, pages 177–184, 2010. → pages 81

[36] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273, 2004. → pages 54

[37] Lawrence Ashley Farwell and Emanuel Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523, 1988. → pages 3

[38] Mehrdad Fatourechi, Ali Bashashati, Rabab K Ward, and Gary E Birch. Emg and eog artifacts in brain computer interface systems: A survey. *Clinical neurophysiology*, 118(3):480–494, 2007. → pages 7

[39] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006. → pages 24

[40] Reza Fazel-Rezai, Brendan Z Allison, Christoph Guger, Eric W Sellers, Sonja C Kleih, and Andrea Kübler. P300 brain computer interface: current challenges and emerging trends. *Frontiers in Neuroengineering*, 5:14, 2012. → pages 3

[41] Yoav Freund and Robert Schapire. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999. → pages 23

[42] Jerome H Friedman. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989. → pages 21

[43] Ferran Galán, Marnix Nuttin, Eileen Lew, Pierre W Ferrez, Gerolf Vanacker, Johan Philips, and J del R Millán. A brain-actuated wheelchair: asynchronous and non-invasive brain–computer interfaces for continuous control of robots. *Clinical Neurophysiology*, 119(9):2159–2169, 2008. → pages 13

[44] Salvador Garcıa and Francisco Herrera. An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008. → pages 26, 61

[45] Bashar Awwad Shiekh Hasan and John Q Gan. Multi-objective particle swarm optimization for channel selection in brain-computer interfaces. → pages 7, 42

[46] Bashar Awwad Shiekh Hasan and John Q Gan. Conditional random fields as classifiers for three-class motor-imagery braincomputer interfaces. *Journal of Neural Engineering*, 8(2):025013, 2011. → pages 73

[47] Pawel Herman, Girijesh Prasad, Thomas Martin McGinnity, and Damien Coyle. Comparative analysis of spectral approaches to feature extraction for eeg-based motor imagery classification. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 16(4):317–326, 2008. → pages 19

[48] T. Hinterberger, S. Schmidt, N. Neumann, J. Mellinger, B. Blankertz, G. Curio, and N. Birbaumer. Brain-computer communication and slow cortical potentials. *IEEE Transactions on Biomedical Engineering*, 51(6):1011–1018, June 2004. → pages 3

[49] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. → pages 99

[50] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991. → pages 24

[51] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. → pages 22

[52] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002. → pages 23

[53] Han-Jeong Hwang, Soyoun Kim, Soobeom Choi, and Chang-Hwan Im. Eeg-based brain-computer interfaces: A thorough literature survey. *International Journal of Human-Computer Interaction*, 29(12):814–826, 2013. → pages 16

[54] Donald R Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001. → pages 43

[55] S Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003. → pages 23

[56] Jun-Yeup Kim, Seung-Min Park, Kwang-Eun Ko, and Kwee-Bo Sim. Optimal eeg channel selection for motor imagery bci system using bpso and ga. In *Robot Intelligence Technology and Applications 2012*, pages 231–239. Springer, 2013. → pages 7

[57] Wlodzimierz Klonowski. Everything you wanted to ask about eeg but were afraid to get the right answer. *Nonlinear Biomedical Physics*, 3(1):2, 2009. → pages 7

[58] Roman Krepki, Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller. The berlin brain-computer interface (bbci)–towards a new communication channel for online control in gaming applications. *Multimedia Tools and Applications*, 33(1):73–90, 2007. → pages 3

[59] Andrea Kübler, Femke Nijboer, Jürgen Mellinger, Theresa M Vaughan, Hannelore Pawelzik, Gerwin Schalk, Dennis J McFarland, Niels Birbaumer, and Jonathan R Wolpaw. Patients with als can use sensorimotor rhythms to operate a brain-computer interface. *Neurology*, 64(10):1775–1777, 2005. → pages 4

[60] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289. → pages 73

[61] Tian Lan, Deniz Erdogmus, Andre Adami, Misha Pavel, and Santosh Mathan. Salient eeg channel selection in brain computer interfaces by mutual information maximization. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 7064–7067. IEEE, 2006. → pages 42

[62] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. → pages 8, 98

[63] Felix Lee, Reinhold Scherer, Robert Leeb, Christa Neuper, Horst Bischof, and Gert Pfurtscheller. A comparative analysis of multi-class eeg classification for brain computer interface. → pages 15

[64] R Leeb, C Brunner, GR Müller-Putz, A Schlögl, and G Pfurtscheller. Bci competition 2008–graz data set b. → pages 28, 56

[65] Xu Lei, Ping Yang, and Dezhong Yao. An empirical bayesian framework for brain–computer interfaces. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 17(6):521–529, 2009. → pages 21

[66] Eric C Leuthardt, Kai J Miller, Gerwin Schalk, Rajesh PN Rao, and Jeffrey G Ojemann. Electrocorticography-based brain computer interface-the seattle experience. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):194–198, 2006. → pages 2

[67] Peiyang Li, Peng Xu, Rui Zhang, Lanjin Guo, and Dezhong Yao. L1 norm based common spatial patterns decomposition for scalp eeg bci. *Biomed. Eng. Online*, 12, 2013. → pages 16

[68] Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, Bruno Arnaldi, et al. A review of classification algorithms for eeg-based brain–computer interfaces. *Journal of neural engineering*, 4, 2007. → pages 15

[69] Steven G Mason and Gary E Birch. A brain-controlled switch for asynchronous control applications. *Biomedical Engineering, IEEE Transactions on*, 47(10):1297–1307, 2000. → pages 4

[70] Dennis J McFarland, Laurie A Miner, Theresa M Vaughan, and Jonathan R Wolpaw. Mu and beta rhythm topographies during motor imagery and actual movements. *Brain topography*, 12(3):177–186, 2000. → pages 3

[71] José del R Millán, Frédéric Renkens, Josep Mouriño, and Wulfram Gerstner. Brain-actuated interaction. *Artificial Intelligence*, 159(1):241–259, 2004. → pages 3

[72] Jonas Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4(4):347–365, 1994. → pages 45

[73] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. → pages 20

[74] Nam Nguyen and Yunsong Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th international conference on Machine learning*, pages 681–688. ACM, 2007. → pages 73

[75] Luis Fernando Nicolas-Alonso and Jaime Gomez-Gil. Brain computer interfaces, a review. *Sensors*, 12(2):1211–1279, 2012. → pages 2, 6

[76] Jian Peng, Liefeng Bo, and Jinbo Xu. Conditional neural fields. In *Advances in neural information processing systems*, pages 1419–1427, 2009. → pages 81

[77] Gert Pfurtscheller and Fernando H Lopes da Silva. Event-related eeg/meg synchronization and desynchronization: basic principles. *Clinical neurophysiology*, 110(11):1842–1857, 1999. → pages 9, 41, 72

[78] Gert Pfurtscheller, Gernot R Müller, Jörg Pfurtscheller, Hans Jürgen Gerner, and Rüdiger Rupp. thought–control of functional electrical stimulation to restore hand grasp in a patient with tetraplegia. *Neuroscience letters*, 351(1):33–36, 2003. → pages 1

[79] Gert Pfurtscheller and Christa Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, 2001. → pages 41

[80] Maja Pohar, Mateja Blas, and Sandra Turk. Comparison of logistic regression and linear discriminant analysis. *MetodoloÅ aki zvezki*, 1(1):143–161, 2004. → pages 21

[81] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. → pages 72, 74

[82] Herbert Ramoser, Johannes Muller-Gerking, and Gert Pfurtscheller. Optimal spatial filtering of single trial eeg during imagined hand movement. *Rehabilitation Engineering, IEEE Transactions on*, 8(4):441–446, 2000. → pages 18

[83] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006. → pages 44

[84] Daran Ravden and John Polich. On p300 measurement stability: habituation, intra-trial block variation, and ultradian rhythms. *Biological psychology*, 51(1):59–76, 1999. → pages 3

[85] Yann Renard, Fabien Lotte, Guillaume Gibert, Marco Congedo, Emmanuel Maby, Vincent Delannoy, Olivier Bertrand, and Anatole Lécuyer. Openvibe: an open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments. *Presence: teleoperators and virtual environments*, 19(1):35–53, 2010. → pages 10

[86] Delgado Saa, F Jaime, and Mujdat Cetin. Discriminative methods for classification of asynchronous imaginary motor tasks from eeg data. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 21(5):716–724, 2013. → pages 73

[87] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *Biomedical Engineering, IEEE Transactions on*, 51(6):1034–1043, 2004. → pages 10

[88] R. Scherer, F. Lee, A. Schlogl, R. Leeb, H. Bischof, and G. Pfurtscheller. Toward self-paced brain-computer communication: Navigation through virtual worlds. *IEEE Transactions on Biomedical Engineering*, 55(2):675–682, Feb 2008. → pages 13

[89] Reinhold Scherer, GR Muller, Christa Neuper, Bernhard Graimann, and Gert Pfurtscheller. An asynchronously controlled eeg-based virtual keyboard: improvement of the spelling rate. *IEEE Transactions on Biomedical Engineering*, 51(6):979–984, 2004. → pages 3

[90] Mark R Segal. Machine learning benchmarks and random forest regression. *Center for Bioinformatics & Molecular Biostatistics*, 2004. → pages 22

[91] AE Selim, M Abdel Wahed, and YM Kadah. Machine learning methodologies in brain-computer interface systems. In *Biomedical Engineering Conference, 2008. CIBEC 2008. Cairo International*, pages 1–5. IEEE, 2008. → pages 15

[92] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012. → pages 52

[93] Heung-Il Suk and Seong-Whan Lee. Subject and class specific frequency bands selection for multiclass motor imagery classification. *International Journal of Imaging Systems and Technology*, 21(2):123–130, 2011. → pages 42

[94] Gufei Sun, Jinglu Hu, and Gengfeng Wu. A novel frequency band selection method for common spatial pattern in motor imagery based brain computer interface. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–6. IEEE, 2010. → pages 42

[95] Kai Ming Ting and Ian H Witten. Issues in stacked generalization. *arXiv preprint arXiv:1105.5466*, 2011. → pages 54

[96] G. Townsend, B. Graimann, and G. Pfurtscheller. Continuous eeg classification during motor imagery-simulation of an asynchronous bci. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(2):258–265, June 2004. → pages 13

[97] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning (ICML)*, pages 104–112, 2004. → pages 78

[98] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005. → pages 76

[99] Chun Sing Louis Tsui and John Q. Gan. *Asynchronous BCI Control of a Robot Simulator with Supervised Online Training*, pages 125–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. → pages 13

[100] Douglas L Vail, Manuela M Veloso, and John D Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 235. ACM, 2007. → pages 73

[101] Jan BF Van Erp, Fabien Lotte, Michael Tangermann, et al. Brain-computer interfaces: beyond medical applications. *Computer-IEEE Computer Society-*, 45(4):26–34, 2012. → pages 1

[102] Jonathan R Wolpaw, Niels Birbaumer, Dennis J McFarland, Gert Pfurtscheller, and Theresa M Vaughan. Brain–computer interfaces for communication and control. *Clinical neurophysiology*, 113(6):767–791, 2002. → pages 1, 7

[103] Peng Xu, Ping Yang, Xu Lei, and Dezhong Yao. An enhanced probabilistic lda for multi-class brain computer interface. *PloS one*, 6(1):e14634, 2011. → pages 16

[104] Xinyi Yong, Rabab K Ward, and Gary E Birch. Sparse spatial filter optimization for eeg channel reduction in brain-computer interface. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 417–420. IEEE, 2008. → pages 42

[105] Rui Zhang, Peng Xu, Lanjin Guo, Yangsong Zhang, Peiyang Li, and Dezhong Yao. Z-score linear discriminant analysis for eeg based brain-computer interfaces. *PloS one*, 8(9):e74433, 2013. → pages 21

[106] Mingjun Zhong, Fabien Lotte, Mark Girolami, and Anatole Lécuyer. Classifying eeg for brain computer interfaces using gaussian processes. *Pattern Recognition Letters*, 29(3):354–359, 2008. → pages 63, 64