THESIS

ELECTROENCEPHALOGRAM CLASSIFICATION BY FORECASTING WITH RECURRENT NEURAL NETWORKS

Submitted by

Elliott M. Forney

Department of Computer Science

In partial fulfillment of the requirements

for the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2011

Master's Committee:

Advisor: Charles Anderson

Asa Ben-Hur
William Gavin

ABSTRACT

ELECTROENCEPHALOGRAM CLASSIFICATION BY
FORECASTING WITH RECURRENT NEURAL NETWORKS

The ability to effectively classify electroencephalograms (EEG) is the foundation for building usable Brain-Computer Interfaces as well as improving the performance of EEG analysis software used in clinical and research settings. Although a number of research groups have demonstrated the feasibility of EEG classification, these methods have not yet reached a level of performance that is acceptable for use in many practical applications. We assert that current approaches are limited by their ability to capture the temporal and spatial patterns contained within EEG. In order to address these problems, we propose a new generative technique for EEG classification that uses Elman Recurrent Neural Networks. EEG recorded while a subject performs one of several imagined mental tasks is first modeled by training a network to forecast the signal a single step ahead in time. We show that these models are able to forecast EEG with an error as low as 1.18 percent of the signal range. A separate model is then trained over EEG belonging to each class. Classification of previously unseen data is performed by applying each model and using Winner-Takes-All, Linear Discriminant Analysis or Quadratic Discriminant Analysis to label the forecasting errors. This approach is tested on EEG collected from two able-bodied subjects and three subjects with disabilities. Information transfer rates as high as 38.7 bits per minute (bpm) are achieved for a two-task problem and 34.5bpm for a four-task problem.

ACKNOWLEDGMENTS

I would like to thank Chuck Anderson for the invaluable ideas, feedback and critiques he has provided me throughout my research. I would also like to thank the members of the BCI and AI research groups and my committee members for their time, insights and discussion. I am very grateful to all of the volunteers that gave up their valuable time for us to record their EEG. I would like to thank all of my teachers for the knowledge, guidance and inspiration that has enabled me to become a successful researcher. I am extremely grateful to my family and friends for their inspiration, support, encouragement and, of course, for occasionally distracting me. Finally, I would like to express my gratitude to the developers of bash, LaTeX, Linux, R, vim and the other tools that I use in my work every day.

TABLE OF CONTENTS

1 Introduction 1

2 Background 5

2.1 BCI Paradigms 5

2.2 Feature Representations 8

2.3 Recurrent Neural Networks 10

3 Methods 13

3.1 Experimental Data 13

3.2 Elman Recurrent Neural Networks 18

3.2.1 Architecture 18

3.2.2 Sigmoid and Initial Weights 20

3.2.3 Training 21

3.3 Forecasting 24

3.4 Classification by Forecasting 26

3.4.1 Averaged Winner-Takes-All 27

3.4.2 Quadratic Discriminant Analysis 28

3.4.3 Linear Discriminant Analysis 29

4 Results and Discussion 31

4.1 Forecasting 31

4.1.1 Parameter Selection 32

4.1.2 Iterated Models 36

4.2 Classification 40

4.2.1 Regularization 40
4.2.2 Accuracy 43
4.2.3 Information Transfer Rate 47
4.2.4 Decision Rate 48
4.2.5 Error Boundaries 52

5 Conclusions 56
5.1 Summary 56
5.2 State-of-the-art 57
5.3 Future Work 58

Bibliography 61

Chapter 1

Introduction

Electroencephalography (EEG) is a technique for measuring synchronized neural activity by placing an array of electrodes on the surface of a subject's scalp. Since the recording of the first EEG signals by Hans Berger in 1924 [1], EEG has been used in a number of clinical applications. For example, EEG is used as a diagnostic tool for epilepsy, brain damage and sleep disorders and is commonly used in a variety of research contexts, such as the localization of neural activity, the study of human development and the characterization of numerous medical conditions [2].

Recently, an interest has also been sparked in the use of EEG for establishing a direct channel of communication between the brain and a computerized device. These Brain-Computer Interfaces (BCI) allow a user to control an external device by voluntarily altering their mental state while a pattern analysis algorithm simultaneously attempts to identify the corresponding change in the EEG signals. A device such as a mouse cursor, robot or wheelchair can be instructed to take an action that has been previously associated with the detected change in mental state. This procedure effectively bypasses our innate, motor-based means of communication.

BCI are of immediate interest to those who have lost all ability to communicate with the outside world. This condition, known as locked-in syndrome (LIS), can be caused by a number of diseases, such as amyotrophic lateral sclerosis, cerebral palsy and stroke [3]. In addition to those suffering from LIS, BCI may also be extremely helpful to those who are able to communicate with the outside world yet have an impaired ability to operate everyday devices, such as wheelchairs, computers and telephones. Such impairments are

common and can be caused by spinal cord injury, multiple sclerosis and traumatic brain injury. Beyond assistive technology, one might also imagine applications for BCI in virtual reality, gaming, monitoring of emotional states and the every-day interaction between humans and machines.

As exciting as these prospects may be, reliable and real-time analysis of EEG is difficult to achieve for a number of reasons. First of all, EEG measurements are extremely sensitive and tend to have a very low signal to noise ratio. The electrical activity observed on the surface of the scalp is measured on the microvolt level, requiring extremely sensitive amplifiers and low impedances between the electrodes and the scalp. As a result of the small electrical potentials involved, EEG is also very susceptible to contamination by various types of artifacts and noise. For example, biological sources of electrical activity such as sinus rhythms, muscle contraction and ocular movement can cause artifacts that overwhelm the signals generated by neural activity. Environmental electrical sources, such as domestic wiring, household appliances and computer peripherals, can also produce a significant amount of noise in EEG recordings, particularly in everyday settings. What's more, the electrical effects generated by neurons are extremely complex and are influenced by cell orientation, surface contours of the brain and the shape of a subject's head [2]. These barriers cause the underlying signals to mix and fade so that only the effects of synchronized firing by large pools of neurons can be measured.

In addition to the limitations of EEG measurement, the very nature of the human brain makes EEG classification difficult. The human brain contains on the order of 100 billion neurons. Each neuron may have numerous connections, each involved in complex electro-chemical interactions. Furthermore, the human brain is stateful, meaning EEG recorded at any given moment in time is closely related to the preceding activity. On top of the sheer complexity of the brain is the fact that human thought has a tendency to wander and perform multiple tasks simultaneously. Even if a specific kind of neural activity can be correctly identified from EEG, it may be difficult to know when a subject has lost concentration or is performing confounding tasks.

In this thesis we investigate a non-linear, generative approach to EEG classification that is designed to be both robust to artifacts and noise as well as powerful enough to capture the complex patterns found in EEG. In this approach, we first construct a model for each class of EEG that attempts to forecast the signal a single step ahead in time. Once a model is constructed for each class, labels can be assigned to novel data

by applying each model and subsequently assigning the class label associated with the model that was best able to forecast the EEG signal or by applying a secondary classifier to the forecasting errors.

In order to forecast EEG we use a type of learning machine known as the Elman Recurrent Neural Network (ERNN) [4]. ERNN belong to a broader set of learning machines known as Artificial Neural Networks (ANN) [5]. ANN are comprised of a number of simple computational units known as neurons. The neurons in an ANN have weighted interconnections, including connections to input and output lines. ANN are trained to map some inputs to outputs by adjusting these weighted interconnections in a way that minimizes a given objective function. The ANN used here are trained using example inputs and outputs for which the correct values are known, i.e., in a supervised fashion. It should be noted that while the ANN used here are biologically inspired neural models, they are quite far from being biologically plausible.

More specifically, ERNN belong to a subset of ANN known as Recurrent Artificial Neural Networks (RNN). While feedforward ANN contain only connections from layer to layer moving from the input layer to the output layer, RNN contain delayed feedback connections. These feedback connections give RNN an intrinsic state, or memory, and the ability to incorporate information from inputs that were previously presented to the network. When combined with non-linear activation functions, RNN are capable of learning complex spatiotemporal patterns. Since RNN are able to model both spatial patterns as well as temporal patterns, and because training of our RNN requires few prior assumptions, we hypothesize that this approach may be able to capture many of the complex patterns found in EEG while remaining robust to artifacts and noise. In previous work, we have offered preliminary results demonstrating that ERNN can be used to construct effective EEG classifiers [6].

In the experiments conducted here, we perform an offline analysis of EEG recorded from five subjects. Two of these subjects are able-bodied, two have high-level spinal cord injuries and one has severe multiple sclerosis. The goal of our analysis is to construct a pipeline for EEG classification that can be used to build BCI systems. Following the seminal work of Keirn and Aunon [7], we choose to focus on discriminating between EEG produced while subject performs each of several different imagined mental tasks. A subject may then control a computerized device by switching their mental state from task to task in a controlled manner. We feel that this approach is general and offers many degrees of freedom to BCI users. Additionally, classifying generic mental tasks retains the potential for mutual learning. In other words, a BCI user may

learn to perform the mental tasks in ways that improve system performance while a computer algorithm simultaneously learns to discriminate between patterns found in the user's EEG. Although this approach seems cumbersome at first, it may become second-nature after extended periods of use. Finally, we hope that this approach is general enough to be applied to other EEG analysis domains.

In Chapter 2 we begin by reviewing the current state-of-the-art in EEG classification and BCI systems. This review includes a discussion of popular feature representations and classification algorithms as well as their potential limitations. In Chapter 3 we outline the details of the experimental protocol used throughout the remainder of this thesis as well as thoroughly introduce ERNN and our classification procedure. In Chapter 4 begin by selecting proper parameters for training and regularizing our forecasters and classifiers. We then present the final classification performance of our techniques for all five subjects. We also offer some discussion regarding the level of temporal information that ERNN are able to capture and how classification performance might translate to the control of a BCI system. In Chapter 5 we conclude by summarizing our results, performing a brief comparison with other methods and commenting on some questions that remain to be answered in future work.

Chapter 2

Background

In this chapter we give a brief overview and discussion of the state-of-the-art in BCI systems. In order to achieve this we begin by reviewing a number of popular paradigms around which BCI are often built. We then attempt to roughly identify the current state-of-the-art for each approach. Next, we offer some discussion regarding several common EEG feature representations. Since this thesis is focused on the classification of EEG recorded during imagined mental tasks, our discussion of feature representations is geared specifically toward BCI that follow this paradigm. Finally, we discuss several papers found in the current literature that are closely related to the work at hand.

2.1 BCI Paradigms

A number of paradigms for building EEG-based BCI systems have been explored in the past two decades. These paradigms vary widely, from using well-known phenomena that occur in EEG during interaction with a stimuli, to the use of biofeedback, to paradigms that use sophisticated machine learning algorithms to classify spontaneous EEG. Each of these approaches has advantages and disadvantages and, in order to justify the methods used in this thesis, it is important to briefly review the most common BCI paradigms currently in use. Furthermore, it is important to identify the performance of the current state-of-the-art in BCI so that we can properly evaluate the performance of the techniques used here. The following review of BCI paradigms is organized to begin with methods that are more constrained, in the sense that they require

a level of overt attention from the user, and concludes with methods that require little or no external stimuli and only the user's covert attention.

Paradigms for building BCI systems that utilize specific changes in EEG that occur following the presentation of a controlled external stimuli have been implemented with considerable success. Since these changes in EEG are brought about by the presentation of a stimulus, they are known as evoked potentials (EP). For example, a number of BCI systems have been built that utilize Steady-State Visually Evoked Potentials (SSVEP). SSVEP can be elicited by instructing a subject to fixate on a box or checkerboard presented on an LCD screen that is flickering at a fixed rate. A corresponding increase in power can then be identified in the subject's EEG at the same frequency and at the harmonic frequencies of this flickering. SSVEP can be used to control a computerized device by flickering several different stimuli at different rates while allowing the user to shift their gaze between the different stimuli [8]. BCI systems that operate in this fashion have proven to be quite effective, with recent research suggesting that an SSVEP speller system can be constructed that achieves information transfer rates as high as 62.5 bits per minute (bpm) with only minimal user training [9].

Another type of EP that is commonly used to construct BCI systems is known as the P300. The P300 is an EP that occurs following the presentation of a rare-but-expected stimulus. The P300 is so named because it appears in averaged EEG signals as a positive deflection roughly 300ms following the onset of the presented stimulus. One example of a BCI system that utilizes the P300 is known as the P300 speller. In this approach, a grid of numbers or letters is presented to the user on an LCD screen. The rows and columns of this grid are then flashed in a pseudo-random order. Ideally, a P300 is then elicited when the user attends to a single character in the grid, since the character flashes relatively infrequently and at intervals that are unknown to the subject [10]. The BCI can then determine which character the user was attending to by tracking when each row and column was flashed. Recent studies indicate that the P300 speller can be used very successfully, achieving information transfer rates as high as 13.3bpm in subjects with amyotrophic lateral sclerosis and a comparable 11.3bpm in healthy subjects [11].

Although approaches that utilize EP often achieve impressive results, they also suffer from a number of fundamental limitations. Since the user of the BCI is required to attend to some stimulus, they may become distracted from the task that they wish the computer to perform or the message they wish to communicate. It

is difficult to imagine, for example, that a subject would be able control an electric wheelchair or prosthetic device while simultaneously attending to a visual stimulus on an LCD screen. Furthermore, methods that require a subject to carefully manipulate their gaze may not be practical for those with some kinds of disabilities. Some users, for example, may not be able to precisely focus on a visual stimulus. Finally, BCI users may find the repetitive presentation of a stimulus to be unpleasant or even irritating.

One way of avoiding the problems found in paradigms that utilize EP is to classify spontaneous EEG that is not directly tied to an external stimulus. To this end, several groups have developed BCI systems that exploit the ability of many subjects to voluntarily manipulate their μ (8-12Hz) and β (13-28Hz) rhythms in EEG recorded over their sensory-motor cortices. For example, Wolpaw, et al., have explored BCI systems that operate by training users to alter the power of their μ and β rhythms through the use of biofeedback [12]. Similarly, Pfurtscheller, et al., have explored techniques for operating BCI systems where the user issues commands by performing imagined motor tasks, which have also been linked to changes in μ and β rhythms [13]. The work done by both of these groups extensively combines the use of biofeedback with machine learning. Although this mutual learning certainly improves the user's ability to use these BCI systems, it can make comparison with other approaches difficult. It should be noted, however, that the classification rates achieved by Pfurtscheller, et al., have been reported to be as high as 98% correct with binary decisions made in intervals of roughly 4s or less after three training sessions conducted over three consecutive days.

As a further generalization upon the idea of classifying EEG recorded during imagined motor imagery, Keirn, et al., have suggested the classification of EEG recorded during a broader set of imagined mental tasks [7]. The imagined mental tasks to be classified are typically chosen to be as neurologically different as possible. For example, the mental tasks used by Keirn, et al., included complex problem solving, geometric figure rotation, mental letter composing and visual counting. In this paradigm, a user can issue commands to a BCI system by performing an imagined mental task that was associated with the desired command before training. Recent research by Gálan, et al., has demonstrated that this paradigm can successfully be used to navigate an electric wheelchair equipped with laser range-finders through an obstacle course [14]. Anderson, et al., have also shown that this approach can achieve a correct decision roughly every three seconds when using four imagined mental tasks [15].

Classification of EEG recorded during imagined mental tasks can be extremely difficult for a number of reasons. First of all, the patterns found in EEG during such mental tasks can vary widely across subjects and even within the same subject but at different times. Second, a subject may be performing more than one task simultaneously without being aware that they are doing so. For example, a subject may be visualizing the numbers in a counting problem, which may overlap with a separate visualization task. Finally, it is unclear for whom and for which mental tasks EEG signals contain enough information to reliably discriminate between the tasks. Nevertheless, this paradigm is extremely appealing because it offers many degrees of freedom to BCI users and because it does not require an external stimulus. Furthermore, it allows subjects to choose between a number of mental tasks if they find one to perform poorly or to be unappealing. For these reasons, the remainder of this thesis is devoted to the classification of EEG recorded during imagined mental tasks.

2.2 Feature Representations

As is the case with many classification problems, finding an appropriate way to represent EEG data is an integral part of any BCI system. Several things are particularly important to consider when selecting a feature representation for EEG classification. First, EEG signals contain information that is both spatial as well as temporal in nature. Any feature representation that neglects patterns that occur either across electrodes or through time may discard important patterns that are present in the signal. Second, EEG is typically recorded from between 8 and 64 electrodes, making it relatively high-dimensional. EEG also has relatively high temporal resolution, typically between 128 and 1024 samples per second, yielding a significant amount of data to process. Since BCI systems deliver results in real-time and on minimal computing hardware, it is important that any feature representation used in a BCI allows for the data to be processed quickly. Finally, EEG signals are known to be extremely noisy, particularly in everyday environments. A feature representation that eliminates noisy components of the signal or components that are not discriminating may ease classification.

Since EEG signals often contain patterns that are oscillatory in nature, feature representations that utilize frequency spectra are commonly used in BCI systems. In the context of classifying imagined mental tasks, Millán, et al., are notable for their successful use of frequency-based feature representations [16, 17, 14]. In

this approach, the Power Spectral Density (PSD) of the EEG signal is typically estimated across a relatively short window for each electrode. Next, each PSD is divided into a number of bins, representing a frequency band. Finally, a feature selection algorithm, such as canonical variates analysis, is used to select relevant features that can ultimately be passed to a classification algorithm.

Although approaches that utilize PSD feature representations have delivered impressive results, it is important to note that phase information is typically not considered. Since both phase and frequency information are required to convert from the frequency domain to the time domain, it is clear that some information is discarded. More specifically, feature representations that are based on PSD cannot readily express differences in phase across electrodes and, therefore, some forms of spatial patterns. Krusienski, et al., note that it may be important to consider phase information [18]. Furthermore, Gysels, et al., have demonstrated that phase-based features can be useful during the classification of imagined mental tasks [19]. Additionally, the fact that a PSD is typically estimated across a window implies that some forms of short-term temporal patterns may be indistinguishable. For example, if the EEG signal moves from a low frequency to a high frequency during the window it is indistinguishable from an EEG signal that moves from a high frequency to a low frequency.

Time-Delay Embedding (TDE) is a feature representation that may be better able to capture both spatial and temporal patterns. In this approach, all of the samples in a window of EEG are concatenated together to form a single, higher-dimensional feature. In the context of classifying imagined mental tasks, Anderson, et al., use TDE in combination with blind source separation or dimensionality reduction, such as Short-Term Principal Components Analysis or Maximum Noise Fraction [15, 20, 21]. Although TDE may avoid some of the problems encountered when using features based on PSD, it also has several drawbacks. Importantly, the length of temporal patterns that can be captured by TDE feature representations is limited by the number of individual samples that are concatenated together, known as the embedding dimension. Furthermore, the dimensionality of TDE features increases linearly with size of the embedding dimension. The higher dimensionality of TDE features may require that either dimensionality reduction techniques be used, which may discard important aspects of EEG if not used carefully, or a relatively large amount of training data be provided to sufficiently sample the feature space.

Although current approaches have demonstrated that it is feasible to classify EEG and construct BCI systems, these methods ultimately do not deliver classification accuracies that are high enough for use in practical and reliable BCI systems. Due to the drawbacks found in current feature representations and because of the difficulties in operating BCI systems that rely on EP, we feel that it is important to continue exploring novel ways of representing the spatiotemporal patterns found in EEG recorded during imagined mental tasks. Since RNN may have the potential to find such patterns, they offer an exciting alternative to current approaches.

2.3 Recurrent Neural Networks

Despite the potential that RNN may have for capturing patterns in EEG, the current literature pertaining to this subject is relatively sparse. There are some works, however, that use RNN to classify EEG outside the realm of BCI. For example, Güler and Übeyli have utilized ERNN to classify normal and epileptiform EEG in subjects with epilepsy. In these approaches, EEG signals are represented using low dimensional statistical features extracted from Lyapunov Spectra or frequency-based properties generated using MUSIC or Pisarenko's method [22, 23]. ERNN are then compared with strictly feedforward networks by training them to output an indicator variable corresponding to the desired class label. These techniques have achieved classification rates in the 90% range for one-second EEG segments and have demonstrated that ERNN typically achieve about 5% better classification accuracy than strictly feedforward networks in this setting. These works did not, however, compare ERNN with feedforward ANN that incorporate TDE.

The approach used in this thesis is quite different than that taken by Güler and Übeyli. As we explain in detail in Chapter 3, we do not train our ERNN to output class labels directly. Rather, we train a separate ERNN for each class to forecast the EEG signals a single step ahead in time. This yields an expert at modeling the EEG from each class. Classification is then performed by examining the forecasting errors produced by each ERNN over novel data. In this way, the errors produced by our ERNN can be thought of as low-dimensional features that can subsequently be passed to a classifier.

Gupta, et al., have suggested a technique for time-series classification [24] that is conceptually similar to the approach described here, although our implementations and methods differ in a number of important ways. First of all, Gupta, et al., did not explore the application of their method to EEG classification. Instead,

they tested their algorithm on a computer vision problem where objects are represented using features in the form of a time-series. A strong emphasis was also placed on the use of different initial context vectors for each ERNN. Our approach, in contrast, simply uses a zero initial context vector in combination with an initial transient period, as is explained further in Chapter 3. Gupta, et al., also explored the use of incremental training and training over concatenated sequences as opposed to our approach of resetting the initial context vector after each sequence. Finally, the details of the backpropagation algorithm used by Gupta, et al., for training their ERNN is left somewhat unclear.

Oeda, et al., have also briefly described an approach to time-series classification that is similar to the technique used here [25]. Described as an ensemble of ERNN, Oeda, et al., train a separate network to forecast sample time-series from each class. Classification is then performed by applying each forecaster and assigning the class label associated with the network that generated the lowest forecasting error. Oeda, et al., demonstrated that this approach can successfully classify artificially generated sinusoids. Additionally, they applied this approach to a small set of electrocardiographs and demonstrated that it is able to correctly identify the subject that produced the signals with an accuracy of about 74.7%. A detailed analysis of generalization performance and regularization, however, was not performed and this approach was not applied to EEG classification.

To our knowledge, Coyle, et al., are the only other group to apply a similar approach to EEG classification. In this approach, Coyle, et al., use feedforward networks in combination with TDE to perform single-step-ahead predictions of EEG signals [26]. Again, a separate network is trained over sample EEG from each class. The mean squared error and mean squared power of the predictions are then used as features that are passed to a Linear Discriminant Analysis classifier. A comparison is then performed with a similar approach that uses autoregressive models in place of neural networks and with an approach that uses autoregressive coefficients instead of forecasting errors. Their analysis was performed using EEG produced during imagined motor imagery and recorded during several separate sessions. Although it is difficult to draw firm conclusions from this analysis since different methods perform differently for different subjects, it seems safe to say that using forecasting errors as features appears to typically work at least as well as autoregressive coefficients and may be more reliable across multiple recording sessions.

Coyle, et al., have also published a number of works where a similar notion is utilized for preprocessing EEG [27, 28, 29, 30]. In this approach, termed Neural Time-Series Prediction Preprocessing (NTSPP), a feedforward network is again used in combination with TDE to forecast the EEG signals for each class a single step ahead in time. Instead of using the forecasting errors as features, however, the actual values of the forecast signal are used in the remaining steps of the classification pipeline. The intuition behind using forecasting as a preprocessing step is that components of the signal that are not very predictable may be noise or information that is not very useful for classification. In these studies, the application of NTSPP often results in a performance increase of roughly 1-10bpm.

The classification of time-series data, and specifically EEG, by forecasting with artificial neural networks is certainly not strictly original to this thesis. The relative sparsity of work directly related to this subject does, however, appear to suggest that it is quite novel and under explored. Numerous questions remain to be answered regarding the algorithms and parameters involved that may perform best. Additionally, the efficacy of this approach and its robustness to the noisy and non-stationary nature of EEG signals remains to be determined.

Chapter 3

Methods

In this chapter we describe the details of all the algorithms and experiments that are used throughout the remainder of this thesis. We begin by describing five EEG datasets used to analyze the performance of our forecasting and classification algorithms. We then provide a detailed description of ERNN and the methods we use for training them. Next, we formalize how our ERNN are used to forecast EEG signals a single step ahead in time. Finally, we describe how our forecasting errors can be used to perform EEG classification and outline three methods for achieving this.

3.1 Experimental Data

In the following experiments we examine five EEG datasets recorded from five different subjects. Subjects A and B are both able-bodied while Subject C has quadriplegia due to a complete lesion at vertebra C4, Subject D has severe multiple sclerosis and Subject E has quadriplegia due to a complete lesion at vertebra C1. For subjects A and B, EEG recording was performed in a controlled laboratory environment while recording from Subjects C, D and E was performed in the subject's home environment in order to more closely replicate the conditions under which a BCI might be used.

Each subject was presented with a visual cue on an LCD screen requesting them to perform one of four imagined mental tasks for the duration of the cue. A five-second break was alloted between each sequence during which the subject was instructed to relax. Presentation of the visual cues and data collection were performed using custom software. For Subjects A and B, the following imagined mental tasks were used:

imagined clenching of right hand, imagined shaking of left leg, silently counting backward from 100 by 3's and visualization of a spinning cube. Ten sequences each lasting five seconds were recorded during each mental task in a randomized order for a total of 200 seconds of EEG data. The protocol used for Subjects A and B is summarized in Table 3.1.

Table 3.1: Experiment protocol for Subjects A and B

Task Id	Duration	Cue	Mental Task
0	10x5s	(none)	Resting state
1	10x5s	Right Hand	Imagine clenching right hand into a fist
2	10x5s	Left Leg	Imagine shaking left leg
3	10x5s	Math	Silently count backward from 100 by 3's
4	10x5s	Spin	Visualize a spinning cube

Table 3.2: Experiment protocol for Subjects C, D and E.

Task Id	Duration	Cue	Mental Task
0	5x10s	(none)	Resting state
1	5x10s	Count	Silently count backward from 100 by 3's
2	5x10s	Fist	Imagine clenching right hand into a fist
3	5x10s	Cube	Visualize a spinning cube
4	5x10s	Song	Silently sing a favorite song

A slightly different experimental protocol was used for Subjects C, D and E, summarized in Table 3.2. These differences in protocol are due to the fact that the data for Subjects C, D and E were borrowed from a larger and ongoing project designed to test the feasibility of using BCI technology with disabled people in their homes. For these subjects the following imagined mental tasks were used: silently counting backward from 100 by 3's, imagined clenching of right hand into a fist, visualization of a spinning cube and silently singing a favorite song. Note the only imagined mental task that is different between across all of the datasets is imagined left leg movement versus silently singing a song. Five sequences each lasting ten seconds were recorded during each mental task for a total of 200 seconds of EEG data. In order to make the datasets recorded from Subjects C, D and E as similar as possible to those recorded from Subjects A and B, each ten second sequence is split in half to yield ten sequences each lasting five seconds.

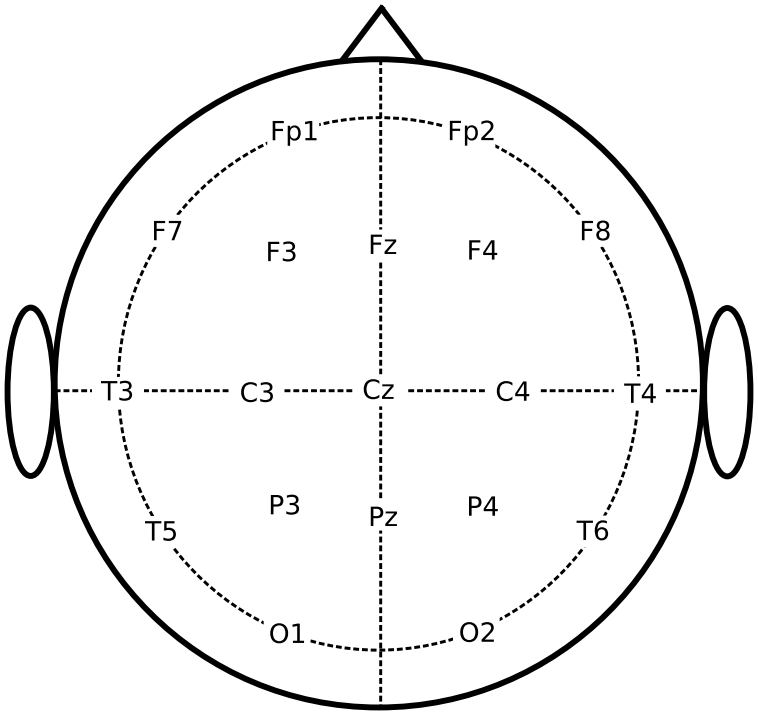


Figure 3.1: 19 channel subset of the 10-20 system used for electrode placement.

All five datasets were recorded using a relatively inexpensive Neuropulse Mindset-24 amplifier [31]. An Electrocap using the 19 channel 10-20 system, shown in Figure 3.1, with linked earlobe references was used for electrode placement [32]. The Mindset-24 contains a hardware Sallen-Key bandpass filter with a passband of 1.5-34Hz. For Subjects A and B, the data was recorded with a sampling rate of 256Hz while the data recorded from Subjects C, D and E was initially recorded at 512Hz and later downsampled to 256Hz. This leaves our sampling rate at roughly four times the Nyquist rate. Channel F8 was ultimately discarded from all five datasets due to a poor connection during recording from Subjects C, D and E, leaving a total of 18 channels.

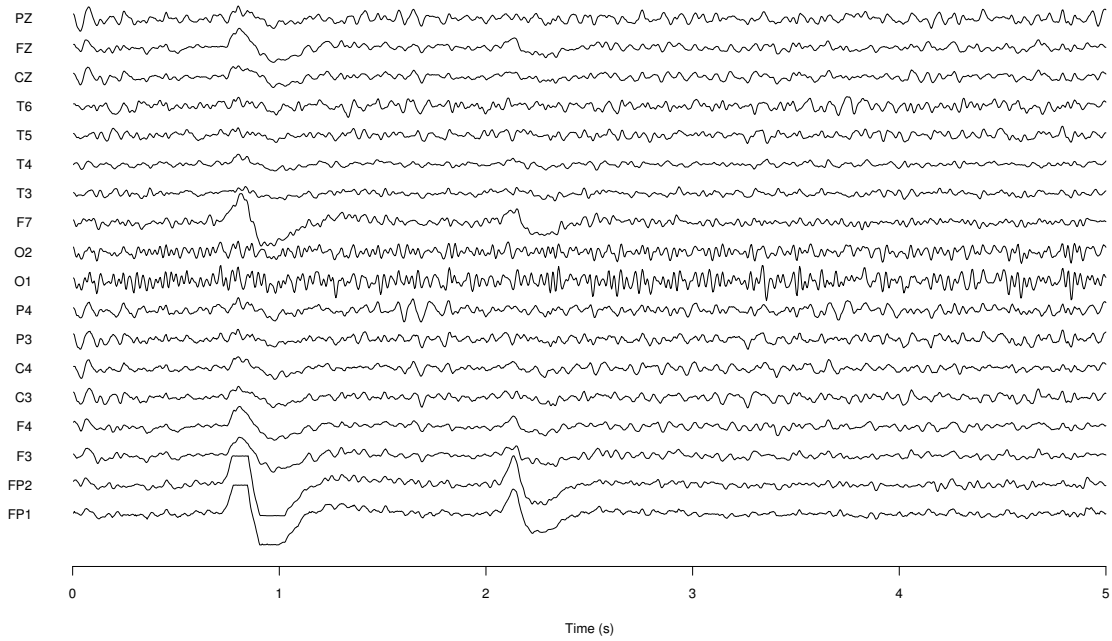
After data acquisition, a software Butterworth bandstop filter with an 18Db per octave roll-off and a stopband of 59-61Hz was applied to remove any remaining 60Hz noise introduced by alternating-current electrical sources in the environment. A Maximum Noise Fraction (MNF) filter was then trained for each subject using a five-second EEG sequence that was found upon visual inspection to contain an eyeblink artifact [33, 34, 35]. The single MNF component with the lowest frequency was then removed and the filter was applied to each sequence in order to attenuate ocular artifacts. Next, the mean signal amplitude

across all channels is subtracted from each channel individually, known as a common average reference, in order to decorrelate the channels. The MNF filter and common average reference are used because our experience has shown relatively small but noticeable improvements in classification accuracy across most of the five subjects when these preprocessing steps are included. Finally, each channel was shifted and scaled to have zero mean and unit standard deviation for reasons that will be further discussed in Section 3.2.2. In Figure 3.2 we see a sample five-second EEG recording from Subject A before and after preprocessing. Notice that the ocular artifacts seen at roughly one and two seconds are essentially removed by the MNF filter.

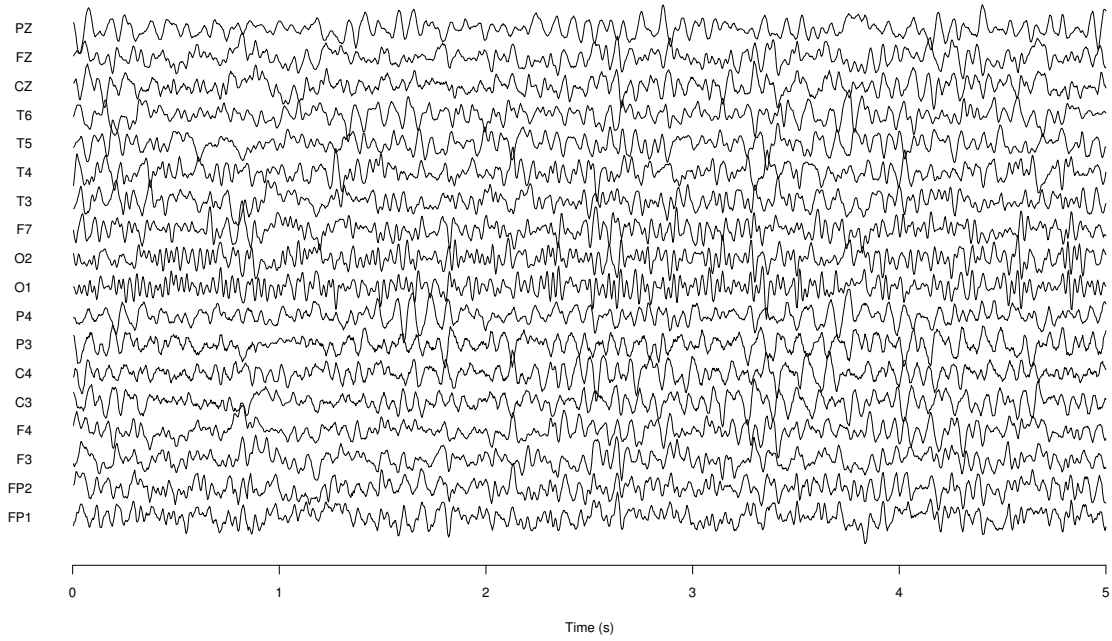
In order to simplify our initial analysis, we first consider a two-task subset of our imagined mental tasks consisting only of the imagined right hand movement and count backward from 100 by 3's tasks. These two tasks were chosen because they were the first two tasks that are common among all five datasets. In Chapter 4 we present our final classification results for both the two-task and four-task problems.

To ensure that all of the experiments performed here more accurately represent the way we can expect our classifiers to perform in the real world, we partition each dataset into a 60% training partition, consisting of the first six sequences, and a 40% test partition, consisting of the remaining four sequences. Unless otherwise noted, we use a six-fold cross-validation procedure over the training partition for all model exploration and hyper-parameter tuning. In other words, a separate model or classifier is trained for each of the six ways, or folds, that five sequences can be chosen from the training partition. Training performance is evaluated by averaging performance over each of these folds. Validation performance, i.e., the performance used to tune hyper-parameters, is evaluated by applying each of these models or classifiers to the sequence in the training partition that was excluded from the given fold and then averaging the results.

Generalization performance is tested, after all hyper-parameter tuning is complete, by training a separate model or classifier over each of the six folds and evaluating each classifier over the entire test partition and, finally, averaging the results. Testing is done in this fashion in order to simulate how our classifiers would perform in a real-world setting. That is, once the classifier is trained, the remaining EEG signals would be utilized by a BCI user in order to issue commands to a computerized device.



(a) Sample EEG sequence before preprocessing.



(b) Sample EEG sequence after preprocessing.

Figure 3.2: A five second electroencephalogram before and after preprocessing.

3.2 Elman Recurrent Neural Networks

The ANN architecture that we explore for modeling and classifying EEG is the Elman Recurrent Artificial Neural Network (ERNN). ERNN were originally developed by Jeffrey Elman in 1990 for finding patterns in natural language and have since been successfully applied to a number of practical problems [4]. In addition to their history of successful application, ERNN are appealing because it has been demonstrated that they are universal approximators of finite state machines [36]. In other words, any given finite state machine can be simulated by some ERNN given enough hidden units and the proper connection weights.

3.2.1 Architecture

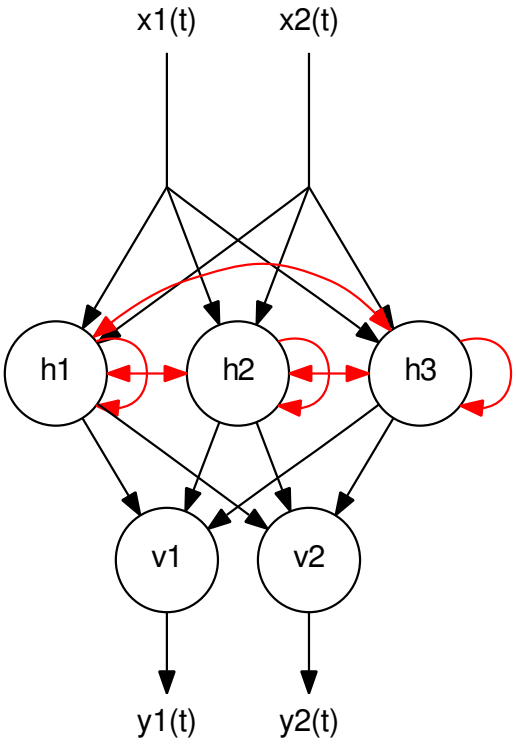


Figure 3.3: An ERNN with two inputs, $\mathbf{x}$, three hidden units, $\mathbf{h}$, two visible units, $\mathbf{v}$, and two outputs, $\mathbf{x}$. Note that the hidden layer has full recurrent connections.

An ERNN, shown in Figure 3.3, consists of two distinct layers. The first layer, referred to as the hidden layer, is composed of a number of neurons with sigmoidal activation functions. The number of neurons in the hidden layer, also referred to as hidden units, is a parameter given during the construction of an ERNN.

Each hidden unit has full incoming connections from each input as well as a constant bias value of one. Additionally, each hidden unit has full recurrent connections between every other hidden unit with a single timestep delay. In other words, the current value of each network input and the output of every hidden unit at the previous timestep is fed into each hidden unit at the current timestep.

The values stored in the delay lines of the recurrent connections of an ERNN fully represent the state of the network at any given time. As such, we refer to these values as the context of the network. As is common practice, the initial context of our ERNN is set to the zero vector with the assumption that the network will sufficiently acclimate to the input signal and achieve an acceptable context after some initial transient period [4].

The second layer, referred to as the visible layer, consists of one neuron per output. Each neuron in the visible layer, also referred to as a visible unit, is strictly linear, i.e., the output of each visible unit is a weighted sum of the outputs of the hidden units and a constant bias value of one.

In order to formalize the ERNN architecture, let us begin by defining the following constants

$$\begin{aligned} L: &\quad \text{the number of network inputs} \\ M: &\quad \text{the number of hidden units} \\ N: &\quad \text{the number of network outputs} \end{aligned}$$

Next, let $\mathbf{x}(t)$ be the $L \times 1$ vector of inputs to our network at time t and $\mathbf{z}(t)$ be the $M \times 1$ output of our hidden layer at time t. Also, let $\bar{\mathbf{x}}(t)$ be $\mathbf{x}(t)$ with a constant one appended and $\bar{\mathbf{z}}(t)$ be $\mathbf{z}(t)$ with a constant one appended for our bias values. Then the output of our hidden layer at time t can be defined by the following recurrence relation

$$\mathbf{z}(t) = \phi(\mathbf{H}\bar{\mathbf{x}}(t) + \mathbf{S}\mathbf{z}(t-1)), \tag{3.1}$$

where $\mathbf{H}$ is the $M \times (L+1)$ matrix of feedforward weights in the hidden layer, $\mathbf{S}$ is the $M \times M$ matrix of recurrent weights in the hidden layer and ϕ is our choice of sigmoid. Note that we define our initial context $\mathbf{z}(0) = \mathbf{0}$ as discussed earlier. Finally, the output of our visible layer at time t can be defined as

$$\mathbf{y}(t) = \mathbf{V}\bar{\mathbf{z}}(t), \tag{3.2}$$

where $\mathbf{V}$ is the $N \times (M+1)$ weight matrix for our visible layer. For the sake of notational brevity, we denote the output of an ERNN at time t as

$$\mathbf{y}(t) = \text{ernn}(\mathbf{x}(t)). \tag{3.3}$$

3.2.2 Sigmoid and Initial Weights

When working with ERNN it is extremely important to use a proper sigmoid and weight initialization scheme. This is important for two reasons. First of all, choosing initial weights that are not in a reasonable region of the weight space results in large changes in the weights during training, resulting in slow convergence. Second, if the initial weights are chosen so that the sigmoid becomes immediately saturated, i.e., the sum of the inputs lands on the tails of the sigmoid, then the gradient approaches zero for the affected neuron. This can cause slow convergence and increases the likelihood of falling into a local minima before escaping the saturated region of the sigmoid. Indeed, our experience has shown that neglecting these details leads to dramatically increased training times as well as an increased frequency of encountering local optima. In order to avoid these pitfalls, we have extended the suggestions for fast training of Feedforward Networks made by LeCun, et al., to ERNN [37, 38].

LeCun's first suggestion is to standardize the network inputs and outputs to have zero mean and unit standard deviation. Furthermore, they suggest that it is desirable to keep the outputs at each layer of the network near zero mean and unit standard deviation. Keeping our average inputs and outputs near zero reduces the learning of unnecessary bias and keeping the standard deviation near unity aids in our initial weight selection.

In order to achieve these goals, LeCun, et al., first suggest the use of a symmetric sigmoid where $\phi(\pm 1) = \pm 1$, the maxima of the second derivatives of ϕ are at ± 1 and where the gain is close to unity in $[-1, 1]$. The constants in

$$\phi(\mathbf{x}) = 1.7159 \tanh\left(\frac{2}{3}\mathbf{x}\right) \tag{3.4}$$

are chosen to approximately possess each of these properties. In short, equation (3.4) is chosen to retain its non-linear properties while still mapping the weighted sum of inputs to nearly the same output over its normal operating range of $[-1, 1]$.

Finally, LeCun, et al., suggest that our initial weights be drawn from a random uniform distribution with

$$\sigma = \kappa^{-1/2}, \tag{3.5}$$

where κ is the number of incoming connections, or fan-in, for the given hidden unit and σ is standard deviation of the initial weights. Assuming uncorrelated inputs and unit standard deviation, equation (3.5) in combination with (3.4) yields outputs that approximately have unit standard deviation. Given the applications that we explore here, however, we have found that equation (3.5) still yields initial weights that are large enough to cause saturation. As such, we scale equation (3.5) down by $2/3$. Ultimately, the initial values for our hidden weights, H_0, are drawn from

$$H_0 \sim U\left(-\sqrt{2/(L+M+1)}, \sqrt{2/(L+M+1)}\right), \tag{3.6}$$

where $U(a, b)$ is the random uniform distribution between a and b. Similarly, since the linear transfer function has zero gain, the initial values for our visible weights, V_0, are drawn from

$$V_0 \sim U\left(-\sqrt{2/(M+1)}, \sqrt{2/(M+1)}\right). \tag{3.7}$$

Ultimately, combining input and output standardization with a carefully chosen sigmoid function and drawing our initial weight values from the corresponding distribution, we achieve an initial ERNN with hidden units that are not saturated and where the input and output at each layer is near zero mean and unit standard deviation.

3.2.3 Training

In order to train our ERNN we use a form of batch gradient descent. It is not a straightforward task, however, to compute the error gradient for an ERNN due to the recurrent connections found in the hidden layer. Nevertheless, it can be achieved through a process known as Backpropagation Through Time (BPTT) [39, 5]. In order to computer the error gradient for the hidden layer of an ERNN, we must first unroll the network through time. This process is illustrated in Figure 3.4. Notice that by duplicating our hidden and

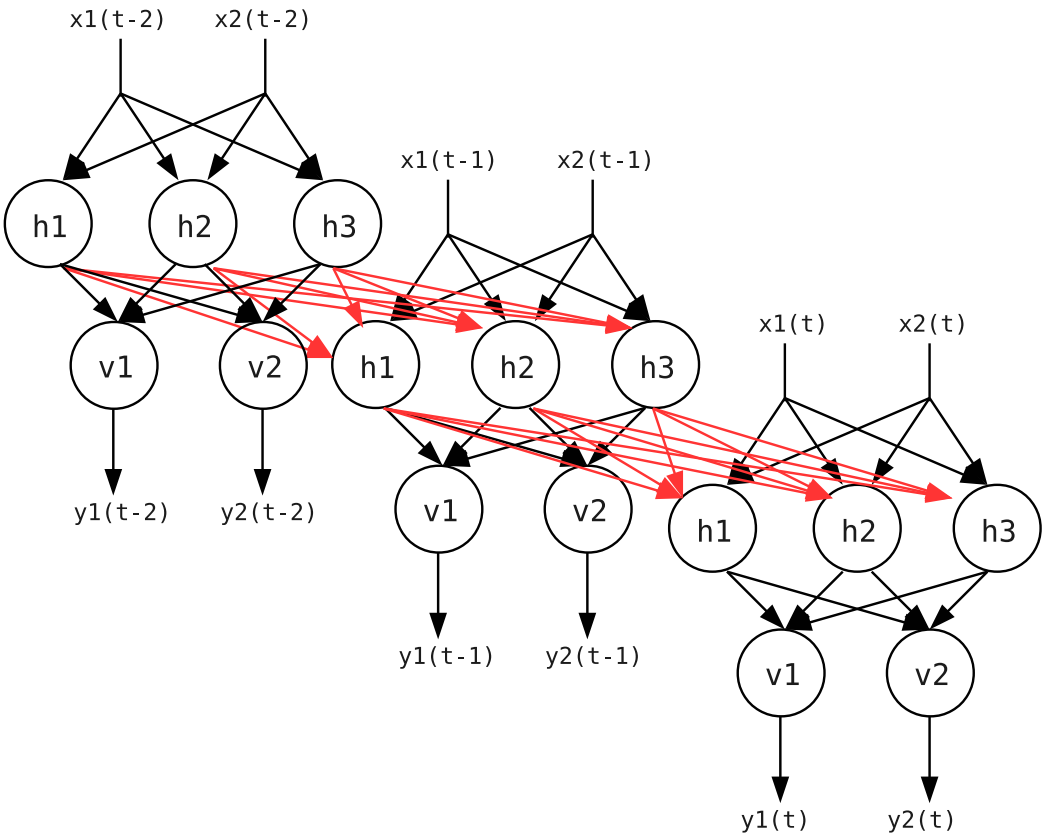


Figure 3.4: An ERNN unrolled three steps through time.

visible layers and attaching them to the inputs and outputs of our training sequence at previous timesteps, we are able to construct a completely feedforward network, albeit a relatively complex one. If we unroll our ERNN over the entire training sequence, we can compute the error gradient directly using the same techniques commonly used for feedforward networks.

Unfortunately, unrolling an ERNN quickly becomes computationally intractable with long training sequences, such as the EEG signals that we wish to investigate here. In order to work around this problem, we simply unroll our ERNN a given number of steps, say ν, before truncating the trailing layers of the network. This process is known as Truncated Backpropagation Through Time and yields an approximation of the true error gradient.

Although the full derivation of BPTT is too lengthy to be presented here, we describe this method by presenting its final form. First, let ξ be the error between the outputs of the network and the target training values. Next, let and let $\delta(t)$ be the $N \times 1$ column vector of contributions to our error for each network output at time t. The ξ and δ used in the experiments conducted here will be thoroughly described in Section 3.3. Additionally, let

$$\mathbf{b}(t) = \phi'(\mathbf{H}\bar{\mathbf{x}}(t) + \mathbf{S}\mathbf{z}(t-1)) \tag{3.8}$$

where ϕ' is the derivative of our sigmoid. First, we seek to find our error gradient with respect to our hidden weight matrix. Using the chain rule for derivatives and by unrolling the recursive $\mathbf{z}$ term in $\mathbf{b}$, we can approximate our error gradient with respect to the hidden weight matrix as

$$\nabla_{\mathbf{H}}\xi \approx \sum_{t=t_0}^{\tau} \sum_{i=t}^{\min(\tau,t+\nu)} \left[\underline{\mathbf{V}}^T \delta'(i) \cdot \mathbf{b}(i) \cdot \prod_{j=t+1}^{i} \mathbf{S}\mathbf{b}(j) \right] \bar{\mathbf{x}}^T(t), \tag{3.9}$$

where t_0 is the first timestep in our sequence, τ is the final timestep in our sequence, ν is the number of steps through time to unroll our network before truncation, $\underline{\mathbf{V}}$ is our visible weight matrix with the bias column removed and $\cdot$ denotes point-wise matrix multiplication. Similarly, the error gradient with respect to the recurrent weight matrix can be approximated by

$$\nabla_{\mathbf{S}}\xi \approx \sum_{t=t_0}^{\tau} \sum_{i=t}^{\min(\tau,t+\nu)} \left[\underline{\mathbf{V}}^T \delta'(i) \cdot \mathbf{b}(i) \cdot \prod_{j=t+1}^{i} \mathbf{S}\mathbf{b}(j) \right] \mathbf{z}^T(t-1). \tag{3.10}$$

Finally, the gradient for our visible layer can be computed in the same batch fashion as would be used for a two-layer feedforward network,

$$\nabla_{\mathbf{V}}\xi = \sum_{t=t_0}^{\tau} \delta'(t)\bar{\mathbf{z}}(t)^T. \tag{3.11}$$

Once the error gradients for our network have been approximated, we iteratively update the network's weight values using Scaled Conjugate Gradients (SCG). SCG is a fast gradient descent algorithm pioneered by Møller that estimates second-order gradient information [40]. Since training ERNN with BPTT requires the propagation of error gradients through many layers, this second-order information may yield faster convergence rates than other methods. Furthermore, the fact that SCG exploits some of the benefits of second-order methods without requiring second derivatives to be explicitly computed, it seems to be an ideal candidate for training ERNN. Although it is uncommon to use SCG to train Recurrent Neural Networks, Gruber and Sick report that SCG outperforms RProp and Memoryless Quasi-Newton with their DYNN recurrent architecture [41].

3.3 Forecasting

The feedback loops found in ERNN and their dynamic nature makes them a promising candidate for short-term, non-linear time-series forecasting. This notion is a cornerstone of the techniques described here and, as such, we begin describing our technique for EEG classification by formalizing a forecasting problem. In this forecasting problem, we utilize an ERNN with $L = 18$ inputs as well as $N = 18$ outputs. We then train our ERNN to model EEG signals by forecasting the value of each of the 18 channels a single step ahead in time.

In order to train our ERNN to forecast EEG signals, we first seek an error measure that is suitable for use with SCG and BPTT, as described in Section 3.2.3. There are a number of things to consider when designing this error measure. First of all, it should increase forecasting performance upon minimization. Second, it must be differentiable at each timestep in order to calculate the error gradient. Furthermore, we must omit the error incurred during an initial transient period. As mentioned in Section 3.2.1, this is due to the fact that our initial context is set to the zero vector. Since it is unlikely for this initial context to align with the learned network response during the initial values of the EEG signal, we ignore the error at the

beginning of each sequence. With this initial transient period, our network is able to acclimate to the signal before being penalized. A relatively short initial transient period of $\rho = 32$ timesteps, or $1/8$ of a second with a sampling rate of 256Hz, was empirically determined to be sufficient.

In order to meet these requirements, we choose to minimize the mean squared error (MSE) between the current outputs of the network and the values of the EEG signal at the following timestep less the outputs during the initial transient period. Our individual error contributions at time t are then the average squared errors,

$$\delta(t) = \frac{(\mathbf{y}(t) - \mathbf{x}(t+1))^2}{(T-1-\rho)N}, \tag{3.12}$$

where $\mathbf{x}(\mathbf{t})$ is the vector of EEG potentials at time t, $\mathbf{y}(\mathbf{t})$ is the vector of outputs from our ERNN at time t and T is the total number of timesteps in the EEG sequence. Note that we can substitute equation (3.12) into equations (3.9), (3.10) and (3.11) by letting $\tau = T - 1$ and $t_0 = \rho$. Our error measure is then the scalar MSE across all channels and all timesteps from the end of our initial transient period to the end of our EEG sequence,

$$\xi = \sum_{t=\rho}^{T-1} \sum_{n=1}^{N} \delta_n(t). \tag{3.13}$$

It should also be noted that, in practice, our training data consists of several disconnected EEG sequences. In order to account for this, we restart our ERNN at the beginning of each sequence and stop the network at the end of each sequence. Thus, our error is actually found by computing the squared errors at each timestep across all sequences, except during the transient periods, and then dividing by the combined length of the sequences less the length of the initial transient periods. For notational simplicity, however, we restrict ourselves to the error measure in equation (3.13).

In order to provide a simple benchmark against which our forecasting performance can be evaluated and in order to demonstrate that our ERNN are learning more than trivial means of forecasting EEG, we also define two naive error measures. We call our first naive error measure the naive repeated, or Naive-R, error which is defined as

$$\textit{Naive-R} = \frac{1}{(T-1-\rho)N} \sum_{t=\rho}^{T-1} \sum_{n=1}^{N} (x_n(t) - x_n(t+1))^2 \tag{3.14}$$

and is equivalent to forecasting by simply repeating the previous input. We call our second naive error measure the naive interpolated, or Naive-I, error which is defined as

$$\textit{Naive-I} = \frac{1}{(T-1-\rho)N} \sum_{t=\rho}^{T-1} \sum_{n=1}^{N} (l_n(t) - x_n(t+1))^2 \tag{3.15}$$

where

$$\mathbf{l}(t) = \mathbf{x}(t) + (\mathbf{x}(t) - \mathbf{x}(t-1))$$

and is equivalent to forecasting the EEG signal by performing a linear interpolation of the previous two inputs.

In Section 4.1.1, we illustrate how our forecasting errors change as we vary each of our hyper-parameters. Since we are using Mean Squared Errors and because our EEG signals are scaled to have zero mean and unit standard deviation, we seek a more intuitive metric for analyzing our forecasting errors. As such, we will also present our forecasting errors as a percent of signal range (psr) defined as

$$\textit{percent signal range} = 100 \, \frac{\textit{RMSE}}{(\textit{max signal}) - (\textit{min signal})} \tag{3.16}$$

where *RMSE* is the root mean squared error, i.e., the square root of our MSE, and where *max signal* and *min signal* are the largest and smallest values across all channels and all timesteps in our standardized EEG signal.

3.4 Classification by Forecasting

Now that we have outlined a method for training ERNN to forecast EEG, we present a classification scheme based on these forecasters. The training procedure for our classification algorithm, summarized in Figure 3.5, entails the training of a separate ERNN to forecast sample EEG signals from each class. Thus, if we have K imagined mental tasks, we must apply BPTT and SCG to K different ERNN. Each of these ERNN can then be considered an expert at forecasting each type of EEG. To formalize this notion, we describe an ERNN trained to forecast sample EEG signals from class $k \epsilon \{1, 2, \ldots, K\}$ as

$$\mathbf{y}^k(t) = \text{ernn}^k(\mathbf{x}(t)), \tag{3.17}$$

where $\mathbf{y}^k(t)$ is the output of network k at timestep t. Once an ERNN has been trained to forecast each class of EEG, our forecasting errors can be thought of as a feature vector with the k^{th} element being

$$e_k(t) = \frac{1}{N}\sum_{n=1}^{N}(y_n^k - x_n(t+1))^2 \tag{3.18}$$

for each ERNN k. Finally, we train a classifier to map our error features to class labels. In Sections 3.4.1 3.4.2 and 3.4.3 we will describe three potential classifiers for mapping our error features to class labels.

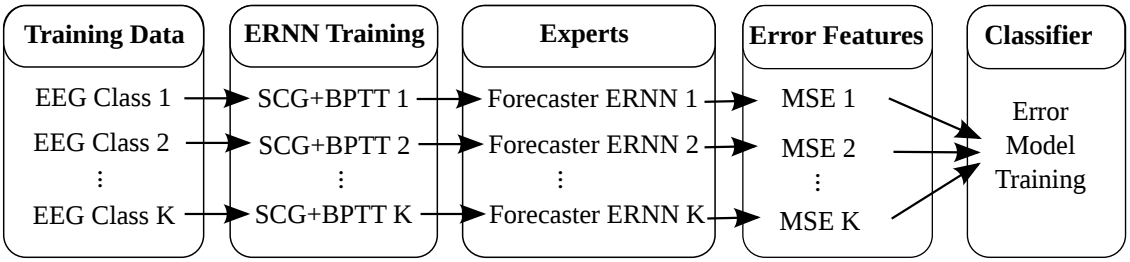


Figure 3.5: Classification training procedure.

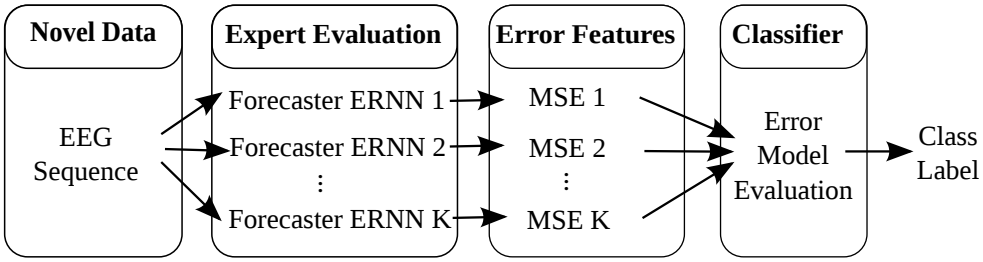


Figure 3.6: Classification testing procedure.

In order to to assign a class label to novel data, summarized in Figure 3.6, we retain each of our ERNN forecasters and our error classifier from the training procedure. Each ERNN is then applied to the new EEG sequence and the resulting forecasting errors are recorded. Finally, the error features are passed to the error classifier which assigns a class label.

3.4.1 Averaged Winner-Takes-All

A simple approach to assigning class labels is to let the winner take all (WTA). This classifier assumes that the lowest forecasting errors for a previously unseen EEG sequence will be produced by the ERNN that was trained to forecast EEG belonging to the same class. Using this insight, we can simply assign the class

label associated with the ERNN that produced the lowest forecasting MSE. There are, however, two things that we consider when taking this approach. First, we seek to classify EEG in a manner that is useful in the context of BCI. Since our sampling rate is is much faster than a BCI user can issue commands in a conscious and purposeful manner, we do not wish to assign class labels at every timestep. Second, EEG signals are both noisy and often quite periodic. As a result, it is common to see brief spikes in our forecasting error that can be smoothed out by averaging our errors over several periods. These issues suggest that our WTA classification scheme should average the forecasting errors over a short window before assigning a class label.

To formally define this approach, we first define the length of a short window, say ω, over which we average our forecasting MSE. Following equation (3.18), our MSE averaged over each interval becomes

$$\hat{e}_k^i = \frac{1}{\omega} \sum_{t=\rho+i\omega}^{(i+1)\omega-1} e_k(t) \tag{3.19}$$

for the i^{th} interval. Finally, our class label C^i at interval i is assigned to be

$$C^i = \underset{c\epsilon\{1,2,\ldots,K\}}{\operatorname{argmin}} \hat{e}_c^i. \tag{3.20}$$

3.4.2 Quadratic Discriminant Analysis

Although the WTA approach is quite intuitive, it is possible that our forecasting errors will not be so easily interpreted. For example, it may be possible that EEG belonging to some class simply better lends itself to forecasting than another. In this case, our forecasting errors would be biased toward one class, causing WTA to fail. It may also be possible that large forecasting errors are indicative of some classes and not others. In an attempt to address these possibilities, we also consider classifying our errors using two generative, statistical classifiers.

The first of these two classifiers that we explore is known as Quadratic Discriminant Analysis (QDA) [42]. When using QDA, we assume that the class conditional probability density of our error features for class c can be modeled using a multivariate Gaussian

$$P(\mathbf{e}|C=c) = \frac{1}{(2\pi)^{K/2}|\Sigma_c|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{e}-\mu_c)^T\Sigma_c^{-1}(\mathbf{e}-\mu_c)\right)}, \tag{3.21}$$

where Σ_c is the sample covariance matrix for our error vectors belonging to class c and μ_c is the sample mean for our error vectors belonging to class c. It is then a relatively straight forward exercise using Bayes' rule to find our posterior probability density $P(C=c|\mathbf{e})$. We then notice that several terms can be canceled when comparing two classes by applying a logarithm to each side $\log P(C=c_1|\mathbf{e}) < \log P(C=c_2|\mathbf{e})$. Thus, it is sufficient to find only the largest of our discriminant functions

$$q_c(\mathbf{e}) = -\frac{1}{2}|\Sigma_c| - \frac{1}{2}(\mathbf{e}-\mu_c)^T\Sigma_c^{-1}(\mathbf{e}-\mu_c) + \log(\gamma), \tag{3.22}$$

where the scalar $\gamma = \frac{\textit{samples in class c}}{\textit{total samples}}$ comes from our prior distribution. Note that the decision boundaries between classes is quadratic. Again, we desire to assign class labels only after an interval of ω. To achieve this we sum the values of our discriminant function for each class over this interval so that

$$\hat{q}_c^i = \sum_{t=\rho+i\omega}^{(i+1)\omega-1} q_c(\mathbf{e}(t)). \tag{3.23}$$

Since we are working with the log of the probability densities, this is equivalent to assuming that our errors are independent at each timestep and finding the joint probability that all samples in our window belong to class c. Our class labels are then chosen by finding the discriminant function that maximizes the joint probability

$$C^i = \underset{c\epsilon\{1,2,\ldots,K\}}{\operatorname{argmax}} \hat{q}_c^i. \tag{3.24}$$

3.4.3 Linear Discriminant Analysis

The last approach that we consider for classifying our forecasting errors is known as Linear Discriminant Analysis (LDA) [42]. LDA is also a generative, statistical classifier that assumes our classes can be modeled with Gaussians. In fact, LDA is derived in the same fashion as QDA with the caveat that we assume a common averaged covariance matrix. The fact that we assume identical covariances leads to more terms canceling out and yields the following discriminant functions

$$l_c(\mathbf{e}) = \mathbf{e}^T \Sigma^{-1} \mu_c - \frac{1}{2} \mu_c^T \Sigma^{-1} \mu_c + \log(\gamma), \tag{3.25}$$

where Σ is the sample covariance matrix averaged across the samples in all classes. Note that the decision boundaries between classes are now linear.

Again, we sum the values of l over our interval i under the assumption that our errors are independent

$$\hat{l}_c^i = \sum_{t=\rho+i\omega}^{(i+1)\omega-1} l_c(\mathbf{e}(t)) \tag{3.26}$$

and, finally, assign our class label based on the highest probability of joint class membership of our errors over the interval i

$$C^i = \underset{c \epsilon \{1,2,\ldots,K\}}{\operatorname{argmax}} \hat{l}_c^i \tag{3.27}$$

Although LDA may seem more restricted than QDA, its linear decision boundaries may make it less susceptible to over-fitting, particularly with noisy data. Additionally, the fact that our sample covariances are averaged over all classes may yield a better estimate of the true covariances, especially if some or all of the classes are undersampled.

Chapter 4

Results and Discussion

In this chapter we present the results of the experiments we outlined in Chapter 3 as well as offer some discussion and speculation pertaining to these outcomes. We begin by revisiting our forecasting problem in order to explore the hyper-parameters involved and determine values that achieve the best forecasting results. Next, we conduct an experiment designed to explore the amount of temporal information that our ERNN are capturing. In this experiment we place a feedback loop between the input and output layers of a trained ERNN to form an iterated model. Upon the conclusion of our forecasting experiments, we revisit our classification problem. Again, we begin by exploring the relevant hyper-parameters, namely the effect of the number of hidden units on regularization. Next, we apply each of our classification algorithms to the datasets described in Section 3.1. First, we present these outcomes as classification accuracies over one-second intervals. Then, we present our classification results in terms of information transfer rates. Finally, we offer some analysis of the error classification boundaries that are learned by WTA, LDA and QDA and the effects of varying the rate at which class labels are assigned.

4.1 Forecasting

In this section we explore the ability of our ERNN to forecast EEG. Since the results presented here are very similar for all five subjects, we only present the forecasting results applied to the EEG recorded from Subject A. Unless otherwise noted, all of the experiments performed here utilize the six-fold cross-validation

procedure explained in Section 3.1. To prevent ourselves from introducing any bias in our later experiments, we only look at the training and validation errors during our forecasting experiments.

4.1.1 Parameter Selection

Thus far, our description of ERNN and our forecasting problem has left us with several unexplored hyper-parameters. Namely, the number of steps through time that we should unroll for BPTT, the number of training epochs that should be used with SCG and the number of hidden units that we should use in our ERNN. In this section we explore each of these hyper-parameters in turn and find appropriate values to be used in the remainder of our experiments and analysis.

First, we should point out that regularization of our ERNN in all subsequent experiments is performed by controlling the number of hidden units. In other words, we limit the number of hidden units available to our ERNN in order to prevent them from learning noisy or overly complex patterns that do not generalize well to new data. Although it may be possible to regularize ERNN using other techniques, such as early-stopping or weight penalties, our experience has demonstrated that limiting the number of hidden units typically provides better generalization than limiting the number of training epochs. Furthermore, using as few hidden units as possible also considerably improves the computational performance of training and evaluating ERNN since the asymptotic runtime for both the forward and backward passes grow quadratically as the number of hidden units increases but only linearly as the number of training epochs increases. Additionally, avoiding weight penalties reduces the number of training parameters that must be explored. Further research may be necessary, however, in order to more clearly define the effectiveness of other regularization techniques in this setting.

Before determining the optimal number of hidden units to use in our ERNN, we must first determine a suitable number of steps to unroll our network during BPTT gradient approximation and a sufficient number of SCG epochs to perform before terminating the training process. In order to find values for these parameters that work well, we first perform a coarse grid search and then fine tune each parameter individually.

In Figure 4.1 we see how the average ERNN forecasting error as a percent of signal range (psr) varies as the number of steps that our network is unrolled is increased. Here, the number of hidden units is fixed

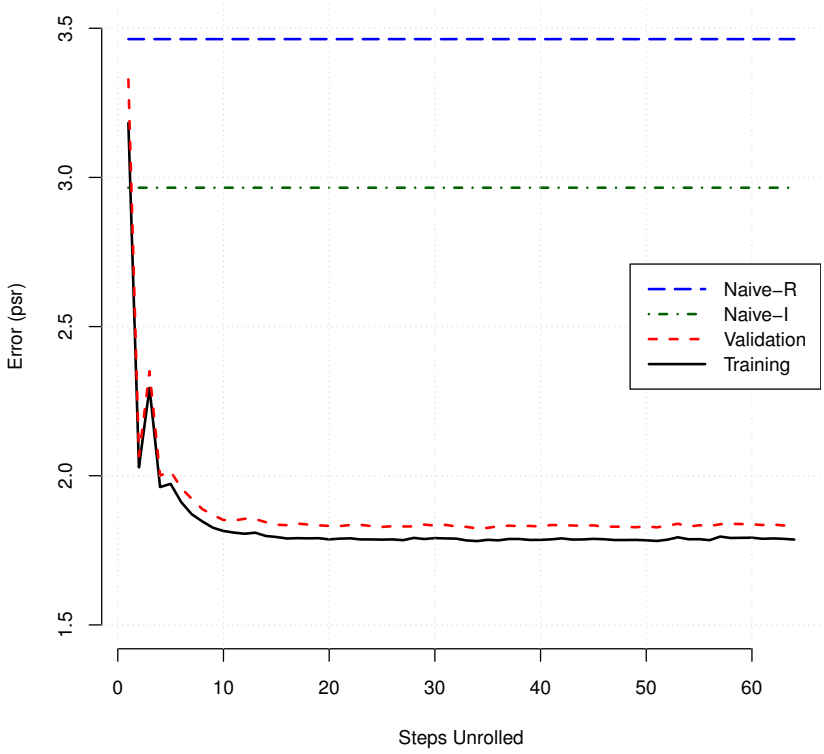


Figure 4.1: Forecasting error as the number of steps unrolled in BPTT is varied.

at 15 and the number of SCG training epochs is fixed at 250. Both the training and validation forecasting errors beat the Naive-R error with only one step unrolled, i.e., no optimization of the recurrent weights. This suggests that purely spatial information may be enough to do better than simply repeating the previous signal value. The Naive-I error is easily beat with somewhere between two and five steps unrolled suggesting that the additional temporal information continues to improve forecasting performance. The forecasting error continues to decrease as the number of steps unrolled increases until leveling off around 15-20 steps unrolled.

It appears that there is a fair amount of variability in the average performance when the number of steps unrolled is less than about five. In Figure 4.1 we see a large spike in error with 3 steps unrolled. This suggests that reliable training of ERNN is difficult to achieve when BPTT is unrolled only a few steps. This may be explained by frequent encounters with local optima during the training process caused by insufficient

training of the recurrent weights. If our recurrent weights are left mostly unoptimized, then the performance of our ERNN may largely depend on our choice of initial weights, which is essentially random.

In the remaining ERNN experiments, we always unroll our networks 20 steps. This value is chosen to be somewhat on the conservative side for two reasons. First, the experiment shown in Figure 4.1 is only a representative sample of the potential network configurations that we explore. It is possible, for example, that networks with a larger number of hidden units may require that the network is unrolled more steps in order to be fully optimized during training. Second, there is a small but observable decrease in validation error from 15 to 20 steps unrolled. Since our forecasting errors are ultimately quite small and since regularization of our networks is performed by limiting the number of hidden units, we do not wish to introduce additional forecasting error by choosing a value for the number of steps unrolled that is too small.

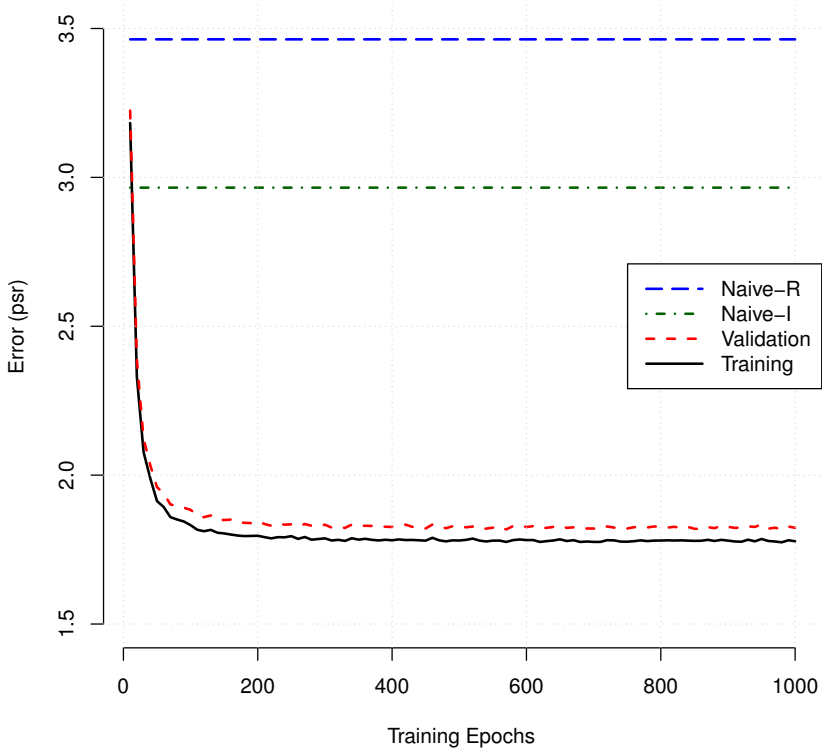


Figure 4.2: Forecasting error as the number of SCG training epochs is varied.

Next, we investigate how our forecasting error changes as the number of SCG training epochs is varied. In Figure 4.2 we see that both the training and validation errors easily beat the Naive-R solution with fewer than 10 training epochs. The ERNN forecaster also beats the Naive-I solution after roughly 20 training epochs. The error then continues to fall until leveling off with a error of approximately 1.79psr near 200 training epochs. Again, we choose to use a conservative value of 250 training epochs to ensure that regularization is controlled strictly by the number of hidden units.

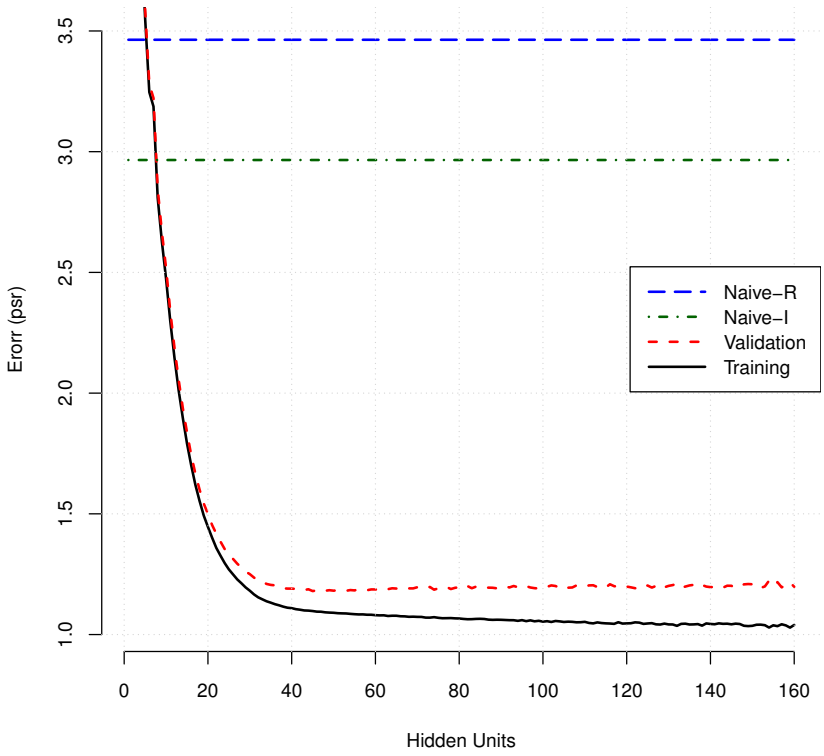


Figure 4.3: Forecasting error as the number of ERNN hidden units is varied.

In Figure 4.3 we see the training and validation forecasting errors as well as the naive errors as the number of hidden units in our ERNN is varied. Notice that the validation error is lowest at 45 hidden units with an error of 1.18psr and increases only very slightly as the number of hidden units increases to 160, where we have an error of 1.19psr. Interestingly, this small change in validation error suggests that our ERNN are only slightly over-fitting the signal. We conjecture that this minimal over-fitting may be caused

by two things. First, it may be the case that our ERNN are learning to predict events that occur only in the training data but occur following a precursor event. If such events as well as their precursors are not present in the validation data, then we would expect that our training error may continue to fall without a detrimental effect on our validation error. Second, it may be that our forecasting errors are dominated by patterns that can be learned using a relatively small number of hidden units. Thus, most of the over-fitting that occurs may cause only very small changes in the observed validation error. It should be carefully noted, however, that our training and validation forecasting errors do begin to noticeably separate with around 15-20 hidden units. This implies that our ERNN are indeed learning patterns that appear in the training partitions that do not generalize to the validation partitions with relatively few hidden units.

Overall, it seems quite clear that our ERNN are able to model EEG well, achieving a training error as low as 1.03psr and a validation error as low as 1.18psr while the Naive-R and Naive-I errors are 3.46psr and 2.97psr respectively. Indeed, the validation error drops below the Naive-R error with a mere 6 hidden units and below the Naive-I error with only 8 hidden units.

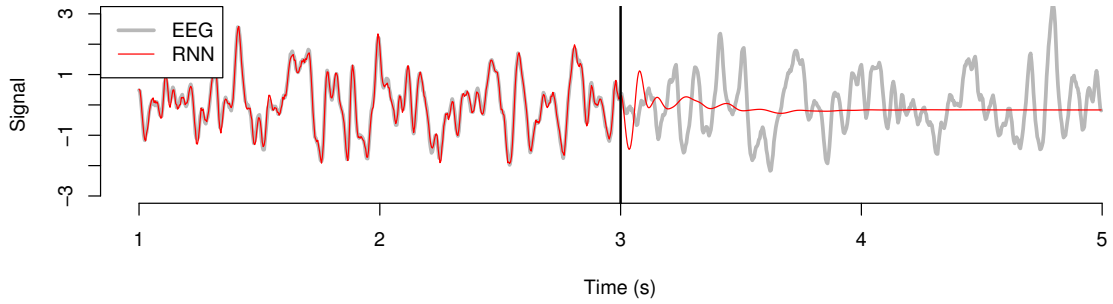
4.1.2 Iterated Models

Now that we have determined suitable hyper-parameters for training ERNN to forecast EEG, we examine an interesting experiment that may provide additional insight into the temporal information that ERNN are able to capture. In this experiment, we first train an ERNN to forecast an EEG signal a single step ahead in time as described by equation (3.3). Once training of our ERNN has completed, we place a feedback loop from the output layer of our network to the input layer so that

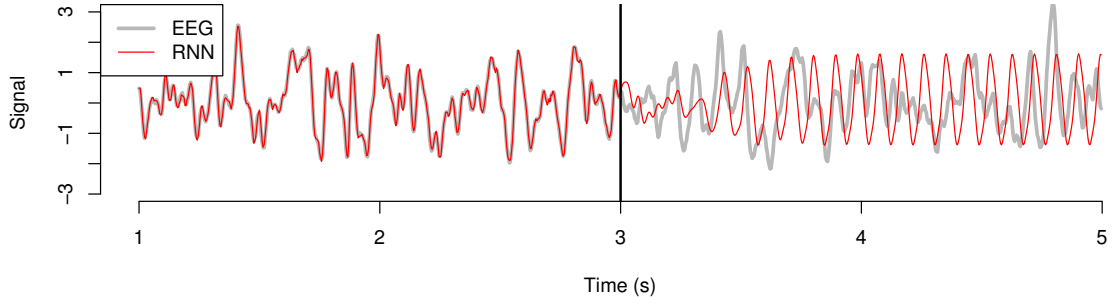
$$\mathbf{y}(t+1) = \text{ernn}(\mathbf{y}(t)) \tag{4.1}$$

when $t > T$ where T is the total number of timesteps in our training sequence. In this way our ERNN becomes an autonomous and self-driven system, also known as an iterated model.

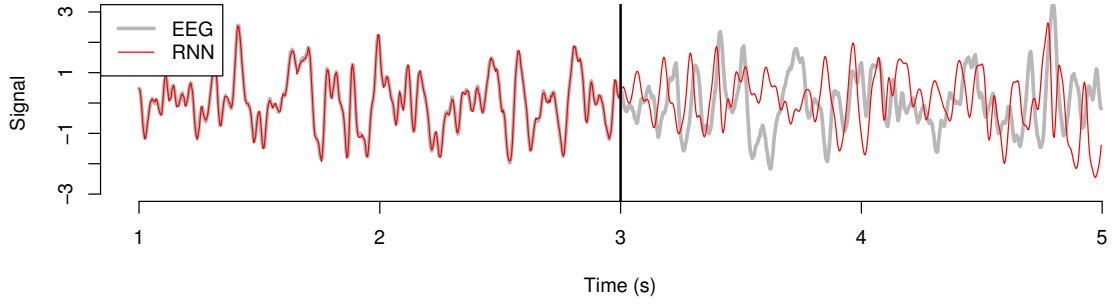
In order to evaluate the performance of our ERNN as they transition from forecasters to iterated models, we first train an ERNN to forecast all 18 of the first three seconds of a six second EEG recording from Subject A. Once training of the ERNN has completed, we allow it to run over the entire three-second training sequence and then transition its behavior to that described by equation (4.1). We then superimpose



(a) Iterated model with 20 hidden units



(b) Iterated model with 40 hidden units



(c) Iterated model with 160 hidden units

Figure 4.4: To the left of the 3 second mark we see an ERNN forecasting EEG a single step ahead. To the right of the 3 second mark we see the ERNN operating in an iterated fashion, driven only by its previous predictions. Both sides are superimposed over the true signal. 4.4a) with 20 hidden units the iterated model quickly dampens to zero. 4.4b) with 40 hidden units the iterated model falls into a periodic state. 4.4c) with 160 hidden units the iterated model has very rich and long-lasting dynamics.

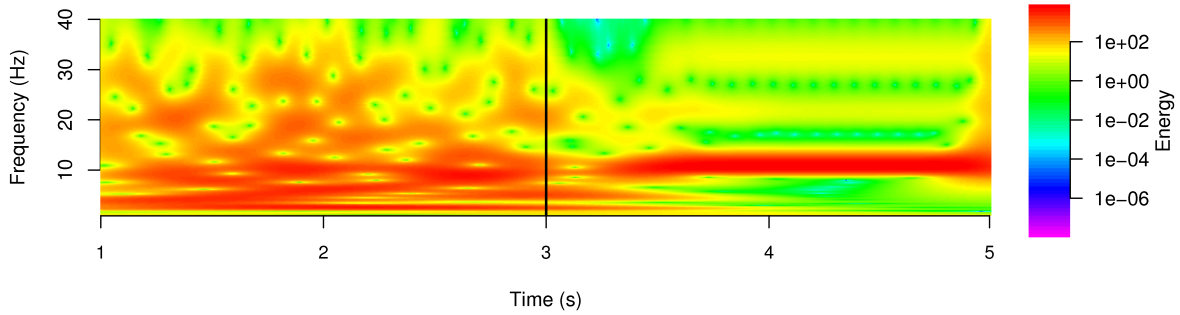
the outputs from our iterated model over the remaining three seconds of the EEG in order to compare the behavior of our iterated model with the true signal.

In Figure 4.4 we see the result of this experiment over channel P4 when run with 20, 40 and 160 hidden units respectively. To the left of the vertical line at the three second mark, we see an ERNN forecasting the data over which it was trained. Notice that the outputs of our ERNN and the true EEG signal are typically very close during this stage, indicating that our ERNN are able to closely track the true signal. At the three second mark we see how our ERNN transitions to an iterated model. Notice that the outputs of our iterated ERNN quickly diverge from the true signal. This suggests that our ERNN are not able to fully capture the full underlying dynamics of the EEG. This is not a surprising result given the fact that the underlying source of the signal is the human brain.

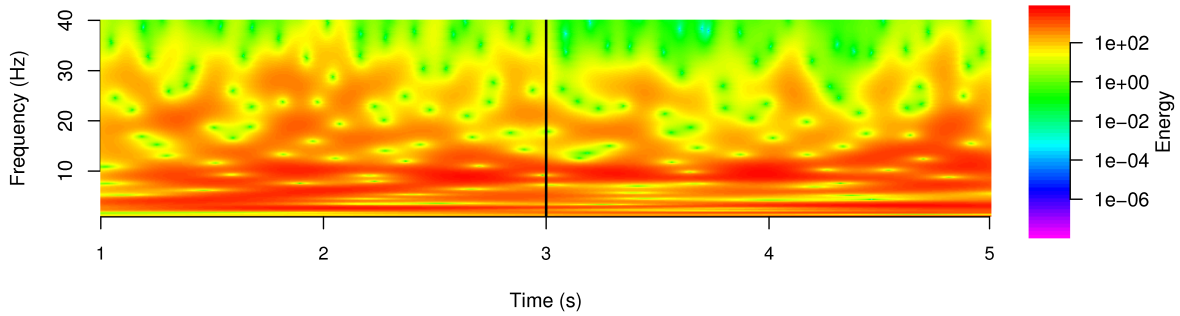
The behavior of our ERNN shortly after transitioning to iterated models, on the other hand, depends greatly on the number of hidden units. Notice in Figure 4.4a that our iterated model with only 20 hidden units quickly dampens to zero, where it remains. In Figure 4.4b, however, we see that our iterated model with 40 hidden units falls into a clearly periodic state where it appears to remain indefinitely. Next, in Figure 4.4c we see that our iterated model with 160 hidden units produces rich and long-lasting dynamics.

Although the behavior of our iterated models is occasionally slightly different across trials, channels, EEG sequences or subjects, the overall trend is quite clear and reproducible. Iterated models with very few hidden units quickly fall to zero. Iterated models with slightly more hidden units, on the other hand, tend to fall into periodic states with a relatively low frequency. As the number of hidden units continues to increase, the iterated models achieve higher frequency periods and more complex dynamics. As the number of hidden units grows past about 150, these dynamics begin to closely resemble EEG signals.

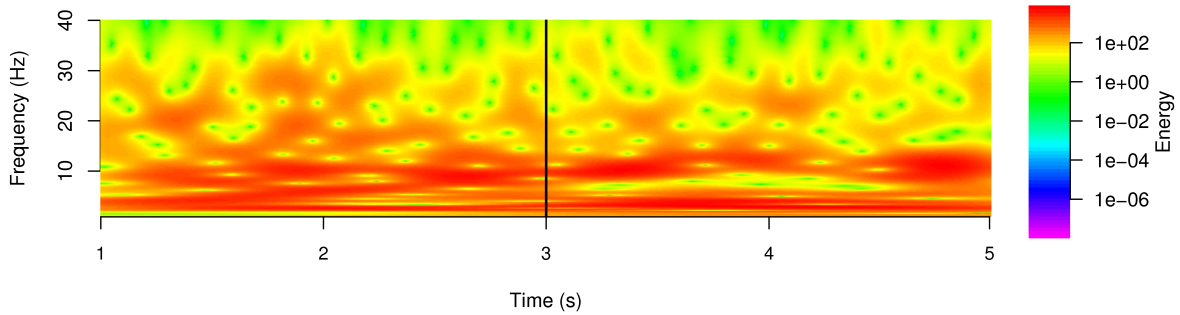
In order to support our proposition that iterated models with many hidden units can produce dynamics that closely resemble EEG, we also examine spectrograms of our iterated models. These spectrograms are generated using a Continuous Wavelet Transform and estimate the log of the energy that the EEG signal contains in the time-frequency domain. In Figure 4.5 we see the results of this analysis. First, in Figure 4.5a we see the spectrogram of an ERNN with 40 hidden units as it transitions from forecasting to an iterated model. Note that we omit the ERNN with only 20 hidden units since the spectrogram would simply show zero energy as the iterated model dies out. From this spectrogram it is clear that the periodic signal generated



(a) Spectrogram of iterated model with 40 hidden units.



(b) Spectrogram of iterated model with 160 hidden units.



(c) Spectrogram of true EEG signal.

Figure 4.5: To the left of the 3 second mark we see the spectrogram of an ERNN forecasting P4 a single step ahead. To the right of the 3 second mark we see the spectrogram of the ERNN operating as an iterated model. Bright regions show areas of high energy. 4.5a) with 40 hidden units our iterated model produces energy primarily at roughly 12Hz. 4.5b) with 160 hidden units our iterated model produces transient energy roughly between 0 and 35Hz. 4.5c) the true signal shows transient energy between roughly 0 and 40Hz.

by our iterated ERNN with 40 hidden units has a period of roughly 12Hz. Next, in Figure 4.5b we see the spectrogram of an ERNN with 160 hidden units as it transitions from forecasting to an iterated model. Finally, in Figure 4.5c we see the spectrogram of the entire true EEG signal. Notice that the spectrogram of our ERNN with 160 hidden units contains transient energy in roughly the same frequency ranges as the true signal with the exception that it often contains slightly less energy in frequencies higher than 35Hz.

It is difficult to draw firm conclusions from our experiments with iterated models for several important reasons. First, the length of time during which our iterated models exhibit complex behavior does not necessarily reflect the complexity of our forecaster. Indeed, even relatively simply iterated functions can exhibit complex, long-term dynamics. Second, the fact that our iterated models rapidly diverge from the true signal makes it difficult to determine if their behavior is meaningful. Nevertheless, these experiments are fascinating and, if nothing else, they suggest that further investigation is required into the relationship between the number of hidden units in an ERNN and amount of temporal information that they are able to capture.

4.2 Classification

In this section we examine the performance of the classification algorithms described in Section 3.4 as they are applied to each of the datasets described in Section 3.1. We begin by exploring the regularization requirements of our classification problem by examining how our training and validation performance changes as we vary the number of hidden units in our ERNN. We then present our final classification accuracies with decisions made at one second intervals. Next, we describe an approach that is frequently used in BCI literature for expressing the performance of our classifiers in terms of information transfer rates and provide our results in bits per minute (bpm). Next, we examine the effects of changing our decision rate on classification accuracy and information transfer rate. Finally, we provide some analysis of the decision boundaries learned by our error classifiers.

4.2.1 Regularization

Most of the parameters used to train our ERNN are carried over from Section 4.1.1, namely the number of training epochs, steps unrolled and the length of our initial transient period. The number of hidden units to

use during classification, however, remains to be determined because a different level of regularization may be required. In order to examine the regularization requirement for our classification methods, we examine the training and validation errors as the number of hidden units in our ERNN are varied. This experiment was repeated for each subject and classifier. Since the results of these experiments are roughly the same, we only show the outcome for the two-task problem for Subject A while using the WTA classifier.

In Figure 4.6 we see the outcome of this experiment when carried out using WTA to assign class labels. Notice that the average classification accuracy for our training partitions approaches 100% with around 40 hidden units and that our average validation accuracy is highest, roughly 85%, with between 10-20 hidden units. These results may seem somewhat surprising since our evidence from Section 4.1.1 suggests that our forecasting errors continue to decrease until 45 hidden units and our evidence from Section 4.1.2 suggesting that ERNN with many hidden units create the richest iterated dynamics. We should keep in mind, however, that our classification problem has a different goal than our forecasting problem and therefore may require a different level of regularization.

Nevertheless, comparing our forecasting performance and our classification performance as the number of hidden units is varied leads us to several interesting observations. Recall from Figure 4.3 that our training and validation forecasting errors begin to separate at roughly 10-20 hidden units, which is also where our classification accuracy peaks in Figure 4.6. This suggests that our ERNN begin to learn some patterns that do not generalize well starting with around 10-20 hidden units. One possibility for explaining the fact that our forecasting errors continue to drop after 10-20 hidden units may be that our ERNN continue to learn some patterns or noise that are present in EEG signals produced during both imagined mental tasks. In other words, despite the fact that our ERNN begin to learn patterns that do not generalize well, they may simultaneously be learning patterns that are common among both of the EEG signals, causing the forecasting error to continue to fall. Another interesting possibility is that the more complex and possibly longer-term patterns learned by ERNN with many hidden units simply may not help discriminate between classes. Presumably, the ability of our ERNN to capture complex and long-term temporal patterns increases as the number of hidden units available increases. As this happens, our training procedure may reduce our forecasting error by favoring these complex, long-term patterns over simpler, shorter-term patterns. While

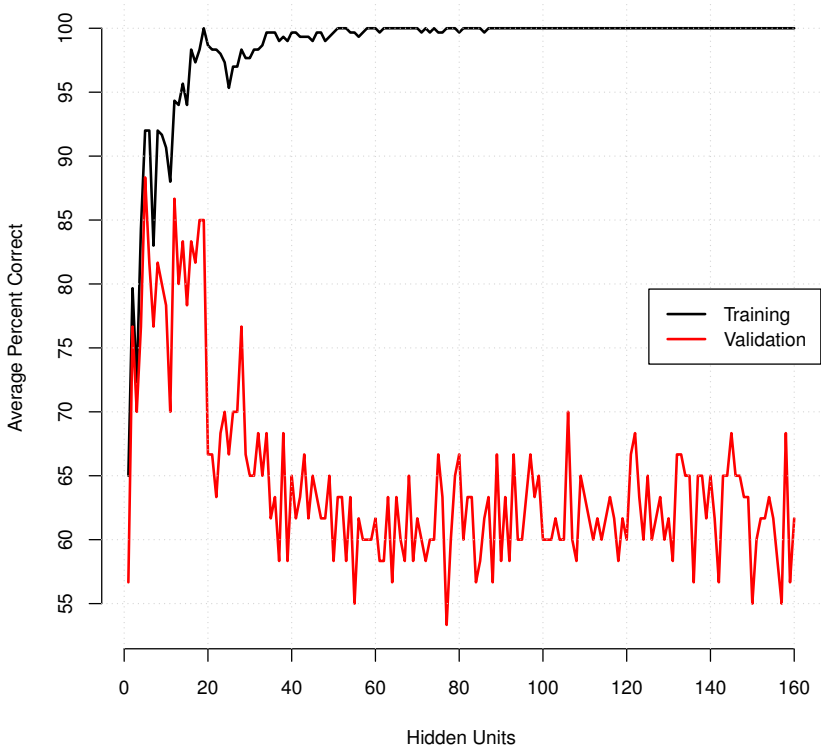


Figure 4.6: Classification accuracy as the number of hidden units is varied using WTA.

this may continue to reduce our forecasting error, it is possible that complex, long-term patterns do not discriminate between classes well.

From a frequency perspective, we may draw somewhat different conclusions. Recall from Section 4.1.2 that our iterated models with fewer hidden units tend to fall into periodic states with relatively low frequencies while iterated models with many hidden units tend to contain more high frequency and transient components. This may suggest that ERNN with fewer hidden units place more of an emphasis on predicting low or medium frequency components of the EEG signals. This notion may be supported by the fact that most of the energy in an EEG signal lies in these frequencies. Since ERNN with relatively few hidden units have limited resources, it makes sense that they first minimize the error associated with the low to medium frequency components. Thus, the fact that ERNN with fewer hidden units produce better classification re-

sults may suggest that relatively low to medium range frequency components contain information that is most useful for discriminating between imagined mental tasks.

4.2.2 Accuracy

Now that we have examined the effects of varying our regularization parameter on classification performance, we apply each of our classifiers to the data collected from each of our five subjects using the six-fold cross-validation procedure described in Section 3.1. The number of hidden units used for testing our classifiers is selected by finding the peak in our validation performance. Our test accuracy is then evaluated by averaging the performance of each of our six classifiers over the test partition, which consists of the final 40% of the data. Testing of our classifiers is done in this manner in order to simulate the real-world use of a BCI system. That is, once a classifier is trained, the user will issue commands to the BCI system by voluntarily changing the mental task they are performing. Although, here, we only consider classification accuracies when decisions are made at one-second intervals, we examine the effect of changing the rate at which our decisions are made in Section 4.2.4.

In Table 4.1 we see our classification accuracies for our two-task problem, consisting of the imagined right hand movement and count backward from 100 by 3's tasks, for all five subjects and for all three methods. Additionally, we see the mean performance of each classifier when averaged across all five subjects. While examining these classification accuracies, we keep in mind that we would expect a random classifier to achieve 50% correct.

A number of interesting observations can be made from Table 4.1. First, notice that our classification accuracies are quite high for the training partitions. This suggests that our classification methods are at least capable of modeling the data from each class. Not surprisingly, however, our classification accuracies decrease as the classifiers generalize to the validation data and degrade further upon application to the test data. These drops in generalization performance suggest that some of the discriminating patterns found in the training data do not appear in the validation or test data. This may be caused by noise in the signal or by transient patterns or by a combination of both.

We also observe that the test classification accuracies vary considerably between the subjects. Specifically, the test accuracies for Subject A and Subject B are quite high, in the 80-90% range, while those for

Table 4.1: Percent Correct Classification for Two Tasks.

Subject	Method	Hidden Units	Training	Validation	Test
A		5	92.0%	88.3%	89.2%
B		8	97.7%	83.3%	67.5%
C	WTA	14	98.3%	93.3%	93.3%
D		17	95.3%	65.0%	52.1%
E		21	97.0%	73.3%	62.9%
Mean		13.0	96.1%	80.6%	73.0%
A		6	87.7%	83.3%	77.5%
B		2	86.3%	83.3%	81.7%
C	LDA	20	98.0%	90.0%	85.4%
D		24	97.3%	63.3%	52.9%
E		16	96.3%	71.7%	63.8%
Mean		13.6	93.1%	78.3%	72.3%
A		11	86.0%	80.0%	80.0%
B		11	95.0%	83.3%	80.4%
C	QDA	15	95.0%	88.3%	87.1%
D		1	64.3%	61.7%	51.3%
E		11	92.3%	73.3%	64.6%
Mean		9.8	86.5%	77.3%	72.7%

Subject B are modest, in the 60-80% range, and those for Subject D and E are relatively low, in the 50-60% range. There are a number of potential causes for this variability across subjects. It may be, for example, that some subjects are more engaged in the recording session or are simply paying more attention. If some of the subjects are distracted or uninterested, it may result in poor classification performance. The individual differences among subjects and the environments in which EEG recording took place should also be considered. Recall from Section 3.1 that Subjects A and B are able-bodied individuals and that their recording sessions took place in a controlled laboratory setting. Subjects C and E, on the other hand, have high-level spinal cord injuries and Subject D has severe multiple sclerosis. Additionally, recording from Subjects C, D and E took place in their homes. Although Subject C actually outperformed all of the other subjects, the recording environments and disabilities of Subject D and Subject E may have contributed to their poor performance.

Interestingly, we also notice that our WTA classification approach delivers roughly the same performance as our LDA and QDA approaches. Looking at the mean performance across subjects for each approach, it appears that WTA may slightly outperform LDA and QDA, although it is difficult to draw any firm conclusions given the small number of subjects. We do notice, however, that Subject B appears to be the only subject that achieves a large performance improvement through the use of LDA or QDA. While the data from Subject B may benefit from the additional flexibility introduced by LDA and QDA, it appears that, in general, WTA is typically enough to discriminate between the forecasting errors produced by our ERNN.

Table 4.2: Percent Correct Classification for Four Tasks.

Subject	Method	Hidden Units	Training	Validation	Test
A		11	89.2%	68.3%	53.5%
B		5	84.5%	62.5%	50.0%
C	WTA	14	90.7%	60.8%	67.5%
D		28	93.8%	42.5%	27.9%
E		12	85.0%	56.7%	36.9%
Mean		14.0	88.6%	58.2%	47.2%
A		12	85.8%	63.3%	41.6%
B		7	81.8%	55.8%	45.6%
C	LDA	21	91.8%	60.0%	56.0%
D		35	88.0%	38.3%	28.8%
E		21	87.3%	49.2%	28.3%
Mean		19.2	86.9%	53.3%	40.1%
A		13	76.7%	59.2%	40.7%
B		5	79.3%	55.8%	46.5%
C	QDA	16	74.3%	52.5%	48.8%
D		23	58.5%	36.7%	27.3%
E		14	72.2%	47.5%	39.0%
Mean		14.2	72.2%	50.3%	40.5%

Now that we have investigated the classification accuracies for our two-task problem, let's examine how they scale up to the full four-task problem. The outcome of these experiments is shown in Table 4.2. Note that with a four-task problem, we expect that a random classifier would achieve a classification accuracy of only 25%. Notice that Subject C again achieves the highest classification accuracy, as high as 67.5%, while Subject A and B achieve moderate performance and Subject D and E again achieve relatively poor

classification accuracy. Since Subject D and E achieve poor performance for both the two-task and four-task problems, we conjecture that it is not simply a poor selection of mental tasks that leads to these low classification accuracies. Overall, the results for our four-task problem seem to align with those with the results found in our two-task problem with two exceptions. First, our classification accuracies are considerably lower. Although this is not surprising due to the increased number of classes, it is also somewhat disappointing. Examining classification accuracy alone, however, provides little insight into any improvement or reduction in overall performance that is seen by increasing the number of mental tasks used. In Section 4.2.3, we examine an approach that makes it easier for us to compare our the performance of our two-task problem and our four-task problem.

Table 4.3: Confusion matrix for four tasks in the test partition for Subject C.

		Predicted			
		Count	Fist	Cube	Song
Actual	Count	80%	5%	10%	5%
	Fist	0%	65%	35%	0%
	Cube	20%	0%	55%	25%
	Song	25%	5%	15%	55%

Table 4.4: Confusion matrix for four tasks in the test partition for Subject E.

		Predicted			
		Count	Fist	Cube	Song
Actual	Count	40%	25%	30%	5%
	Fist	10%	65%	20%	5%
	Cube	20%	20%	30%	30%
	Song	20%	15%	50%	15%

Although we have seen how our classification accuracies vary across subjects, we have yet to examine how each imagined mental task performs for a given subject. In Table 4.3 we see the confusion matrix for Subject C and for all four imagined mental tasks when using WTA. Since the classification accuracies

for Subject C were quite high, it is not surprising that most of the values along the diagonal are also quite high. Specifically, notice that the classification accuracies for the Count and Fist tasks are particularly high, suggesting that our two-task problem was beneficial for Subject C. On the other hand, the Cube and Song tasks are correctly classified far less often. Interestingly, notice that the Cube task is often confused for the Song and Count tasks and vice versa. This suggests that our classification algorithms found the Cube and Song tasks to be quite similar. Eliminating one of these tasks or replacing it with a different task might increase classification for Subject C.

In Table 4.4 we see a confusion matrix for Subject E, again for the four-task problem and using WTA. Recall that we achieve far lower classification accuracies for Subject E. Again, notice that the Count and Fist tasks performed best, suggesting that our two-task problem was also favorable for Subject E. Notice that we see above random classification accuracies for all values along the diagonal with the exception of the Song task, which is classified as the Cube task 50% of the time. Interestingly, however, the Count and Fist tasks are only rarely confused with the Song task. This leads us to a similar conclusion to the one that we reached for Subject C. Namely, the Song task should be eliminated or replaced with another imagined mental task.

The differences in performance that we see for various mental tasks in different subjects suggests that a practical BCI system should consider the performance of various mental tasks in a subject-specific manner. Each subject may find some mental tasks to be unpleasant or difficult to perform. Furthermore, variations in brain organization and development among subjects may cause differences in the patterns produced by various mental tasks.

4.2.3 Information Transfer Rate

Classification accuracies can be difficult to compare because they do not take the number of classes or the rate at which decisions are made into account. In order to provide a more universal indicator of our classification performance, we also present our classification results in the form of information transfer rates. In the BCI literature, information transfer rates are commonly calculated in bits per minute (bpm) according to

$$\textit{bits per minute} = V\left(\log_2 K + P\log_2 P + (1-P)\log_2\frac{1-P}{K-1}\right), \tag{4.2}$$

where $V = 60$ is the classification rate in decisions per minute, K is the number of classes and P is the classification accuracy as the fraction of correct decisions over total decisions. Equation (4.2) is derived from an equation used in information theory used to describe the number of bits per unit of time that can be transmitted across a noisy channel [43, 44].

In Table 4.5, we see our classification results in terms of information transfer rates for our two-task problem. When viewed in this way, we see that Subject A and Subject C may be expected to be able to communicate as many as 30.4bpm and 38.7bpm respectively when WTA is used to assign class labels. Although Subject B is only expected to communicate 5.4bpm with WTA, up to 18.8bpm is achieved using LDA. For Subject D we see that very little if any communication is expected using any our classifiers. For Subject E, we expect only around 3bpm. Although our information transfer rates vary considerably between subjects, we do achieve an encouraging average 15.5bpm across subjects when using WTA. When using LDA and QDA, our mean bitrates drop by about 2-3bpm.

In Table 4.6, we see our classification results in terms of information transfer rates for our four-task problem. We can now see that our classification algorithms do not scale as well as we might hope as the number of imagined mental tasks is increased. When using WTA, our mean information transfer rate drops from 15.5bpm to 13.2bpm, with LDA our mean information transfer rate drop from 12.0bpm to a mere 6.7bpm and with QDA our mean information transfer rate drops from 12.9bpm to 6.0bpm. Additionally, the information transfer rate drops for each subject and for each algorithm as the number of mental tasks is increased to four. Without online tests, however, it is difficult to know the impact that this drop might have on a BCI user. Although the information transfer rates are lower, the increased degrees of freedom may provide a better user experience and feedback.

4.2.4 Decision Rate

Although we have investigated how our classification accuracies and information transfer rates are affected by scaling from a two-task to a four-task problem, we have yet to examine how the rate at which our class labels are assigned effects performance. Recall from Sections 3.4.1, 3.4.3 and 3.4.2 that each of our classification methods involves processing our forecasting errors over an interval consisting of ω timesteps. In all of the experiments performed up until now, we have used one-second intervals, $\omega = 256$.

Table 4.5: Classification Bitrates for Two Tasks.

Subject	Method	Hidden Units	Training	Validation	Test
A		5	35.9bpm	28.8bpm	30.4bpm
B		8	50.5bpm	21.0bpm	5.4bpm
C	WTA	14	52.5bpm	38.7bpm	38.7bpm
D		17	43.6bpm	4.0bpm	0.1bpm
E		21	48.3bpm	9.8bpm	2.9bpm
Mean		13.0	46.2bpm	20.5bpm	15.5bpm
A		6	27.7bpm	21.0bpm	13.8bpm
B		2	25.4bpm	21.0bpm	18.8bpm
C	LDA	20	51.5bpm	31.9bpm	24.0bpm
D		24	49.3bpm	3.1bpm	0.1bpm
E		16	46.3bpm	8.4bpm	3.3bpm
Mean		13.6	40.0bpm	17.1bpm	12.0bpm
A		11	24.9bpm	16.7bpm	16.7bpm
B		11	42.8bpm	21.0bpm	17.2bpm
C	QDA	15	42.8bpm	28.8bpm	26.8bpm
D		1	3.6bpm	2.4bpm	0.0bpm
E		11	36.5bpm	9.8bpm	3.7bpm
Mean		9.8	30.1bpm	15.7bpm	12.9bpm

Table 4.6: Classification Bitrates for Four Tasks.

Subject	Method	Hidden Units	Training	Validation	Test
A		11	80.1bpm	35.8bpm	16.0bpm
B		5	67.9bpm	27.1bpm	12.5bpm
C	WTA	14	84.3bpm	24.8bpm	34.5bpm
D		28	94.0bpm	6.3bpm	0.2bpm
E		12	69.1bpm	19.6bpm	3.0bpm
Mean		14.0	79.1bpm	22.7bpm	13.2bpm
A		12	71.1bpm	28.2bpm	5.7bpm
B		7	61.6bpm	18.6bpm	8.6bpm
C	LDA	21	87.7bpm	23.7bpm	18.8bpm
D		35	76.8bpm	3.7bpm	0.3bpm
E		21	75.0bpm	11.7bpm	0.2bpm
Mean		19.2	74.4bpm	17.2bpm	6.7bpm
A		13	50.8bpm	22.7bpm	5.1bpm
B		5	56.2bpm	18.6bpm	9.3bpm
C	QDA	16	46.2bpm	14.9bpm	11.3bpm
D		23	21.8bpm	2.9bpm	0.1bpm
E		14	42.4bpm	10.2bpm	4.1bpm
Mean		14.2	43.5bpm	13.9bpm	6.0bpm

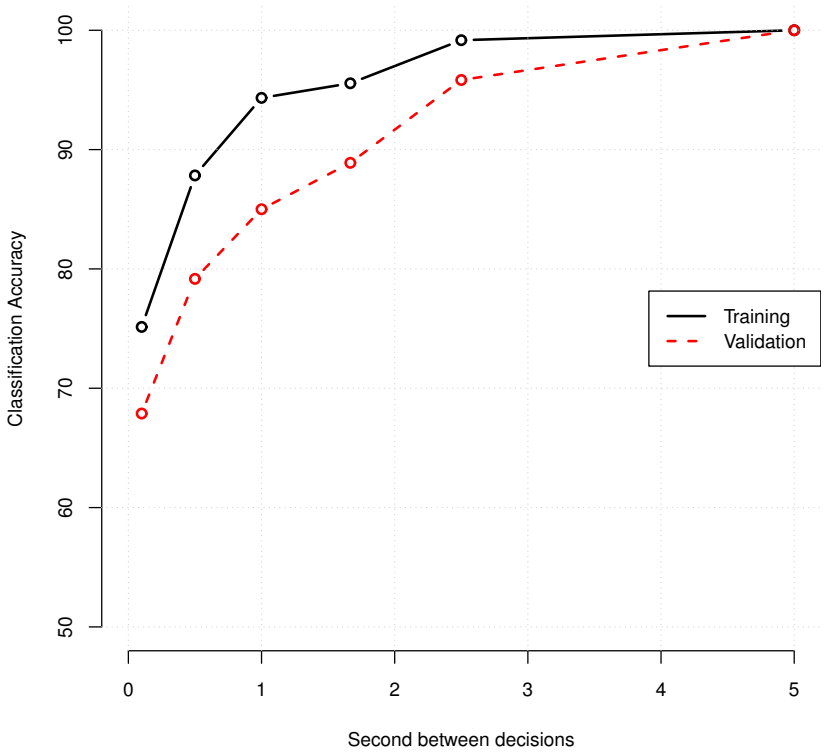


Figure 4.7: Classification accuracy as the number of seconds between decisions is varied.

In Figure 4.7, we see how our test classification accuracy changes for Subject A in our two-task problem as the seconds between decisions is varied. Interestingly, we are able to achieve 100% correct classification when class labels are assigned every five seconds. It is not ideal, however, to increase our decision interval until 100% correct classification is achieved. In Figure 4.8 we see how our information transfer rate changes as the decision interval is increased. Clearly, the highest information transfer rates are achieved when class labels are assigned very quickly. Most BCI users, however, will not issue commands at a rate faster than about once per second until they have become extremely skilled at using the system. It is for this reason that we use a decision interval of one second in all of the other experiments performed here.

It seems that there is a balance to be found between classification accuracy and information transfer rate. If a BCI system makes decisions very slowly, they will often be correct but it will take a long time to convey very much information. If decisions are made too quickly, a BCI system may operate faster than a user

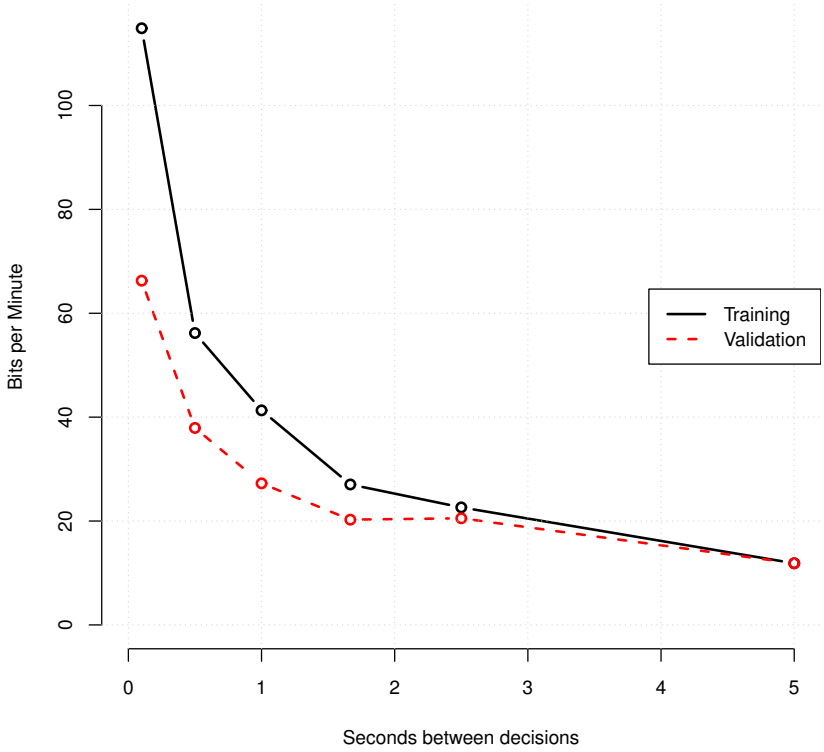


Figure 4.8: Information transfer rate as the number of seconds between decisions is varied.

can control it. Additionally, a rapid decision rate leads to many incorrect commands being issued. Online experiments are required in order to determine where exactly this balance lies and how user experience is affected by decision rate. In fact, individual users may prefer flexible decision rates, especially as their skill at operating a BCI system evolves.

4.2.5 Error Boundaries

In order to gain further insight into the differences between WTA, LDA and QDA, it is important that we explore the decision boundaries that they are generating. In Figure 4.9, we see the forecasting errors produced by each of the ERNN in our two-task problem plotted against each other when applied to the data from both tasks in the test partition for Subject A. Note that for errors to the left of the diagonal, the ERNN trained to forecast EEG from our counting task was better able to forecast the signal. On the other hand,

errors to the right of the diagonal were better forecast by the ERNN trained over EEG from our imagined right hand movement task. Although it appears that each ERNN is generally better at forecasting the EEG over which it was trained, it is clear that our errors at each timestep are extremely variable and largely overlap.

In Figure 4.10, we see the effect of averaging our errors over one-second intervals, as is done by WTA. Note that when classifying these errors with WTA, the diagonal defines the class boundary. Clearly, averaging the errors over a number of timesteps dramatically reduces the noise and overlap seen in the individual errors. In Figure 4.11 we see contours depicting the Gaussians used by LDA along with their linear intersection superimposed on top of our forecasting errors from each individual timestep. Interestingly, these contours intersect only slightly below and to the right of the diagonal. Although our class labels are assigned by multiplying the probabilities determined by these Gaussians over an interval, this plot suggests that LDA assigns these probabilities largely based on their relative location to the diagonal. In Figure 4.12 we see contours depicting the Gaussians used by QDA along with their quadratic intersection superimposed on top of our forecasting errors from each individual timestep. Although our Gaussians now take on different shapes, we see that our intersection of equal probability is again near the diagonal and then cuts through the region where the ERNN trained over the count task clearly produces lower forecasting error. This evidence leads us to believe that LDA and QDA are assigning class labels in a way that is extremely similar WTA.

It is also important to note that LDA and QDA rely on a number of assumptions. First, both LDA and QDA assume sample means and covariances. Since the exact values of these parameters can not determined, it is possible that our Gaussians are not ideally placed. Second, LDA assumes that both classes have identical covariances. Figure 4.9 illustrates that the forecasting errors for each class are distributed somewhat differently. Additionally, both approaches assume that our classes can be effectively modeled by Gaussian distributions. Again, Figure 4.9 suggests that our forecasting errors are not normally distributed since the vast majority of samples are located in the lower left hand corner with fewer samples fanning out along the diagonal. Finally, our method of combining errors over an interval assumes that the errors are independent. Given the periodic and autocorrelated nature of EEG, this assumption certainly does not hold. We conjecture that the violation of these assumptions is the reason that LDA and QDA typically perform less favorably than the simpler WTA.

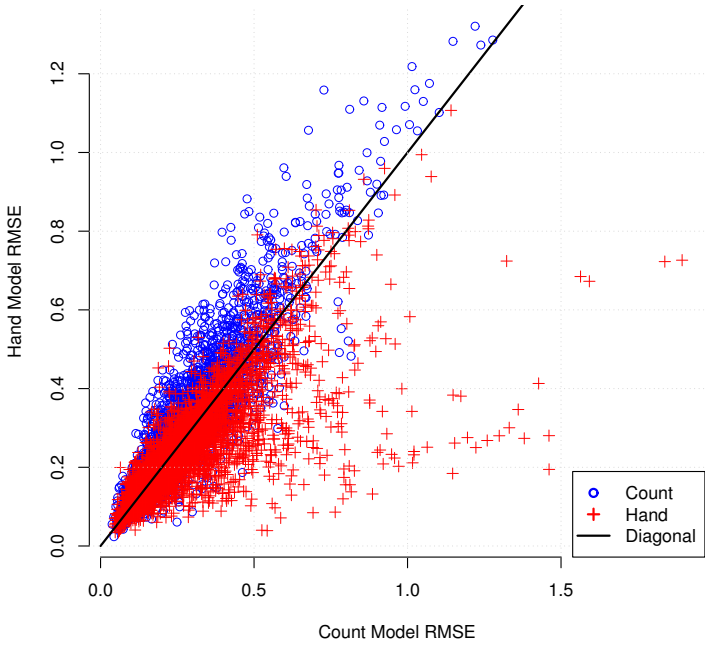


Figure 4.9: Forecasting errors.

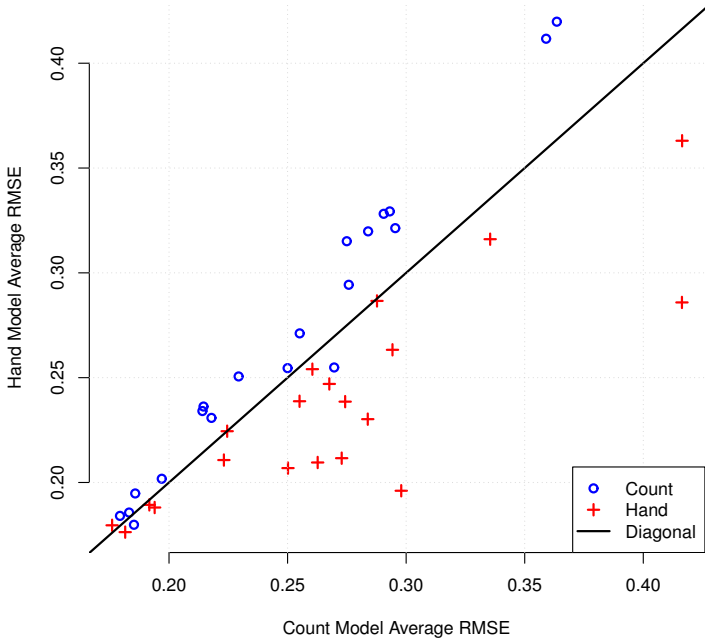


Figure 4.10: Averaged forecasting errors. Diagonal is WTA class boundary.

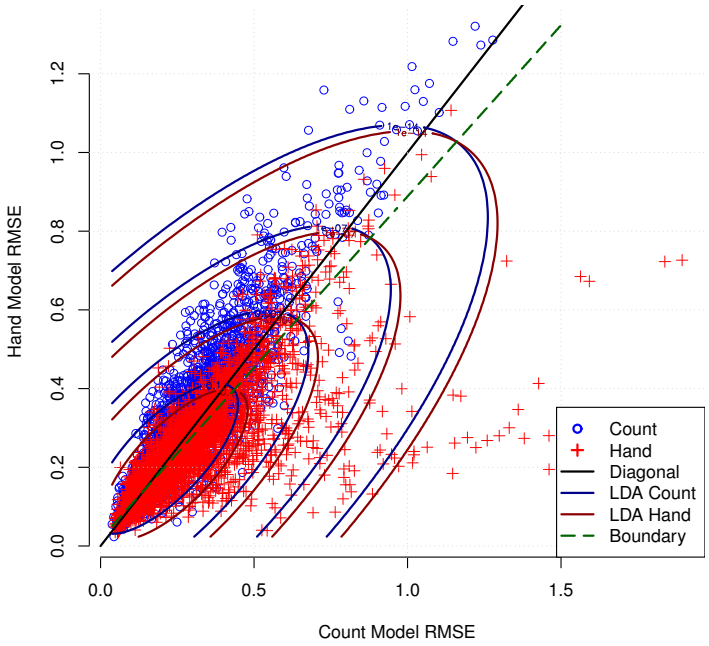


Figure 4.11: Forecasting errors with LDA probability contours.

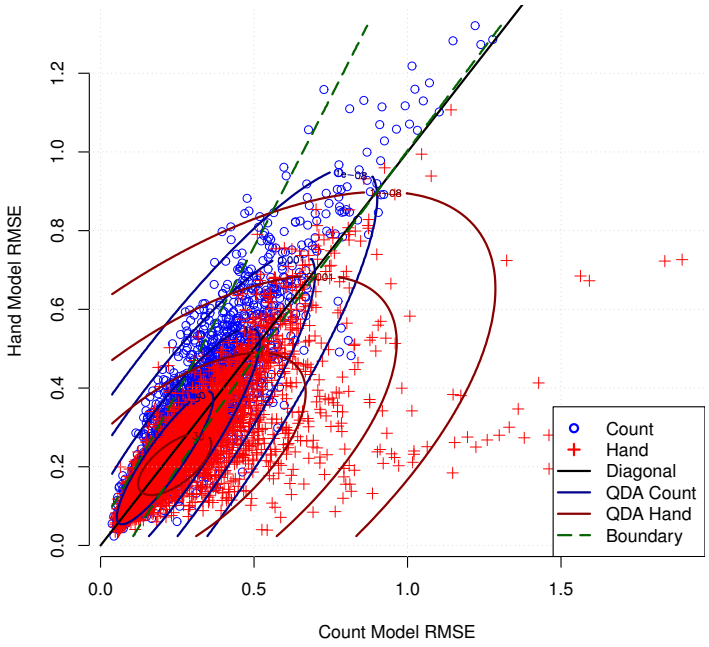


Figure 4.12: Forecasting errors with QDA probability contours.

Chapter 5

Conclusions

In this chapter we offer some concluding remarks about the work performed in this thesis. We begin by briefly summarizing the methods and results that were reported here. Next, we offer a comparison of our results with the current state-of-the-art in BCI. Finally, we discuss a number of questions that remain to be answered in future works.

5.1 Summary

In this thesis we developed a new algorithm for EEG classification that can be used to build Brain-Computer Interfaces. To this end, we outlined a technique where ERNN are used to model EEG by forecasting the signal a single step ahead in time. These ERNN are trained using Backpropagation Through Time in combination with Scaled Conjugate Gradients and careful initial weight selection. A classifier can then be constructed by training a separate network to forecast sample EEG from each of the mental states of interest. Each of these forecasters can be thought of as an expert at forecasting the type of EEG over which it was trained. Class labels are then assigned to novel data by applying each network and using one of three classifiers on the resulting errors. The first of these classifiers, Winner-Takes-All, simply averages the forecasting errors over a brief window and assigns the class label associated with the ERNN that was able to forecast the EEG with the lowest error. The remaining two classifiers, Linear Discriminant Analysis and Quadratic Discriminant Analysis, are generative, statistical classifiers that model our forecasting errors with

Gaussians and assign class labels by finding the joint probability that a short window of errors belongs to a given class.

Each of the hyper-parameters involved in these approaches was explored and it was determined that with 20 steps unrolled, 250 training epochs and 40 hidden units, an ERNN is able to forecast an EEG signal with a validation error as low as 1.18 percent of the signal range. We also found that when a feedback loop is placed between the outputs and inputs of an ERNN trained to forecast EEG that interesting dynamics can be observed. Notably, ERNN with this configuration and above 160 hidden units generate signals with a frequency spectrum similar to that of the true EEG signal.

We then applied each of our classification algorithms to five EEG datasets, two of which were collected from able-bodied individuals in a laboratory setting and three of which were recorded from subjects with disabilities in their home settings. A relatively inexpensive EEG amplifier was used for data acquisition. These classification experiments demonstrated that two subjects were able to achieve information transfer rates in the 30-38bpm range, one subject was able to achieve information transfer rates as high as 18bpm and the remaining two subjects were only able to achieve 0-3bpm. The mean information transfer rate across all five subjects was 15.5bpm. Additionally, our examination of WTA, LDA and QDA has reveled that WTA typically outperforms LDA and QDA and that the decision boundaries found by each of these approaches are very similar. It is encouraging that the highest classification rate was achieved with Subject C. As a person with quadriplegia and in their home environment, Subject C certainly belongs to a target demographic for this technology.

5.2 State-of-the-art

In Chapter 2 we reviewed a number of state-of-the-art BCI paradigms and EEG classification algorithms. Due to the wide range of approaches that can be taken to build a BCI systems and because of the various ways in which their performance is evaluated, it is difficult to perform a direct comparison. We should recall, however, that current SSVEP speller systems can achieve information transfer rates as high as 62.5bpm and that P300 speller systems can achieve information transfer rates as high as 13.3bpm in healthy subjects. Although our classification algorithm achieved a peak information transfer rate of 38.7bpm, which

is between the rates achieved by the aforementioned approaches, it should be noted that classification of imagined mental tasks does not require the user to deliver their overt attention to an external stimulus.

Due to the extensive use of user training, biofeedback and variable decision rates, it is uncommon for information transfer rates to be reported in current literature concerning the classification of imagined mental tasks. Nevertheless, an examination of two recent works, one by Millán, et al., and one by Anderson, et al., reveals that these state-of-the-art approaches achieve anywhere from 52% correct on a three-task problem every 0.5 seconds to 78% correct on a five-task problem every 1.8 seconds before any user feedback [17, 20]. Substituting these values into our equation (4.2), we arrive at bitrates of 12.7bpm and 37.4bpm, respectively. This suggests that the experiments performed here are competitive with the current approaches that utilize PSD and TDE feature representations.

To our knowledge, the only other work where EEG is classified using the errors produced by neural networks trained to forecast EEG signals is that by Coyle, et al. [26]. Recall that this approach was used to classify a two-task imagined motor movement problem and that feedforward networks were used in combination with TDE, as opposed to recurrent networks. The information transfer rates achieved in this particular study did not exceed 9.96bpm. This suggests that ERNN may provide an advantage over feedforward architectures.

Given these comparisons, it appears that our classifiers perform well relative to the state-of-the-art for some of our subjects and relatively poorly for others. Without testing these methods on more datasets and performing interactive testing it is difficult to know if they will generally outperform other methods or not. The use of ERNN for classifying EEG does, however, show considerable potential.

5.3 Future Work

A number of questions related to the work at hand remain to be answered. One of the most obvious of these questions is whether or not other classifiers can outperform WTA, LDA and QDA when applied to our forecasting errors. Since LDA and QDA rely on a number of prior assumptions that may not hold in this setting, perhaps other classifiers would do better. Support Vector Machines and Feedforward Artificial Neural Networks are both examples of state-of-the-art classifiers that could be applied to our forecasting errors.

In addition to applying different classifiers to our forecasting errors, modeling EEG at different or multiple timescales should be investigated. For example, increasing the number of steps ahead that our networks are forecasting may yield different results. It is also possible to train a single network or even several networks to forecast the signal with different forecasting horizons. EEG signals could also be filtered and downsampled to provide different frequency ranges to each of these forecasters. Such configurations may provide better forecasting performance, since the memory requirement of the networks would be reduced. Although we believe that these approaches should be investigated, one must be careful not to introduce too many hyper-parameters into the classification pipeline since the time required to search for these values may make such a configuration impractical for use in a BCI system.

One of the major limitations of our current implementation is, in fact, computational performance. Although the forward passes of our ERNN and the assignment of class labels can easily be performed in real-time, training of a classifier with 10 hidden units takes approximately 10 minutes on a 2.83Ghz Intel Core2 Q9550. We should note, however, that our current implementation is written entirely in R, which is an interpreted programming language designed for analysis and rapid prototyping [45]. Although an optimized version written in a native language may run considerably faster, it still seems unlikely that a full cross-validation and hyper-parameter search could be run on portable hardware while a BCI users waits. Recent advances in distributed systems, however, suggest that this may not be necessary. Ericson, et al., have suggested that a portable device may be used to stream EEG to a cloud computing service, thereby reducing the need to perform demanding computation on site [46]. Since the use of ERNN in a real-world setting may be computationally demanding, further investigation into more efficient, parallel and distributed implementations should be investigated.

The use of RNN architectures other than ERNN should also be investigated. In particular, Echo State Networks (ESN) have received considerable attention in recent literature [47, 48]. Since the recurrent weights of an ESN do not adapt during training, performance can be orders of magnitude faster than ERNN, potentially overcoming the performance limitations found in our current implementation. It remains to be determined, however, whether or not ESN can offer the same forecasting and classification performance as ERNN. Future experiments should compare the performance of ESN and ERNN to determine if the use of ESN is a possible alternative.

Finally, we conclude this thesis with a note about online testing. Although the algorithms explored here have delivered exciting results and have demonstrated an ability to classify EEG in an offline setting, it is impossible to know whether or not they will prove useful in interactive BCI systems. In order to properly evaluate the performance of these methods, it is essential that they be integrated with a BCI system and interactively tested with a number of subjects. Only then will it be clear whether or not our classifiers work with an acceptable number of subjects and whether or not performance can be improved with feedback. Questions about how many and which mental tasks to use, how fast decisions should be made, what user experience is like and whether or not people with disabilities find BCI systems useful can not be fully answered without real-world application and testing.

Bibliography

[1] H. Berger and P. Gloor. *Hans Berger on the electroencephalogram of man: The fourteen original reports on the human electroencephalogram*. Elsevier Pub. Co., 1969.

[2] P.L. Nunez and R. Srinivasan. *Electric fields of the brain: the neurophysics of EEG*. Oxford University Press, USA, 2006.

[3] J.D. Bauby. *The diving bell and the butterfly: A memoir of life in death*. Vintage, 1998.

[4] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[5] Simon Haykin. *Neural networks and learning machines*. Prentice Hall, New Jersey, USA, 2009.

[6] E.M. Forney and C.W. Anderson. Classification of eeg during imagined mental tasks by forecasting with elman recurrent neural networks. *International Joint Conference on Neural Networks (IJCNN)*, pages 2749–2755, 2011.

[7] Z.A. Keirn and J.I. Aunon. A new mode of communication between man and his surroundings. *IEEE Transactions on Biomedical Engineering*, 37(12):1209–1214, 1990.

[8] TA Skidmore and HW Hill. The evoked potential human-computer interface. *Proceedings of the Annual International Conference of the IEEE*, 13:407–408, 1991.

[9] T. Cao, X. Wang, B. Wang, C.M. Wong, F. Wan, P.U. Mak, P.I. Mak, and M.I. Vai. A high rate online ssvep based brain-computer interface speller. *5th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 465–468, 2011.

[10] L.A. Farwell and E. 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.

[11] F. Nijboer, EW Sellers, J. Mellinger, MA Jordan, T. Matuz, A. Furdea, S. Halder, U. Mochty, DJ Krusienski, and TM Vaughan. A p300-based brain-computer interface for people with amyotrophic lateral sclerosis. *Clinical neurophysiology*, 119(8):1909–1916, 2008.

[12] J.R. Wolpaw, DJ McFarland, and TM Vaughan. Brain-computer interface research at the wadsworth center. *IEEE Transactions on Rehabilitation Engineering*, 8(2):222–226, 2000.

[13] G. Pfurtscheller and C. Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, 2001.

[14] F. Galán, M. Nuttin, E. Lew, P.W. Ferrez, G. Vanacker, J. Philips, and J.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.

[15] C. Anderson, E. Forney, D. Hains, and A. Natarajan. Reliable identification of mental tasks using time-embedded EEG and sequential evidence accumulation. *Journal of Neural Engineering*, 8:25–23, 2011.

[16] J.R. Millán, P.W. Ferrez, F. Galán, E. Lew, and R. Chavarriaga. Non-invasive brain-machine interaction. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(10):1–13, 2007.

[17] J.R. Millán, J. Mourino, M. Franze, F. Cincotti, M. Varsta, J. Heikkonen, and F. Babiloni. A local neural classifier for the recognition of EEG patterns associated to mental tasks. *IEEE Transactions on Neural Networks*, 13(3):678–686, 2002.

[18] D.J. Krusienski, M. Grosse-Wentrup, F. Galán, D. Coyle, K.J. Miller, E. Forney, and C.W. Anderson. Critical issues in state-of-the-art brain-computer interface signal processing. *Journal of Neural Engineering*, 8(2):025002, 2011.

[19] E. Gysels and P. Celka. Phase synchronization for the recognition of mental tasks in a brain-computer interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(4):406–415, 2004.

[20] Charles W. Anderson and Jeshua A. Bratman. Translating thoughts into actions by finding patterns in brainwaves. *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, pages 1–6, 2008.

[21] C.W. Anderson, J.N. Knight, T. O'Connor, M.J. Kirby, and A. Sokolov. Geometric subspace methods and time-delay embedding for eeg artifact removal and classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):142–146, 2006.

[22] N.F. Güler, E.D. Übeyli, and İ. Güler. Recurrent neural networks employing lyapunov exponents for eeg signals classification. *Expert Systems with Applications*, 29(3):506–514, 2005.

[23] E.D. Ubeyli. Analysis of eeg signals by implementing eigenvector methods/recurrent neural networks. *Digital Signal Processing*, 19(1):134–143, 2009.

[24] Lalit Gupta, Mark McAvoy, and James Phegley. Classification of temporal sequences via prediction using the simple recurrent neural network. *Pattern Recognition*, 33(10):1759–1770, 2000.

[25] Ikusaburo Kurimoto Shinichi Oeda and Takumi Ichimura. Time series data classification using recurrent neural network with ensemble learning. *Lecture Notes in Computer Science*, 4253:742–748, 2006.

[26] D. Coyle, G. Prasad, and T.M. McGinnity. A time-series prediction approach for feature extraction in a brain-computer interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 13(4):461–467, 2005.

[27] D. Coyle, T.M. McGinnity, and G. Prasad. Creating a nonparametric brain-computer interface with neural time-series prediction preprocessing. *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 2183–2186, 2006.

[28] D. Coyle, G. Prasad, and T.M. McGinnity. Extracting features for a brain-computer interface by self-organising fuzzy neural network-based time series prediction. *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, 2:4371–4374, 2004.

[29] D. Coyle. Neural network based auto association and time-series prediction for biosignal processing in brain-computer interfaces. *Computational Intelligence Magazine, IEEE*, 4(4):47–59, 2009.

[30] D. Coyle, T.M. McGinnity, and G. Prasad. A multi-class brain-computer interface with sofnn-based prediction preprocessing. *IEEE International Joint Conference on Neural Networks*, pages 3696–3703, 2008.

[31] Neuropulse-systems, llc. `http://www.np-systems.com`, 2011.

[32] Electro-cap international, inc. `http://www.electro-cap.com/`, 2011.

[33] D.R. Hundley, M.J. Kirby, and M. Anderle. Blind source separation using the maximum signal fraction approach. *Signal processing*, 82(10):1505–1508, 2002.

[34] J.N. Knight. Signal fraction analysis and artifact removal in EEG. *Masters Thesis, Department of Computer Science, Colorado State University, Fort Collins, CO.*, 2003.

[35] C.W. Anderson, J.N. Knight, T. O'Connor, M.J. Kirby, and A. Sokolov. Geometric subspace methods and time-delay embedding for EEG artifact removal and classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):142–146, 2006.

[36] Stefan C. Kremer. On the computational power of elman-style recurrent networks. *IEEE Transactions on Neural Networks*, 6:1000–1004, 1995.

[37] Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*, page 546. Springer, 1998.

[38] Y. LeCun. Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels, editors, *Connectionism in Perspective*, pages 143–155. Elsevier, 1989.

[39] Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2:490–501, 1990.

[40] Martin F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[41] C. Gruber and B. Sick. Fast and efficient second-order training of the dynamic neural network paradigm. *Proceedings of the International Joint Conference on Neural Networks*, 4:2482–2487, 2003.

[42] J. Friedman, R. Tibshirani, and T. Hastie. *The elements of statistical learning: Data mining, inference, and prediction*. Springer-Verlag New York, 2009.

[43] J.R. Pierce. *An introduction to information theory: symbols, signals & noise*. Dover, 1980.

[44] J.R. Wolpaw, H. Ramoser, D.J. McFarland, and G. Pfurtscheller. EEG-based communication: improved accuracy by response verification. *IEEE Transactions on Rehabilitation Engineering*, 6(3):326–333, 1998.

[45] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. ISBN 3–900051–07–0.

[46] K. Ericson, S. Pallickara, and C.W. Anderson. Analyzing electroencephalograms using cloud computing techniques. *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 185–192, 2010.

[47] Herbert Jaeger. Adaptive nonlinear system identification with echo state networks. *Advances in Neural Information Processing Systems (Proceedings of NIPS 15)*, 15:593–600, 2003.

[48] Herbert Jaeger. Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the echo state network approach. *GMD Report 159: German National Research Center for Information Technology*, 2002.