2014-06-25

Dynamic Feature Selection in a Reinforcement Learning Brain Controlled FES

Scott A. Roset
University of Miami, scott.roset@gmail.com

UNIVERSITY OF MIAMI

DYNAMIC FEATURE SELECTION IN A REINFORCEMENT LEARNING BRAIN CONTROLLED FES

By

Scott Roset

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

August 2014

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

DYNAMIC FEATURE SELECTION IN A REINFORCEMENT LEARNING BRAIN CONTROLLED FES

Scott Roset

Approved:

_____ Justin C. Sanchez, Ph.D. Associate Professor of Biomedical Engineering	_____ Edelle C. Field-Fote, Ph.D. Professor and Vice Chair of Physical Therapy
_____ Ozcan Ozdamar, Ph.D. Professor and Chair of Biomedical Engineering	_____ Chris Bennett, Ph.D. Research Assistant Professor of Music Engineering Technology
_____ Jorge E. Bohorquez, Ph.D. Assistant Professor of Professional Practice of Biomedical Engineering	_____ M. Brian Blake, Ph.D. Dean of the Graduate School

ROSET, SCOTT (Ph.D., Biomedical Engineering)

Dynamic Feature Selection in a Reinforcement Learning Brain Controlled FES. (August 2014)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Justin C. Sanchez.
No. of pages in text. (79)

Each year, more than 10 people per million will incur a spinal cord injury (SCI). Of these injuries, one-third is reported to result in tetraplegia. People living with tetraplegia rank hand function as the ability they would most like to see restored. With decrease use of hand movements, plastic reorganization causes secondary damage in the motor cortex. Methods are needed to help restore or supplement motor abilities.

One approach to produce a more comprehensive therapy is to augment standard rehabilitation with new developments from the study of Brain-Computer Interfaces (BCI). BCI's record brain activity and translate it into actions in the physical world. BCI's do this by decoding electroencephalography (EEG) data with a computer system to determine a user's intent. By engaging the user's brain to actively control extremities during rehabilitation, BCI's combined with rehabilitation could offer the unique ability to rehabilitate the motor system as a whole, including secondary damage in the motor cortex. Not all EEG signals can be directly mapped to desired outputs; however including some of them may improve the performance of the BCI. One possible EEG signal to include in a BCI is ErrPs. These potentials occur when the subject notices an error has been

made. A new BCI architecture that incorporated reinforcement learning and ErrPs could better process the EEG signal.

To validate the reinforcement learning based BCI for rehabilitation a closed-loop system was developed. The system presented cues to the user instructing them to perform motor imagery thus generating motor potentials. The system then provided feedback to the user through a display and functional electrical stimulation (FES), which caused the user to generate an ErrP if an error occurred. The system was able to use reinforcement learning to determine the mapping of motor potentials to intended actions based on user generated ErrPs.

Choosing an appropriate size for a neural network when using reinforcement learning for a BCI application is difficult because of the bias-variance tradeoff. By starting with a small network and using dynamic feature addition to grow the number of inputs to the network over time the performance of the BCI can be improved over both small and large networks in both early trials and later trials. The order in which features are added during dynamic feature addition can affect the performance of the system. By taking into account how useful features are for discriminating between different cues and adding features that are more useful in early trials, the performance of the system can be improved.

Various update rules could be used in the rehabilitation system: back propagation, scaled back propagation, Hebbian style learning, and scaled Hebbian style learning. In simulations Hebbian style learning performed better than back propagation. While scaled Hebbian style learning performed better

than Hebbian style learning. Scaled Hebbian style learning also takes advantage of the online nature of reinforcement learning used in the system. By adjusting the learning rate, the algorithm adapted the weights more quickly in areas where the slope of the error surface is small and converges on a minimum more quickly in areas where the slope is high.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACO	Ant Colony Optimization
ASIA	American Spinal Injury Association standards
BCI	Brain-Computer Interface
DNAL	Dynamic Node Architecture Learning
DNC	Dynamic Node Creation
EEG	Electroencephalography
ERD	Event-Related Desynchronization
ERN	Error-Related Negativity
ErrP	Error-Related Potential
ERS	Event-Related Synchronization
FES	Functional Electrical Stimulation
FFT	Fast Fourier Transform
MLP	Multilayer Perceptron
PSD	Power Spectral Density
RL	Reinforcement Learning
SCI	Spinal Cord Injury
SD	Standard Deviation
SMR	Sensorimotor Rhythms
SSVEP	Steady State Visual Evoked Potentials
TMS	Transcranial Magnetic Stimulation

Chapter 1: INTRODUCTION

Each year, more than 10 people per million will incur a spinal cord injury (SCI). Of these injuries, one-third is reported to result in tetraplegia (Wyndaele and Wyndaele 2006). People living with tetraplegia rank hand function as the ability they would most like to see restored (Anderson 2004). With decrease use of hand movements, plastic reorganization occurs in the brain (Hoffman and Field-Fote 2006). Methods are needed to help restore or supplement motor abilities because of these diseases and injuries. To alleviate motor disabilities caused by spinal cord injury will require rehabilitation that restores top-down (brain activity) control of the motor system.

1.1 REHABILITATION

While individual therapies can be combined to treat the motor system, few therapies treat the motor system as a whole or have other drawbacks. For example massed practice, repetitive movements for several hours a day over a period of weeks, has been shown to cause positive cortical changes (Hoffman and Field-Fote 2007). However, massed practice has the limitation that the subject must have at least a limited ability to perform the movements before therapy.

One type of massed practice rehabilitation is constraint-induced movement therapy. In constraint-induced movement therapy, the use of

unaffected hand is limited to force the use of the affected hand (Liepert, Bauder et al. 2000). The subject needs to be able to perform movements with the affected hand even more effectively than with massed practice, since the affected hand will now be the only means to perform daily activity. Constraint-induced movement therapy also relies on the assumption that subjects have been using their unaffected hand and neglecting their affected hand. So, this technique will only work on a subject with only one affected hand.

Massed practice rehabilitation have the subject perform actions without assistance. However, some interventions use external assistance to increase the effectiveness of rehabilitation. Electromyography signals can be used to control an orthosis that mechanically extends the fingers. The orthosis enables subjects to practice moving their affected hands, if they can control their electromyography signals (Ochoa, Listenberger et al. 2011). By giving users feedback of their electromyography signals, users are also able to increase control of their muscle activity. With this intervention, users still need to have some control of their affected hands remaining, to control their electromyography signals.

Another rehabilitation approach uses transcranial magnetic stimulation (TMS) (Hummel and Cohen 2006). During TMS a coil is placed over the subject's head and a magnetic field is generated within the subject's cortex. The magnetic field generates electrical currents within the cortex, depolarizing neurons. TMS is used to excite neurons associated with the affected hand and sometimes inhibit neurons of the unaffected hand. Only a component of the motor system, the

motor cortex, is targeted during TMS, and the motor system is not rehabilitated as a whole. The subject is also a passive participant during the rehabilitation and exerts no control over the stimulation.

One approach to produce a more comprehensive therapy is to augment standard rehabilitation with new developments from the study of Brain-Computer Interfaces (BCI). BCI's record brain activity and translate it into actions in the physical world (McFarland and Wolpaw 2011). BCI's do this by decoding electroencephalography (EEG) data with a computer system to determine a user's intent. By engaging the user's brain to actively control extremities during rehabilitation, BCI's combined with rehabilitation could offer the unique ability to rehabilitate the motor system as a whole, including secondary damage in the motor cortex (Daly and Wolpaw 2008).

1.2 BRAIN – COMPUTER INTERFACES

Brain control paradigms can broadly be divided into two categories: those that record brain activity from intracranial electrodes and those that records brain activity from external EEG electrodes. EEG has an advantage of being noninvasive, which especially important in human subject research (Millan, Renkens et al. 2004). EEG is also portable and has high temporal resolution.

EEG can be classified into two types of potentials, spontaneous brain rhythms and evoked potentials. Evoked potentials occur at a fixed time after a stimulus enabling them to be averaged over tens of trials to increase the signal-

to-noise ratio. Some BCIs use the evoked potential called P300, a positive potential that occurs 300msec after the stimulus. The oddball paradigm, rare events mixed in with more common events, generates the P300 potential when the user notices the rare event. BCI systems based on P300 potentials are slower than other BCI systems since all possible choices must be cycled through. This drawback limits P300 based system to applications where time is not a priority, such as spellers (McFarland and Wolpaw 2011).

Another evoked potential used in BCIs are steady state visual evoked potentials (SSVEP). They were the first EEG signals used in a BCI, Jacques Vidal (Vidal 1977) used the term “brain-computer interface” to describe his research with SSVEP in the 1970s. SSVEP are evoked potentials that match the frequency of the stimulus. In SSVEP based BCI systems, every choice has a unique stimulus frequency associated with it. The system is able to determine which choice the user is looking at based off the frequency of the evoked potentials. SSVEP based BCIs are also slower than other BCI approaches because the evoked potentials do not reach a discernible amplitude quickly (Middendorf, McMillan et al. 2000).

Another spontaneous brain rhythm used as a signal for a BCI is sensorimotor rhythms (SMR), potentials related to movement or imagined movement. Initiating real hand movement or imagining hand movement causes an event-related desynchronization (ERD) in the area of the brain associated with hand movement. Initiating real hand movement or imagining hand movement also causes an event-related synchronization (ERS) in motor regions

not associated with hand movement. Once the hand movement or imagined hand movement stops, ERS is detected in the area of the brain associated with hand movement (Neuper, Wörtz et al. 2006).

Wolpaw et al. (McFarland and Wolpaw 2011) were the first to use SMRs for cursor control in 1991. SMR based BCIs main drawback is that they are susceptible to degradation in performance without frequent training sessions. In general, EEG based BCIs were thought to have too slow a bit rate to do complicated tasks, however Millan et al. (Millan, Renkens et al. 2004) demonstrated asynchronous control of a mobile robot with EEG based BCI. The mobile robot had sensors and its own limited intelligence, reducing the required bitrate. For rehabilitation of hand function, imagined hand movement would be the most appropriate control signal for a BCI system, since imagined hand movement is most closely related to the target of the rehabilitation.

Not all EEG signals can be directly mapped to desired outputs; however including some of them may improve the performance of the BCI. One possible EEG signal to include in a BCI is error-related potentials (ErrP). These potentials occur when the subject notices an error has been made. Different ErrPs have been found for different types of error the user observed: response ErrP, feedback ErrP, observation ErrP, and interaction ErrP. Response ErrP, when a subject makes an error in a choice selection task with a time limit. Feedback or reinforcement ErrP, when a subject knows an error was made from feedback. Observation ErrP, when a subject watches another person make an error, and interaction ErrP, when a subject controlling a device sees it did not follow a

command. Components of ErrPs were first described by Hohnsbein and Falkenstein in 1989 (Hohnsbein, Falkenstein et al. 1989). They termed the components error negativity, Ne, and error positivity, Pe. A similar potential related to errors was found by Gehring et al. (Gehring, Coles et al. 1990) in 1990. Gehring termed the potential as error-related negativity, ERN. Recently, the various potentials associated with error have been classified under the term ErrPs (Ferrez and Millan 2008).

ErrPs increase in amplitude with the degree of the error made. ErrPs main component is a negative potential 250 ms after feedback. ErrPs most likely originate in an area of the brain responsible for regulation of emotional responses, the anterior cingulate cortex. The EEG signals generated by the ErrP have a fronto-central distribution along the midline and are most prominent on the FC_Z and C_Z locations. Since ErrP are relatively slow cortical potentials, they can be acquired with a 1-10 Hz bandpass filter. Ferrez et al. (Ferrez and Millan 2008) describe the successful classification of signals generated shortly after feedback as either a correct response or an ErrP. To generate the ErrP, a target and cursor were shown on a display. The user then had to press the left or right key to move the cursor to the target. 50% of the time, the cursor moved in the opposite direction of the button pressed. Classifiers created for each subject to detect ErrP from the first day of recordings were able to recognize the ErrP over 82% of the time for both subjects on the second day.

ErrPs originating in the anterior cingulate cortex along with the mesencephalic dopamine system have been proposed to be part of a

reinforcement learning (RL) system in the brain. The anterior cingulate cortex acts as a filter that selects between different motor controllers. The mesencephalic dopamine system then evaluates outcomes. The mesencephalic dopamine system increases dopamine when outcomes are better than anticipated and decreases dopamine when outcomes are worse than anticipated. When less dopamine is released in the anterior cingulate cortex, an ErrP is generated in the anterior cingulate cortex. When more dopamine is available, behaviors of the anterior cingulate cortex and certain motor controllers are reinforced (Holroyd and Coles 2002). ErrPs and RL could be incorporated into a BCI; however a new BCI architecture is needed.

1.3 NEW ARCHITECTURE WITH REINFORCEMENT LEARNING

The BCI architecture commonly used today forms one closed-loop: modulation in the user's brain activity is detected by EEG, features are extracted from these EEG signals, and a classifier uses these extracted features to determine the user's intent, Figure 1.1. Previous attempts to improve BCI performance tried to optimize this architecture: better EEG signals, better feature extraction, better classifiers, or a combination of these approaches.

However, the current BCI architecture has several drawbacks. The BCI is usually trained through supervised learning (DiGiovanna, Mahmoudi et al. 2009). This training requires the users to perform a boring training task that does not engage their attention every day. The training must be repeated daily because

the BCI is static, a fixed input-to-output mapping, between training sessions. Yet, brain activity changes constantly as the user adjusts to the BCI.

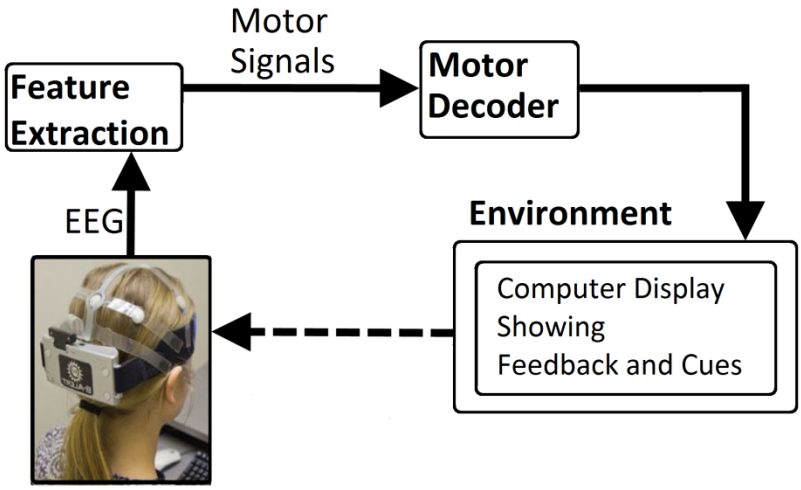

Figure 1.1 Standard BCI Architecture – static feature extraction and decoding during use.

A new BCI architecture that incorporated RL could avoid these drawbacks (Mahmoudi, DiGiovanna et al. 2008). In RL an agent maximizes its rewards from the environment by critically adapting its behaviors. With experience, the agent learns how to maximize rewards for a state of the environment by comparing outcomes to expectations.

There are several RL schemes, however the actor-critic model fits the BCI augmented rehabilitation application the best (Mahmoudi and Sanchez 2011). In the actor-critic model, the agent is comprised of two parts, an actor and a critic. The actor is a policy (π) with parameters (θ) that maps user's brain states (s_t) to actions (a_i).

$$\pi(a \mid s\ ;\theta) = \Pr(a_t = a \mid s_t = s)$$

The critic provides a reinforcement signal to adapt the actor's parameters (θ) by estimating reward at each time step to form the reward expectation (ν).

$$\nu_t(s,a) = E\left[r_{t+1} \mid s_t = s, a_t = a\right], \forall s \in S, \forall a \in A$$

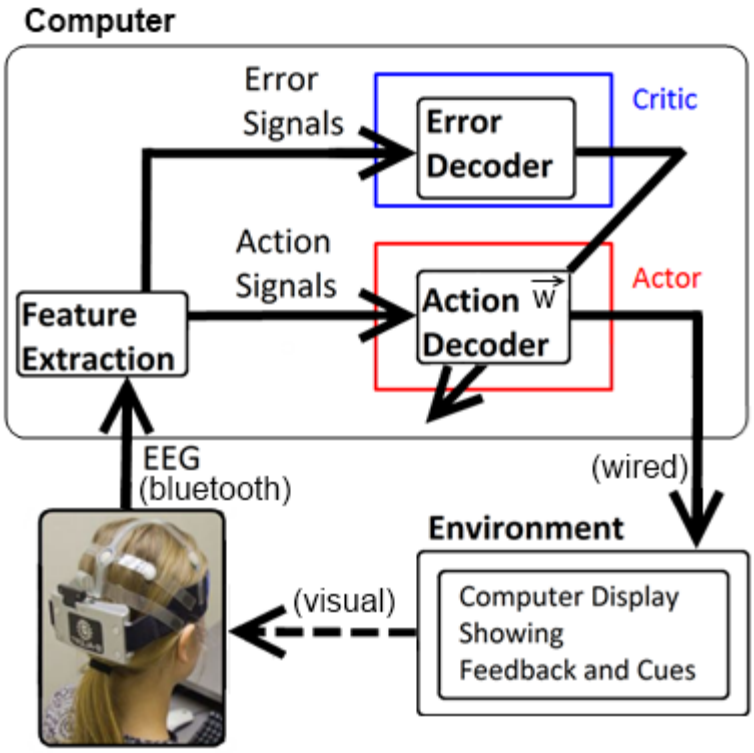

Figure 1.2 Reinforcement Learning BCI Architecture – dynamic decoding of motor potentials.

Figure 1.2 shows an architecture based on the actor-critic model that uses the information present in ErrPs to better process the EEG signal. The actor decodes the action signals, mapping them to desired outputs. The critic decodes

the error signals, detecting ErrPs and using them to improve the performance of the actor. The new BCI architecture could help rehabilitate the motor system as a whole, especially the motor cortex through brain plasticity.

1.4 BRAIN PLASTICITY

By combining a BCI's ability to bypass the injury and connect the motor cortex to the spinal motoneurons that control the affected limbs and RL's ability to increase rehabilitation time during a session, the motor cortex could be rehabilitated through brain plasticity for subjects with complete paralysis. In a similar way that massed practice techniques rehabilitate the motor cortex for subjects with partial paralysis. Donald O. Hebb (Hebb 1949) proposed in the 1940's that the brain's structure can be changed by experiences and the environment, a process now called brain plasticity. Rats trained in a task that required wrist and digits movement showed an increase in area devoted to the wrist and digits in the motor cortex at the expense of area devoted to the elbow and shoulder (Kleim, Barbay et al. 1998). Similar changes have also been seen in humans. In violin players, the digits of the left hand have larger cortical area than the right hand and are larger than people who do not play violin. The area representing the left hand digits also increases with how long the violin has been practiced (Elbert, Pantev et al. 1995). Braille readers show increased cortical area devoted to their reading fingers compared to fingers on the non-reading hand or to non-braille readers. The cortical area devoted to the reading finger

expands at the expense of other fingers (Pascual-Leone, Cammarota et al. 1993). The brain is also very dynamic and changes during everyday activities. While a new task is being learned, cortical area devoted to the task shows a temporary increase in size and returns to the original size once the task is mastered (Pascual-Leone, Grafman et al. 1994).

Brain plasticity can also reduce the effects of injuries that destroy brain tissue within a cortical area. When cortical areas associated with hand functions were damaged by a focal ischemic infarct in squirrel monkeys, they showed a dramatic reduction in hand function (Nudo and Milliken 1996). In further studies, the monkeys received rehabilitation after the infarct, the cortical areas associated with hand function moved into adjacent tissue and hand function was partially restored (Nudo, Wise et al. 1996).

Similar results have been seen in humans. In stroke patients receiving constraint-induced movement therapy for hand rehabilitation, the area and location of the affected hand's cortical region was shown to change, as measured by TMS. In a six month follow-up, the cortical area of the affected hand increased in size to match the unaffected hand's map, showing long-term improvement (Liepert, Bauder et al. 2000). Rehabilitation has also been shown to make positive changes in cortical area following a SCI (Hoffman and Field-Fote 2007; Beekhuizen and Field-Fote 2008; Hoffman and Field-Fote 2010). Incorporating a RL BCI into rehabilitation could accelerate and improve the positive changes in cortical area.

Chapter 2: BRAIN-COMPUTER INTERFACE AUGMENTED REHABILITATION

2.1 OVERVIEW

To validate the RL based BCI for rehabilitation a closed-loop system was developed. The system presented cues to the user instructing them to perform motor imagery thus generating motor potentials. The system then provided feedback to the user through a display and functional electrical stimulation (FES), which caused the user to generate an ErrP if an error occurred. The various components of the system were tested. An ErrP detector was created and tested for each user to ensure ErrPs were detected during the closed-loop experiment. A closed-loop experiment with the system starting with no prior knowledge of the user's motor potentials was conducted with each user. We compared the decoder characteristics in the closed-loop environment between controls and subjects living with SCI. With these tests and experiments, we hope to learn if a RL based BCI can learn the motor potentials of a user leading to future work as a rehabilitation system.

2.2 METHODS

Study Participants

The system design function was demonstrated and compared between a control subject and a subject with a chronic SCI. All procedures followed in the study were approved by the University of Miami Institutional Review Board. The subjects provided written informed consent. The inclusion criteria followed for recruiting subjects with SCI included: chronic injury (longer than 1 year), no denervation of target muscles, and C5 or C6-level motor complete injury classified by the American Spinal Injury Association (ASIA) standards (Marino, Barros et al. 2003). Both subjects were 30 years old males. The subject with SCI was injured playing football, and his injury (duration = 15 years) was classified by ASIA standards as incomplete (ASIA B), with bilateral motor levels of C6. Motor scores of 5 (normal function) were attained at the C5-level bilaterally, with scores of 5 (right) and 3 (left) at the C6-level. All motor scores below level C6 were zero. The subjects had no history of other serious medical issues.

Experimental Task

Hand grasp/open function was chosen as the experimental task as restoration of hand/arm function is the highest priority for people with tetraplegia (Anderson 2004). The goal of the task was to enable direct brain actuation of

hand closing and opening. In addition to extracting motor potentials, an evoked potential from the brain was of interest: ErrP which are generated when an error is observed. In this experiment, ErrPs were generated when the user perceived the action of the BCI was incorrect. Both motor and error potentials are necessary for conducting closed-loop RL in this context.

A preliminary session was used to collect representative ErrPs to develop an ErrP classifier. During the preliminary session, feedback was random and approximately 50% of the 120 trials resulted in a “wrong” outcome. No stimulation was delivered to the subjects during the preliminary session. The subjects sat facing a display with their right forearm resting on a table (Figure 2.1A). After a fixation cross was shown on the display for three seconds to minimize eye movements, cues of “open” or “close” were presented for one second that instructed the person to either open or close his hand. Random visual feedback of “correct” or “wrong” was then shown for one second, along with a corresponding plot of the unthresholded output of the system (Figure 2.1B:row 3).

Four closed-loop sessions were performed and consisted of 300 trials during the 1st session, 450 trials each during the 2nd and 3rd sessions, and 300 trials during the 4th session. Time between sessions was varied to test the adaptation of the network with two days between the 1st and 2nd sessions, four days between the 2nd and 3rd sessions, and one day between the 3rd and 4th sessions. During closed-looped sessions, ErrPs were collected and used to adapt the BCI. The same visual cues were displayed on the screen as in the

preliminary session. However, in closed-loop sessions the displayed feedback matched the output of the adaptive BCI. When the output of the adaptive BCI was determined to be “open,” FES was delivered to the hand muscles of the SCI subject. No FES was delivered for trials when the output of the adaptive BCI was “close”. All trials were used in the analysis.

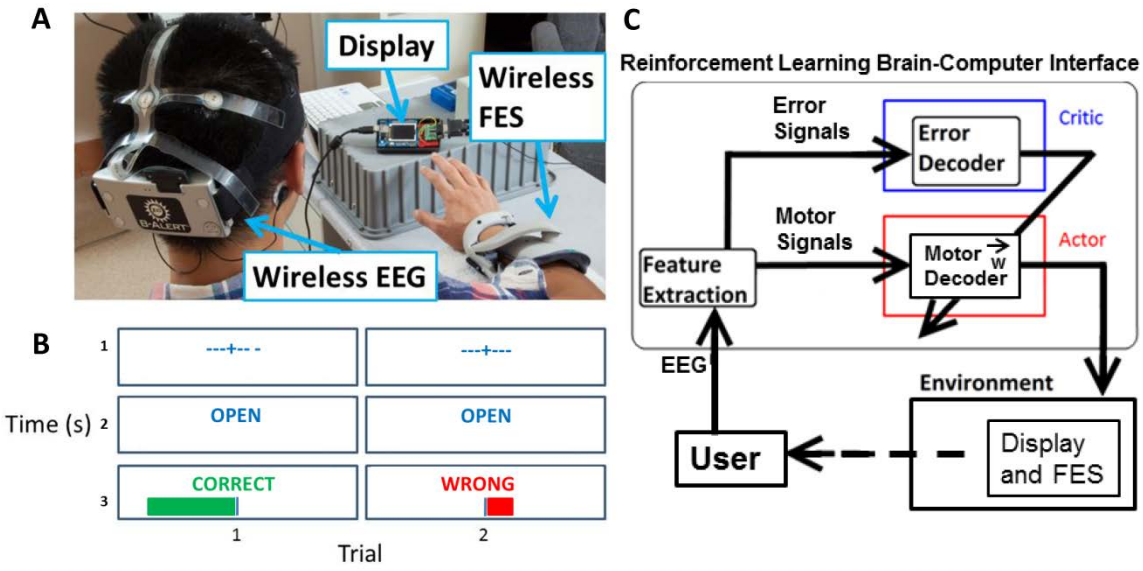

Figure 2.1 (A) Experiment setup overview, visible are the EEG headset, display, and FES. (B) For each trial during the experimental task, the display showed a fixation cross, followed by a cue for “open” or “close” for 1s, and then feedback of “correct” or “wrong” for 1s. A magnitude plot also showed the unthresholded output of the motor potentials decoder. (C) Actor-critic RL BCI architecture. The actor decodes motor potentials and outputs an action. The critic detects an ErrP and provides feedback to actor. The actor uses feedback from the critic to adapt to the user.

Neural Data Acquisition

A wireless 9-channel EEG system (256 Hz sampling rate, 16-bits of resolution, X10 headset, Advanced Brain Monitoring, Carlsbad, CA) was fitted to

the subject's head (Figure 2.1A). Electrodes (F_Z, F_3, F_4, C_Z, C_3, C_4, PO_Z, P_3, P_4) were arranged according to the International 10-20 system standards. Foam sensors attached to the sensor sites on the headstrips were saturated with Synapse (Kustomer Kinetics, Arcadia, CA) conductive electrode paste and the corresponding sites on the head were abraded and cleaned before placing the sensors on the scalp. Electrode impedances were tested before and after each experimental session using the manufacturer provided software.

ErrPs were recorded from the C_Z electrode and the motor potentials for the intent to open or close the hand were recorded from the C_3 electrode (Qin, Ding et al. 2004; Ferrez and Millan 2008). For ErrPs, EEG generated from 0.15 to 0.70 seconds after display of feedback ("correct" or "wrong") was used. For motor potentials, EEG generated between 0.15 and 1.0 seconds after the display of cues ("open" or "close") was used. EEG was transformed into the frequency domain using the Fast Fourier Transform (FFT) to obtain a power spectral density (PSD) of 1 Hz resolution. Frequencies of 1-50 Hz were used for the motor potential decoder and frequencies of 1-12 Hz were used for the ErrP decoder. The inputs to both decoders were normalized PSD z-scores (LeCun, Bottou et al. 1998). The z-scores of the PSD were created by subtracting the mean of previous trials at each frequency and dividing by the standard deviation (SD) of previous trials for that frequency.

Muscle Stimulation

A neuroprosthetic wrist-hand orthosis (NESS H200, Bioness Inc, Valencia, CA) was fitted to the right hand of the subject. FES was delivered to the extensor muscles (extensor digitorum communis and extensor pollicis brevis) to produce opening movements of the fingers and hand. Stimulation intensity was set by holding the pulse duration (300 μs) and frequency (35 Hz) constant, while slowly increasing the current amplitude. Once a maximal muscle contraction was attained (i.e., increases in current intensity did not produce additional muscle contraction), the current amplitude was increased an additional 25% in order to maintain consistent muscle contractions throughout the experiment.

Actor-Critic Reinforcement Learning Architecture

The adaptive BCI is based on an actor-critic RL architecture (Figure 2.1C) (Mahmoudi and Sanchez 2011). The actor decodes motor potentials from the user to determine the user's intent to open or close the hand. The critic provides feedback to the actor by detecting ErrPs generated by the user (Falkenstein, Hoormann et al. 2000). The actor-critic RL algorithm is a semi-supervised machine learning algorithm that optimizes the actor’s decoding of the user’s motor potentials based on feedback from the critic (Sutton and Barto 1998).

The actor is parameterized by a 3-layer fully connected feedforward neural network. The hidden and output processing elements of the neural network

perform a weighted sum on their inputs. The weighted sum at each processing element is passed through a hyperbolic tangent function with an output in the range of -1 to 1. The weights between the actor's processing elements are initialized randomly and then updated after each trial based on feedback. The actor's weights update can be expressed as:

$$\Delta \boldsymbol{w_{ij}} = \gamma f \left(\boldsymbol{x_i}(\boldsymbol{p_j} - \boldsymbol{x_j}) \right) + \gamma (1 - f) \left(\boldsymbol{x_i}(\mathbf{1} - \boldsymbol{p_j} - \boldsymbol{x_j}) \right) \quad (2.1)$$

Here w_{ij} is the weight connecting processing elements i and j, γ is the learning rate, p_j is a sign function of output x_j (positive values become +1 and negative values become -1) and *f* is feedback from the critic. The weight update equation is based on Hebbian style learning (Mahmoudi and Sanchez 2011; Pohlmeyer, Mahmoudi et al. 2012). The critic provides the feedback by decoding the user's EEG to determine if an ErrP was generated. If an ErrP is detected, a feedback of -1 is provided to the network for adaptation. If not, a feedback value of 1 is given. The functional mapping between neural activity and behavior in the actor is constructed using the weight update equation (Equation 2.1).

Adaptive BCI Usage

Adaptive BCI usage was broken down into several intermediate steps (Figure 2.2). Representative ErrPs were collected in the preliminary session and used to develop the critic through supervised learning (Prechelt 1998). Once the

critic was created, the weights of the actor were initialized to random initial values and trained through RL and feedback from the critic. After the first closed-loop session, in which the weights are initialized to random values, all subsequent closed-loop sessions used the weights from the previous session with no offline adjustments.

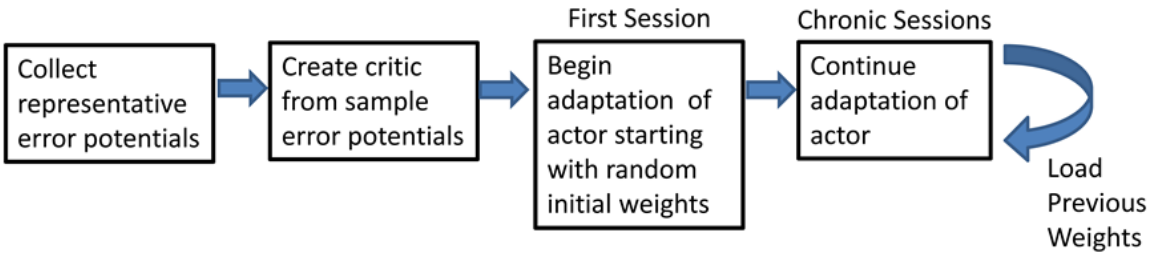

Figure 2.2 Flowchart shows the preliminary steps of the experiment and how the final step can be repeated.

Critic as Error Potential Classifier

The error potential classifier "critic" detects ErrPs in the user's EEG to determine if the user perceived that an error occurred. The critic then provides binary feedback, -1 or 1, to the actor. The input to the error potential classifier was the normalized PSD from 1-12 Hz in 1 Hz bins computed on the 0.15 to 0.70 seconds of EEG data after the actor's output (action) was shown on the display.

The error potential classifier in the critic is a 3-layer neural network with 12 inputs processing elements, for the 1-12 Hz in 1 Hz bins, and 5 hidden processing elements. Representative ErrPs were collected in the 120 trials of the preliminary session and were randomly assigned to either a training set or test

set, approximately 60 trials each. The training set was used to optimize the weights of the critic with supervised learning. The weights produced from the supervised learning were assessed by applying them to the test set and computing the classification accuracy. The weights with the best classification accuracy were used for closed-loop sessions.

To test the critic training procedure during the preliminary data collection, 10 training and testing data sets were created by randomly assigning trials to either set, a 10-fold cross validation (Table 2.1). The minimum and maximum accuracy in the 10-fold cross validation were within 5% of the mean accuracy, showing that the critic should have reasonable performance during the closed-loop sessions.

SCI

	Predicted Correct	Predicted Error
Correct	66.8%± 4.1	33.2%± 4.1
Error	40.8%± 5.0	59.2%± 5.0

Control

	Predicted Correct	Predicted Error
Correct	68.6%± 2.6	31.4%± 2.6
Error	30.9%± 3.6	69.1%± 3.6

Table 2.1 10-fold cross validation classification results of the critic for both the healthy and SCI subject.

2.3 RESULTS

Closed-Loop Trials

Figures 2.3 and 2.4 show representative trials from the closed-loop experiments and give insight into how the system processes the EEG to create

features for the classifiers. The first row of Figure 2.3 shows the filtered (1-50 Hz) EEG from the C_3 electrode for the 0.15 to 1.0 seconds after the cue is presented. The second row shows the PSD computed from the raw EEG. The z-scores of the PSD are shown in the third row as inputs to the actor. The first column shows the filtered EEG and processing after an "open" cue. Similarly, the second column shows the filtered EEG and processing after a cue of "close" was shown. The features for the cue of "close" correspond to lower power, in general, than the features of the cue for "open"; in the sample trial of the SCI subject, 44 of the 1 Hz bins have lower power for the "close" cue.

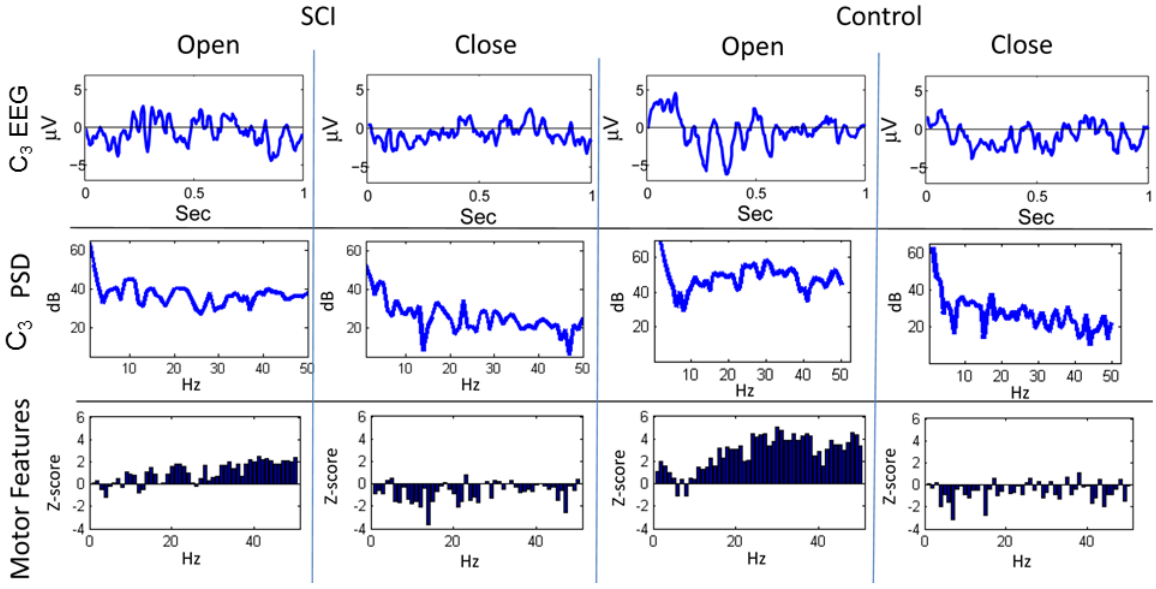

Figure 2.3 Sample trials from closed-loop sessions. Columns show samples for cues and feedback of "open" and "close" for both the SCI and control subject. Rows show filtered EEG (1-50 Hz) from electrode **C_3**, PSD, and motor features.

A similar process was used for inputs to the critic. The first row of Figure 2.4 shows the filtered, 1-12 Hz, EEG from the C_Z electrode for the 0.15 to 0.70

seconds after the feedback was shown. PSD of the raw EEG was computed from the C_Z electrode, shown in the second row. Finally, the inputs to the critic are shown in the third row as z-scores of the PSD from the C_Z electrode. The first column shows the filtered EEG and processing after the feedback of "correct" was presented. The second column shows the filtered EEG and processing for feedback of "error." Notice that the error potential has a biphasic shape characteristic of this neural oscillation. The features for feedback of "correct" correspond to lower power, in general, compared to features of "error;" in the sample trial for the SCI subject, all 1 Hz bins except 1, 8, 11, and 12 Hz. Figure 2.5 shows the ErrPs generated by the users, the average of error trials minus the average of correct trials. The ErrPs collected from the users are similar to published results (Ferrez and Millan 2008).

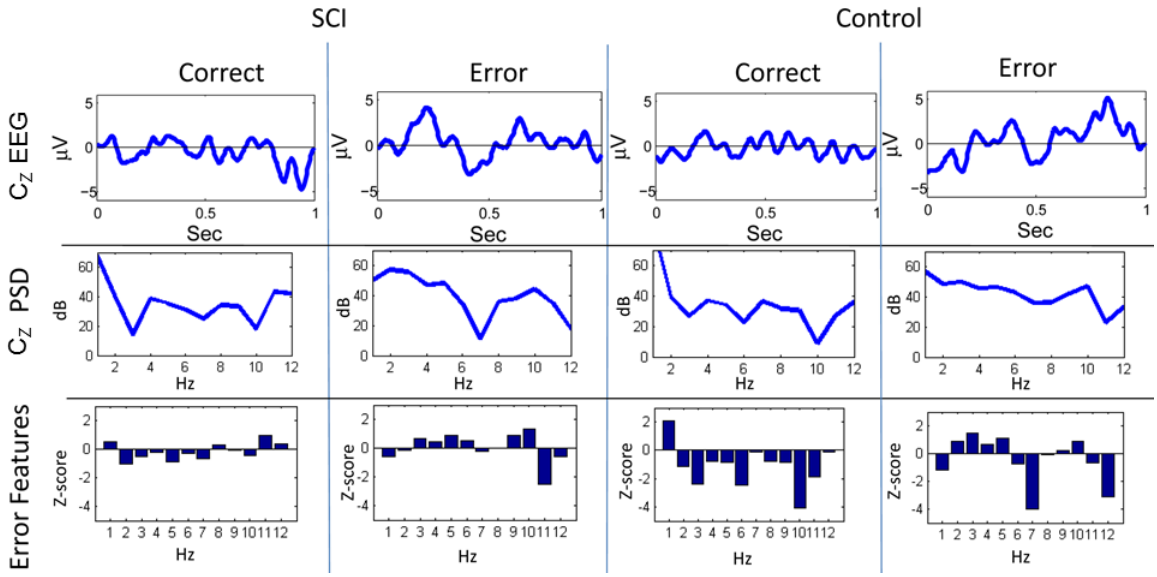

Figure 2.4 Sample trials from closed-loop sessions. Columns show samples for cues and feedback of "correct" and "error" for both the SCI and control subject. Rows show filtered EEG (1-12 Hz) from electrode **C_Z**, PSD, and error features.

Performance of the System

Figure 2.6 shows the overall performance of the actor in classifying motor potentials across 4 sessions, for control and SCI subjects. The classification accuracy starts below 50% (chance level) for the SCI subject, due to the random initial values of the actor's weights. The performance of the actor improves as the actor's weights adapt to feedback from the critic through RL. Over time, the actor's performance approaches the classification accuracy of the critic. Changes in weight values become smaller after the first 2 sessions; however, changes in weight values continue throughout the 1500 trials. The actor made fewer mistakes during the last session than the first, as the actor adapted and learned the user's motor potentials based on feedback from the critic.

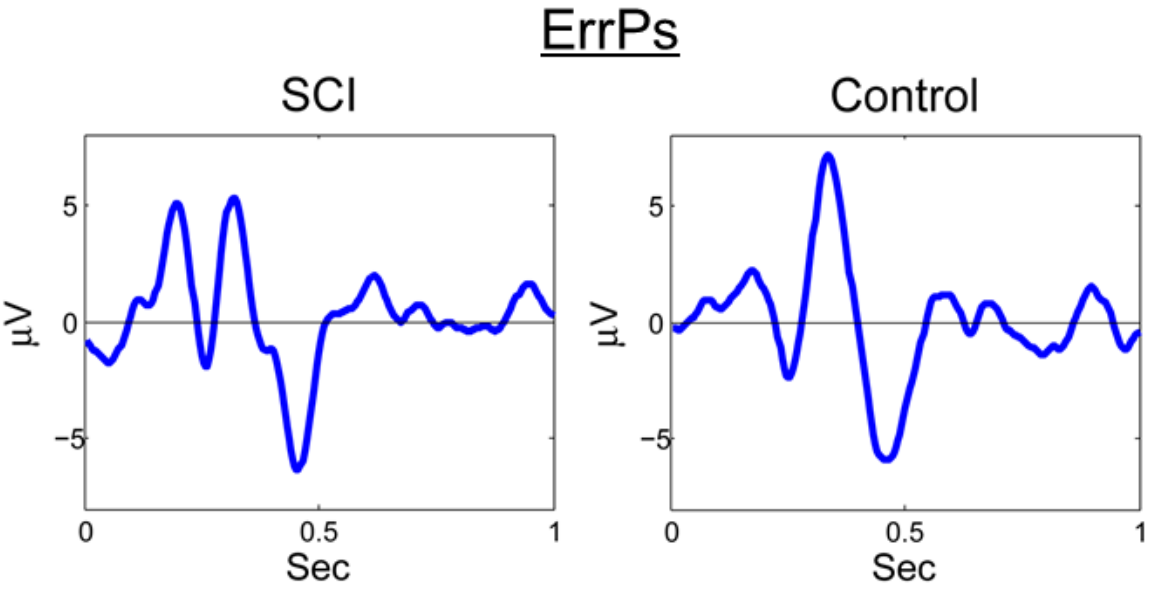

Figure 2.5 Average EEG for the difference error–minus–correct trials at channel $\mathbf{C_z}$ for the SCI and control subjects. Feedback is delivered at time 0 s.

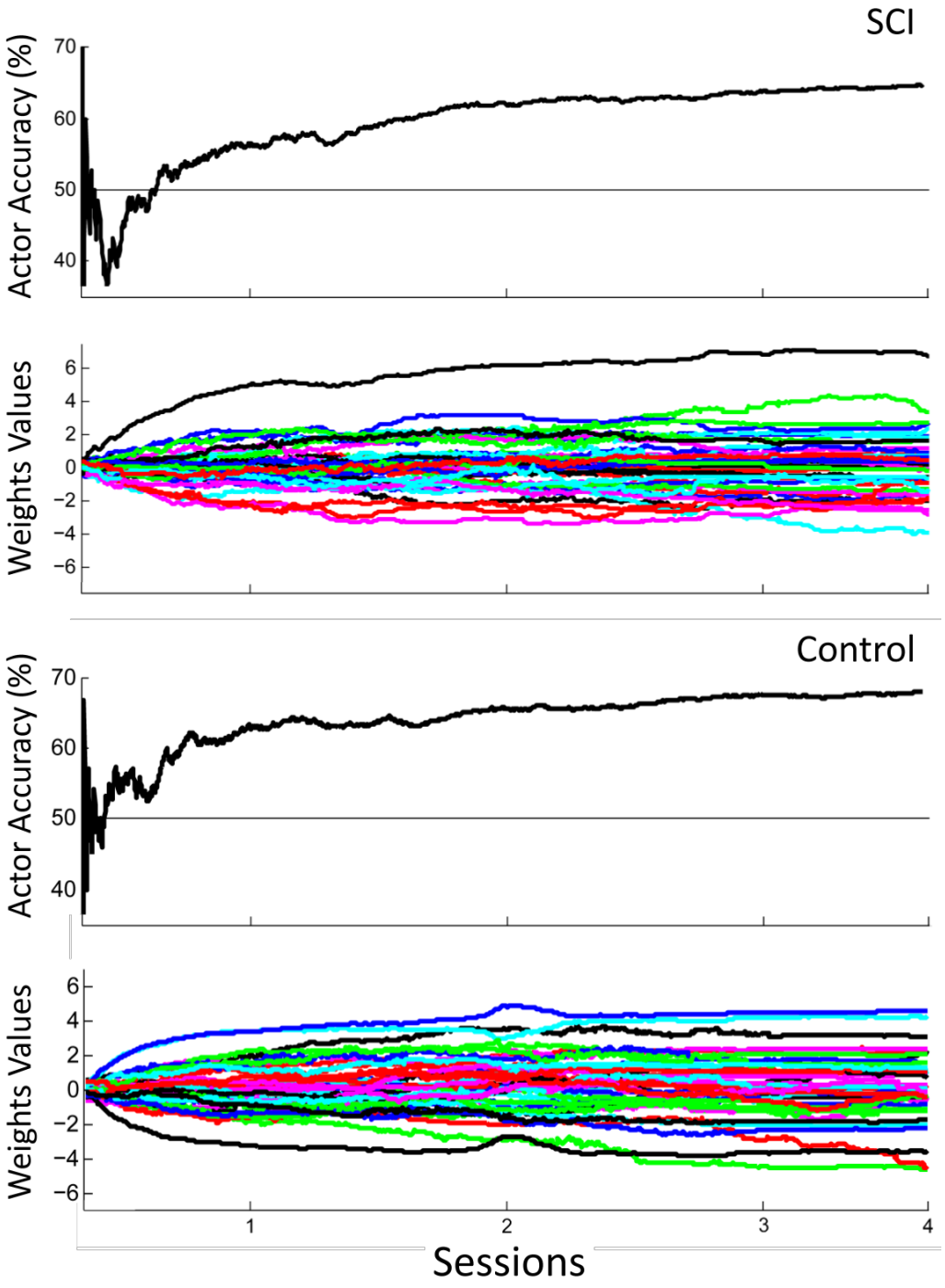

Figure 2.6 Actor's performance across 4 sessions. The first row shows the actor's cumulative classification accuracy and the second row shows the actor's weights adapting for the SCI subject. The third row shows the actor's cumulative classification accuracy and the fourth row shows the actor's weights adapting for the control subject.

Comparison of Performance across Subjects

The overall performance of both subjects across sessions is shown in Figure 2.7. The subjects had comparable performance, above chance level (50%) starting at the end of the first session. The performance of the control subject was slightly higher than that of the SCI subject during the first session. This performance difference can be explained by the random initial weight values of the actor more closely matching the desired weight values by chance. The overall performance of the SCI subject was only slightly lower than the control subject, by 0.9%. The system also had lower accuracy for detecting the SCI subject's ErrPs, 64.2%, than for the control subject, 68.8%. This lower performance in detecting the SCI subject's ErrPs could explain the lower overall performance of the SCI subject compared to the control subject. Importantly, the performance of the critic had a small SD, 3.6% for the SCI subject.

2.4 DISCUSSION

This study showed a new EEG based BCI system using RL intended for application to control of a FES and developed as an experimental test bed for augmenting rehabilitation with a BCI. The system used RL to determine the mapping of motor potentials to intended actions based on user generated ErrPs. The BCI continued to adapt to the users throughout the experiment and did not require any offline training after the first session. Comparable performance levels were achieved for the control and SCI subject. The ability to adapt to the user

without daily initialization could be beneficial in a rehabilitation setting. Cortical reorganization from the rehabilitation could change the user's motor potentials, increasing the need for daily adjustments to the system.

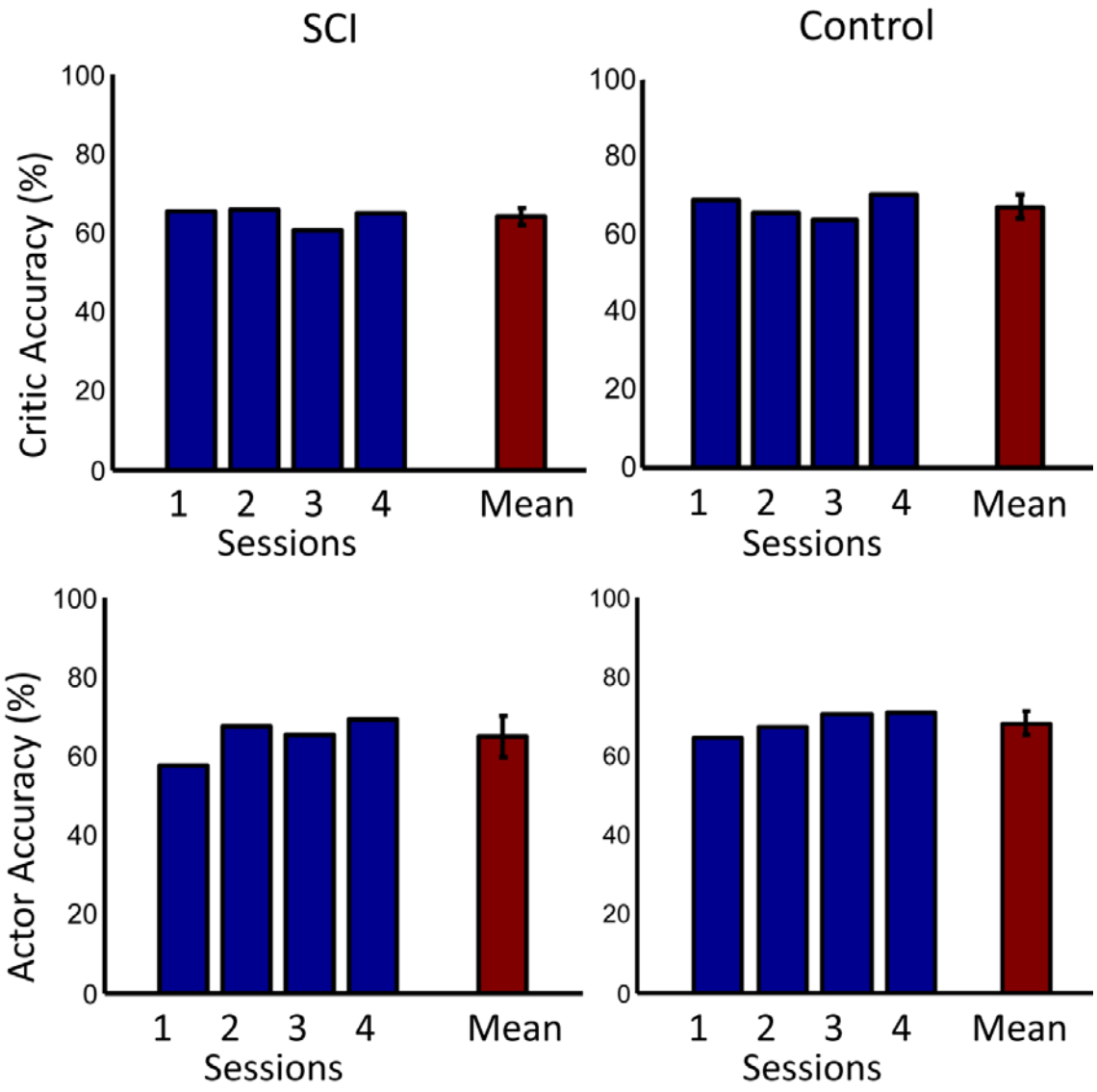

Figure 2.7 The first row shows the accuracy of the critic for both the SCI and control subjects. The second row shows the accuracy of the actor. Accuracy for each day is shown in blue. Mean accuracy across days is shown in red.

After a SCI, the brain experiences measurable maladaptive brain reorganization from disuse (Green, Sora et al. 1998; Cramer, Lastra et al. 2005; Hoffman and Field-Fote 2006; Kokotilo, Eng et al. 2009). These plastic changes can be partially reversed with rehabilitation techniques such as bimanual training and somatosensory stimulation (Hoffman and Field-Fote 2007; Hoffman and Field-Fote 2010). The motor cortex of chronic SCI subjects also experiences changes when they perform motor imagery training (Cramer, Orr et al. 2007). The ability to rehabilitate the motor cortex by motor imagery alone is important in the context of BCI augmented rehabilitation because motor imagery is often used to control BCIs. Notably, motor imagery has been used to control hand grasp FES in BCI systems (Pfurtscheller, Müller et al. 2003; Müller-Putz, Scherer et al. 2005). The combination of motor imagery and BCI controlled FES has been shown to rehabilitate finger extension in a stroke subject (Daly, Cheng et al. 2009). This improvement occurred with only 3 sessions a week over 3 weeks. By using an adaptive BCI, the subject could participate in rehabilitation over a longer period of time without needing to stop the rehabilitation to recalibrate the system. The proof-of-concept presented in this work also opens the possibility for the subjects to take the system home and use it continuously. This is due to not only the continuous RL that does not require calibration by a scientist but also to the design which uses the commercial Bioness H200 and an easy to use wireless Advanced Brain Monitoring EEG system.

In this study, ErrPs were collected from two users and were similar to published results (Figure 2.5) (Ferrez and Millan 2008). A classifier to detect the

ErrPs during the closed-loop sessions was created (Table 2.1). Feedback from the ErrP classifier was used to adapt the system to the user using RL (Figure 2.6). The system was able to classify both single trial ErrPs and motor potentials from features created from EEG recordings (Figures 2.3 and 2.4). The performance of the system improved over successive sessions until the performance reached the accuracy level of the ErrP classifier (Figure 2.7:row 2). Maintaining continuity in the performance over time is a critical aspect in the rehabilitation process. The user is able to pick up from the last level of progress achieved from the previous session.

Several additional results are also applicable to the use of the system during rehabilitation. The weights' values during later trials became stable, meaning the user would not experience sudden decreases in performance (Figure 2.6:row 2). The weights continued to adapt even in later trials, so the system can be expected to continue to adapt to the user in future trials, and during rehabilitation. The performance of the system increased above chance during the first day and continued to show improvement in later trials, both factors in maintaining user motivation and engagement (Figure 2.6:row 1). The ability of rapidly gaining control and maintaining it over time is an advancement over other approaches.

Chapter 3: FEATURE ADDITION

3.1 OVERVIEW

The experiments in chapter 2 suffered from unnecessarily low performance in early trials because of the number of inputs to the neural network. However, reducing the number of inputs could result in unnecessarily low performance in later trials. This dilemma of balancing the number of inputs and performance is known as the bias-variance tradeoff (Geman, Bienenstock et al. 1992). One possible solution to the bias-variance tradeoff in the case of RL and neural networks is to change the number of inputs over time. Changing the number of inputs could improve the performance and the experience for users in future experiments.

Bias-Variance Tradeoff

A BCI can be broken into several components: EEG recording, preprocessing, feature extraction, classifier, and the environment. The classifier is based on a model that maps modulation in its input features to the possible actions to be controlled in the environment. The model in the classifier is subject to a dilemma known as the bias-variance tradeoff (Geman, Bienenstock et al. 1992; Bishop 1995).

Prediction models, using known data to predict the class of new data, must manage the bias-variance tradeoff. An error associated with variance is caused by overfitting. In overfitting, the model is too closely based on known data. Predicting the class of new data causes a large error. The magnitude of the error caused by overfitting varies depending on how similar the new dataset is to the known dataset used to create the model. On the other hand, an error caused by bias is a result of underfitting the data. In underfitting, the model based on known data is too generic. While the error across new datasets has about the same magnitude, the error is unnecessarily large.

For models built using neural networks, the bias-variance tradeoff influences the decision on how large to make the neural network (Geman, Bienenstock et al. 1992). When neural networks are trained with supervised learning, a rule of thumb is the number of weights in the neural network is equal to half the number of trials used to train the network (Masters 1993). Networks with too many weights are prone to overfitting and error caused by variance. Conversely, networks with too few weights are prone to underfitting and error caused by bias. In supervised learning, the number of weights can be adjusted to fit the rule of thumb, the number of weights equals half the number of trials, because supervised learning uses a known number of trials to train the neural network. When using RL to train a neural network the number of trials is constantly changing. The data set size starts at zero trials and constantly increases with every trial.

When a classifier is trained on a fixed sample size, using too many features can lower the performance of the classifier. The decrease in performance happens most often when the sample size is small. So, the choice in the number of features is important, especially when using a small sample size (Hua, Xiong et al. 2005). Large neural networks have many degrees of freedom and can solve for many functions. However, large neural networks take a long time to train and require many examples before they generalize (Abu-Mostafa 1989; Śmieja 1993). Vapnik and Chervonenkis described the likelihood that a network given a set of examples would be able to generalize the information contained in the examples to classify future examples (Vapnik and Chervonenkis 1971). The ability of a network to generalize from a set of examples is expressed in the V-C dimension. The V-C dimension increases with network size (Baum and Haussler 1989). A large network with many weights trained on a small number of examples tends to overfit the data. While, a small network gives good generalization, if it converges. Taking into account these considerations, a network should be as small as possible, while still being large enough to converge (Zhang and Muhlenbein 1993).

The topology of networks is usually set before training begins. However, the topology of a network can greatly affect its performance. While, the performance of a network can only be evaluated after training. In offline training many different network topologies can be explored with trial and error and the best performing topology used (Hirose, Yamashita et al. 1991; Bartlett 1994; Weng and Khorasani 1996). However, this does not guarantee that the best

topology has been found. The best topology is not known at the beginning of training and might even vary during the learning task (Hirose, Yamashita et al. 1991; Bartlett 1994). By using an algorithm to design the topology of the network, time can be saved compared to a human using trial and error. Topologies designed by algorithms have been shown to provide comparable performance (Gruau, Whitley et al. 1996). A algorithm that adds new hidden processing elements can help a network escape a local minimum by changing the shape of the weight space (Hirose, Yamashita et al. 1991; Weng and Khorasani 1996).

Several different algorithms have been developed to adjust the topology of neural networks. The upstart algorithm tries to improve the performance of a network by adding processing elements. If the network makes a mistake, a processing element is added to recognize that pattern. The new processing element is connected to the output processing element, so the output is influenced when the pattern is presented again (Frean 1990). With evolutionary programming different network topologies can be tried. Topologies and weights can evolve with the best performing networks remaining after each iterations to be evolved further (Jian and Yugeng 1997). Instead of starting with many different topologies, topologies of a smaller networks can be used in the beginning and new topologies of larger networks can be added later to the mutations in the evolutionary program (Stanley and Miikkulainen 2002). Another approach called incremental evolution grows networks by adding processing elements instead of trying many different topologies and keeping and evolving the best performing networks (MacLeod and Maxwell 2001).

The approach of growing and adding processing elements does not depend on evolutionary programs and can be implemented independently. Dynamic Node Creation (DNC) starts with a small network and adds processing elements over time. The initial small network is trained until the performance of the network reaches a plateau. At this point another processing element is added. The network is trained again until the performance reaches another plateau and a processing element is added again. This process is repeated until the performance of the network reaches the desired level. By starting with a small network and adding processing elements over time, the initial small network is able to learn the gross parameters of the desired mapping and as new processing elements are added the finer parameters of the mapping are learned (Ash 1989). Similar algorithms have been developed: dynamic node architecture learning (DNAL), an algorithm that checks if the decrease in total error is below a threshold every 100 iterations, and a variation that uses a different update rule, a quasi-Newton based method (Hirose, Yamashita et al. 1991; Bartlett 1994; Setiono and Hui 1995). Another approach adds modules of small neural networks to the existing network (MacLeod, Maxwell et al. 2009) . Instead of adding processing elements, current processing elements in the network can be split into multiple processing elements. After the network has converged if the total error is above a threshold, one of the processing elements is split into several new processing elements. The processing element split is the one that shows the most fluctuations in its weights (Weng and Khorasani 1996).

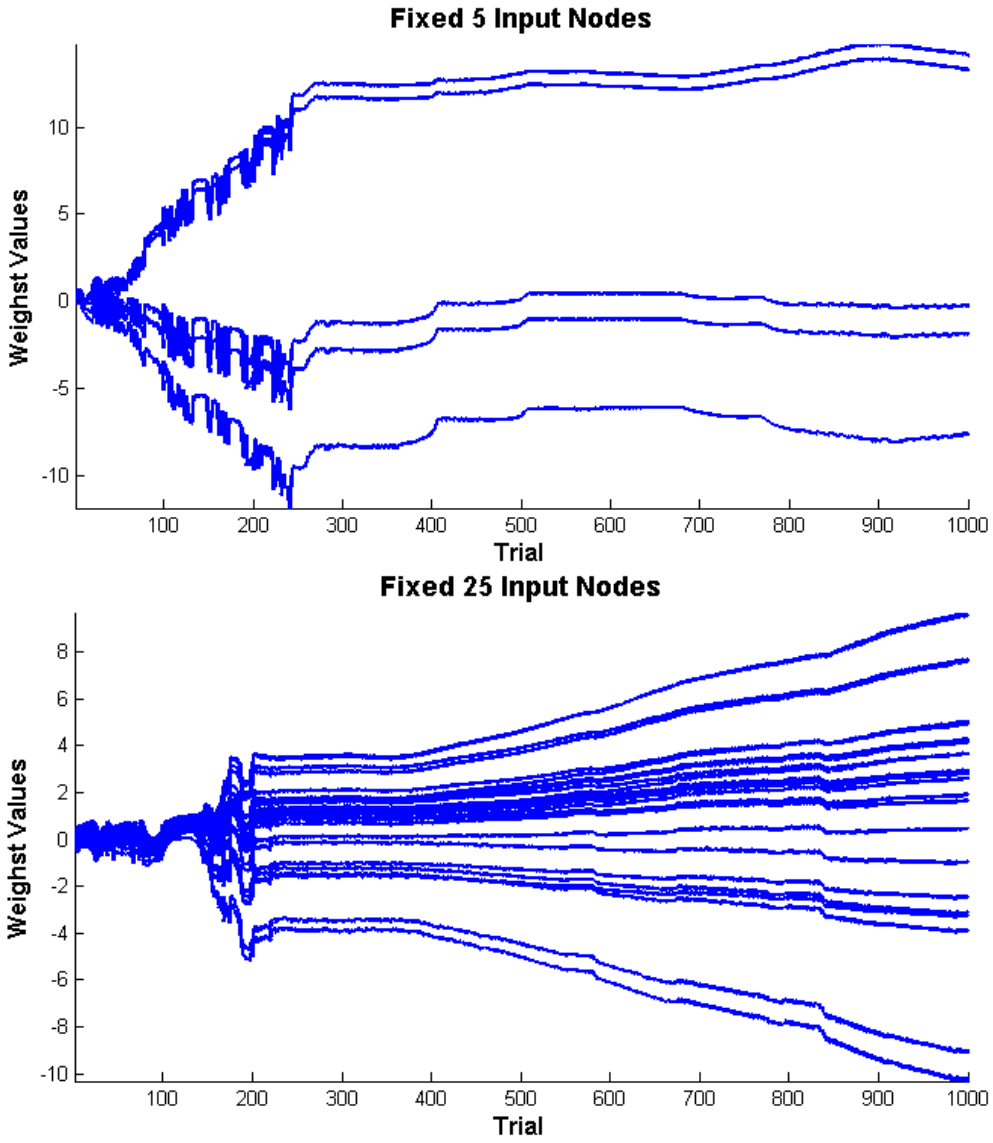

Figure 3.1 Weight values of two neural networks trained on the same data, a network with fixed 5 inputs and a network with fixed 25 inputs. The weights values of the 5 input network converge on a local minumum relatively early compared to the network with 25 inputs.

Network Size and Reinforcement Learning

A challenge with using RL with neural networks is choosing a size for the neural network to meet the design goals of adequate performance during early trials and undiminished performance during later trials. In the case of a BCI, the

performance during early trials should be high enough so the user does not become discouraged. Smaller neural networks could provide adequate performance during early trials by avoiding error produced by bias. However smaller neural networks might have diminished performance during later trials because of error caused by variance. Figure 3.1 shows the weight values, during training, of a smaller neural network with fixed 5 inputs. The weight values converge on a local minumum relatively early. However, with so few weights the network over-fits to early trials and experiences error during later trials from variance. In the bias-variance tradeoff, the small network is tilted to produce error from variance. Using a larger neural network could produce higher performance in later trials. Figure 3.1 also shows the weight values, during training, of a neural network with fixed 25 inputs. The weights adapt very slowly during the training. The larger network is underfitting during early trials causing error produced from bias.

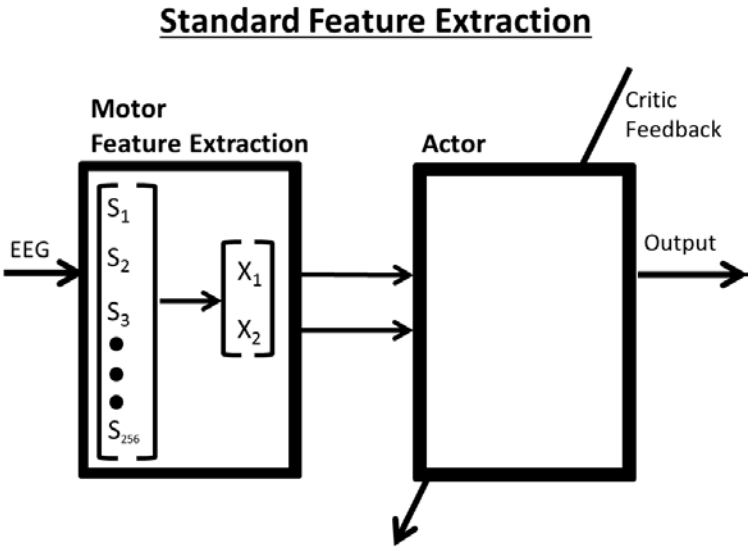

Figure 3.2 In standard feature extraction a fixed number of features are chosen before training and given as input to the classifier. The number of features is constant which can reduce performance because of the bias-variance tradeoff.

To minimize the problem of the bias-variance tradeoff one approach is to combine the ability of a small neural network to adapt quickly and perform well in early trials and the ability of a large network to perform well in later trials. A neural network that grew from a small number of inputs to many inputs would accomplish this goal. To change the number of inputs to a neural network will require a change in feature extraction. Figure 3.2 shows standard feature extraction, the number and type of features are chosen and all features are presented to the classifier, neural network, throughout training. Figure 3.3 shows an alternate approach for feature extraction, the number of features is chosen as before; however, only a subset of features is presented to the neural network at the beginning of training. Additional features are added over time until all the chosen features are presented to the neural network.

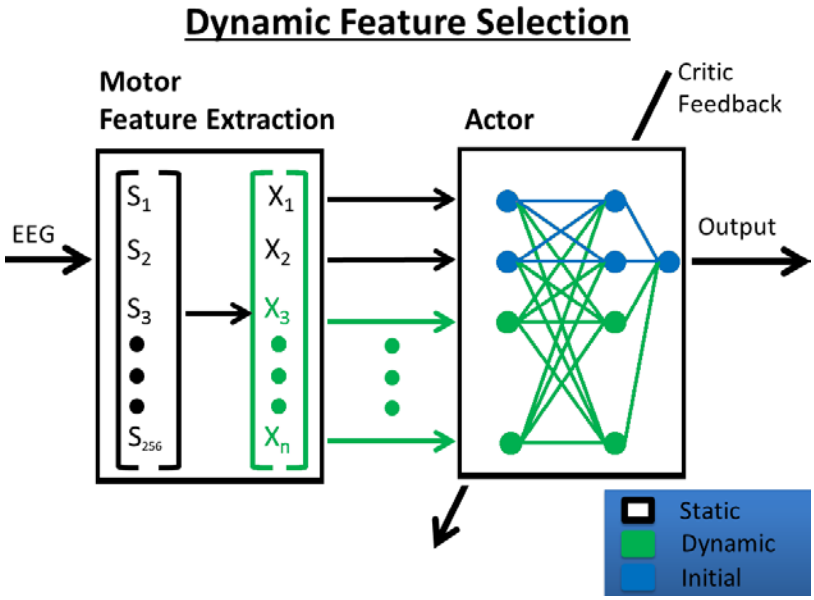

Figure 3.3 In dynamic feature selection, additional features are given as inputs to the classifier while the classifier is being trained. The ability to change the number of inputs can increase performance by ameliorating the bias-variance tradeoff.

Figure 3.4 shows how inputs and weights are added to the neural network. Processing elements and weights that are present throughout the session are shown in blue. Processing elements and weights that are added on a fixed schedule throughout the session are shown in green. As inputs and weights are added they adapt to the feedback given to the neural network. The values of new weights follow the same progression of other weights, adapting to feedback until they converge on a solution.

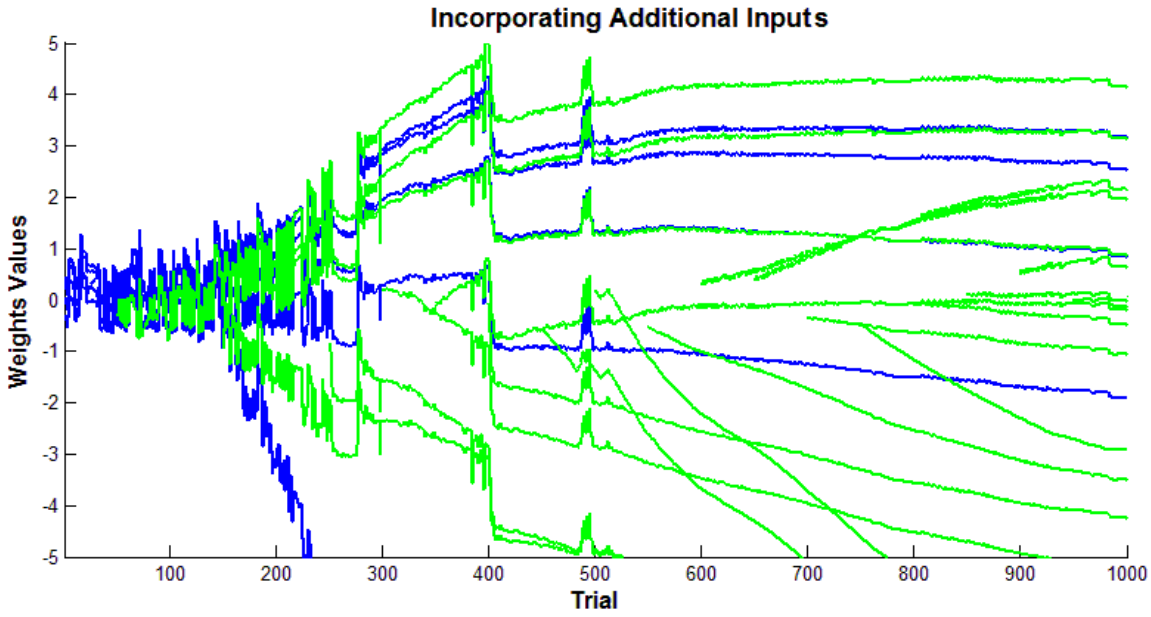

Figure 3.4 Weight values during training of a neural network with initially 5 inputs. An additional input is added after every 50 trials, until there are 25 inputs. The original 5 inputs are shown in blue and new inputs are shown in green. New inputs are able to be added, and the weights associated with the new inputs adapt to feedback in the same way as the original inputs.

The values of the initial weights present throughout the session adjust in a way similar to the weights in the smaller neural network. The initial weights adapt quickly during the beginning of the session until they converge on a solution. The values of weights added at later trials adjust in a way similar to weights in the large neural network. The new weights are associated with new inputs that provide new information the neural network uses to make a better classification of new trials. The new weights do not converge on a solution early in the session and some weights show large changes in later trials.

	Small Network	Large Network	Feature Addition
At 100 Trials	56.4% SD 5.5%	59.7% SD 11.7%	60.6% SD 9.9%
Session End	75.8% SD 16.8%	76.4% SD 12.4%	92.0% SD 4.3%

Table 3.1 Performance of simulations for a small and large network and a network using feature addition.

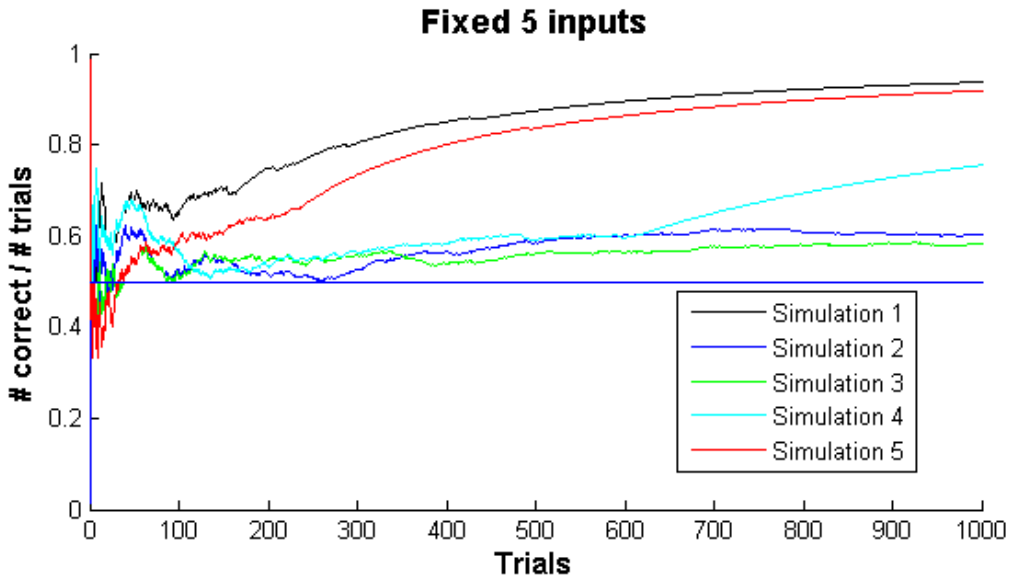

Figure 3.5 Performance of a neural network classifier with fixed 5 inputs during 5 Monte Carlo simulations of randomized initial weight values and trials order.

Dynamic feature addition improves performance over both the performance of the small and large neural networks. Figure 3.5 shows the performance of 5 Monte Carlo simulations for a small network and Figure 3.6 shows the performance of 5 Monte Carlo simulations for a large network. Both large and small networks had simulations that the classification accuracy did not increase across hundreds of trials and the end performance was less than 65%. The small network's simulations had a mean of 56.4% with SD of 5.5% at 100 trials and mean of 75.8% and SD of 16.8% at the end of the session. The large network's simulations had a mean of 59.7% with SD of 11.7% at 100 trials and mean of 76.4% and SD of 12.4% at the end of the session. The dynamic feature addition simulations had a mean of 60.6% with SD of 9.9% at 100 trials and mean of 92.0% and SD of 4.3% at the end of the session. The mean performance of dynamic feature addition simulations exceeded the mean performance of both the small and large network at both 100 trials and at the end of the session.

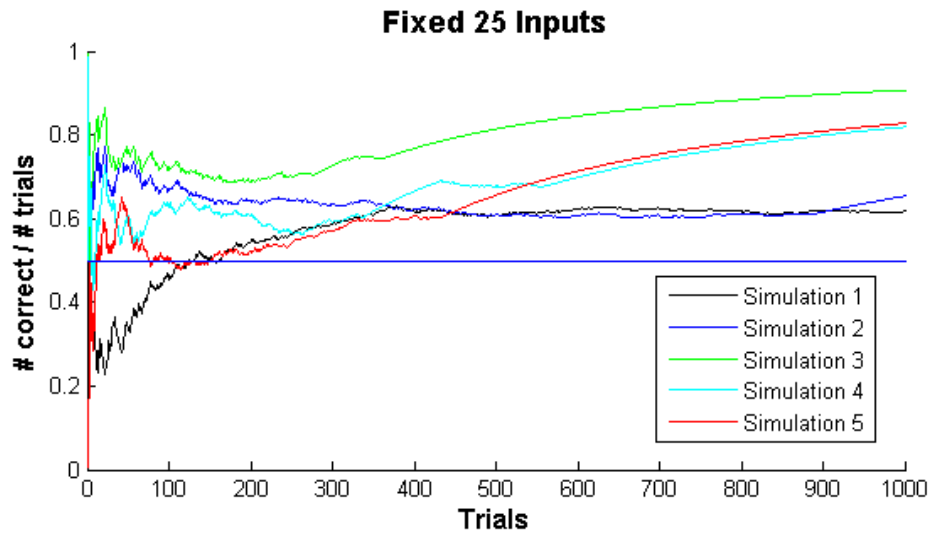

Figure 3.6 Performance of a neural network classifier with fixed 25 inputs during 5 Monte Carlo simulations of randomized initial weight values and trials order.

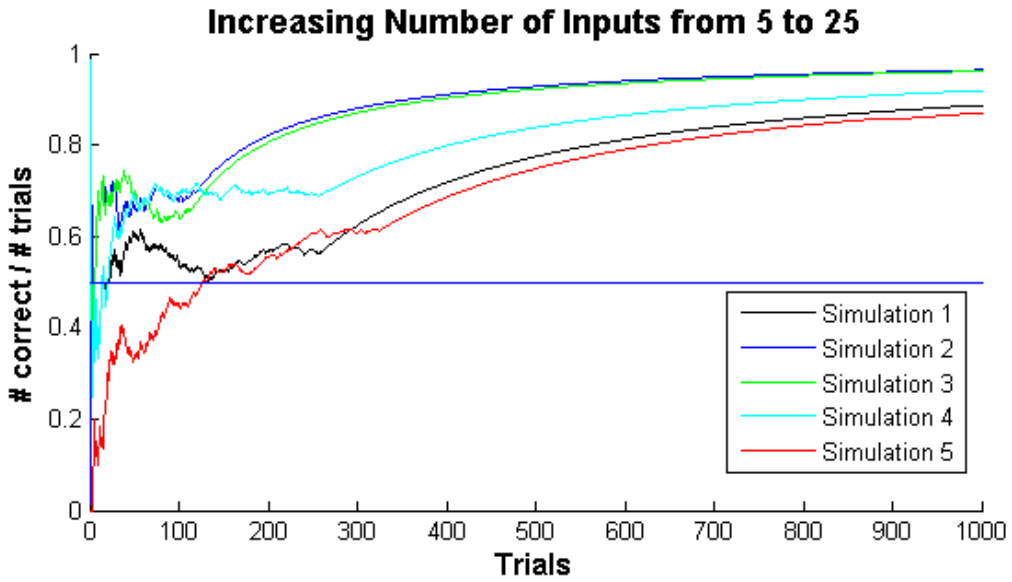

Figure 3.7 Performance of a neural network classifier using dynamic feature addition during 5 Monte Carlo simulations of randomized initial weight values and trials order. Initially 5 inputs with an input being added after every 50 trials until there are 25 inputs.

3.2 DISCUSSION

Choosing an appropriate size for a neural network when using RL for a BCI application is difficult because of the bias-variance tradeoff. A network too small will have unnecessarily poor performance during later trials and a network too large will have unnecessarily poor performance during early trials. By starting with a small network and using dynamic feature addition to grow the number of inputs to the network over time the performance of the BCI can be improved over both small and large networks in both early trials and later trials. Adding features can help the classifier escape local minimums in the error surface by increasing the dimension of the error surface. The addition of dynamic

feature addition to RL neural networks further improves RL BCIs beyond static BCIs.

Many different stopping conditions for adding features could be designed. With knowledge about what features could be useful, the number of features could be limited to what features may be useful. For example, features of power between 1-50 Hz have been shown to be useful in EEG. So, features could be limited to these frequencies. The number of features could be linked to performance with the number of features increasing until the performance reaches a plateau. If the desired performance has not been met and the computation time is too long for an individual trial, features could be pruned and other features added.

There are several limits to this approach. Incrementally adding features works best for a neural network being trained with an online algorithm. Offline training might be able to train the classifier more quickly by trying many combinations of features, instead of adding them incrementally. If the features are added too quickly, so the number of weights in the network increases more rapidly than the number of examples, the network could constantly under fit the data for each trial. Adding features only after enough trials have been recorded to train would fix this problem. If the number of possible features is small and all are known to be useful, adding them individually might not be helpful, especially if the algorithm is trained offline with a large data set.

To use this approach the initial size of the feature set should be small enough so the network can be trained on a small number of trials. Based on the

initial number of trials, and the quality of the features, the initial feature set size can be chosen. Features should only be added when enough samples exist to train them effectively and the features are relevant. Features that are not relevant should not be added. Adding a feature that does not help distinguish between the classes could slow the algorithm by increasing the number of weights.

Chapter 4: FEATURE SELECTION

4.1 OVERVIEW

Feature extraction involves the creation of features from measurements. Feature selection is a special case of feature extraction that involves the selection of certain features and measurements as inputs to the neural network (Verikas and Bacauskiene 2002). Only features that contribute significant information to the classification should be used. Features that contribute little information, irrelevant information, or are correlated or duplicates of other features should be excluded as inputs to the network (Brill, Brown et al. 1992; Priddy, Rogers et al. 1993; Belue and Bauer Jr 1995; Steppe and Bauer Jr 1996; Setiono and Liu 1997; Steppe and Bauer Jr 1997; Yang and Honavar 1998; Verikas and Bacauskiene 2002; Sivagaminathan and Ramakrishnan 2007; Aghdam, Ghasem-Aghaee et al. 2009; Kabir, Shahjahan et al. 2012). Measurements used to determine if a feature contains significant information are called saliency measures (Steppe and Bauer Jr 1997; Laine, Bauer et al. 2002; Kabir, Shahjahan et al. 2012). By using saliency measures, a subset of the many differnet measurements that exist and make up the measurement space can be chosen as the feature space (Ruck, Rogers et al. 1990; Brill, Brown et al. 1992; Setiono and Liu 1997; Yang and Honavar 1998; Pal, De et al. 2000; Sivagaminathan and Ramakrishnan 2007; Aghdam, Ghasem-Aghaee et al. 2009).

The different features chosen as inputs to the neural network can be organized into a vector (Ruck, Rogers et al. 1990; Priddy, Rogers et al. 1993; Yang and Honavar 1998). Since many features contain irrelevant information or contain information contained in other features, the dimensionality of the feature vector can be reduced with little loss of information (Setiono and Liu 1997; Pal, De et al. 2000; Verikas and Bacauskiene 2002; Aghdam, Ghasem-Aghaee et al. 2009). By decreasing the number of features, fewer training examples are needed. This is the inverse of “the curse of dimensionality” where increasing the number of features means the number of training examples must also be increased (Yang and Honavar 1998; Verikas and Bacauskiene 2002). Using fewer features can also improve the accuracy of the neural network. By not including features composed of irrelevant or redundant information, the network does not have to minimize the effects these features have on the output of the network (Ruck, Rogers et al. 1990; Brill, Brown et al. 1992; Battiti 1994; Setiono and Liu 1997; Yang and Honavar 1998; Bauer Jr, Alsing et al. 2000; Laine, Bauer et al. 2002; Verikas and Bacauskiene 2002; Sivagaminathan and Ramakrishnan 2007; Aghdam, Ghasem-Aghaee et al. 2009; Kabir, Shahjahan et al. 2012). Reducing the number of features also reduces the time needed to train the network, create features, and classify a feature vector. While time considerations might not be very important for offline applications, they can be critical for online applications (Ruck, Rogers et al. 1990; Battiti 1994; Setiono and Liu 1997; Yang and Honavar 1998; Laine, Bauer et al. 2002; Sivagaminathan and Ramakrishnan 2007). Depending on the application the cost of collecting data for additional

features could be high both financially and because of increased risk (Setiono and Liu 1997; Yang and Honavar 1998; Verikas and Bacauskiene 2002; Sivagaminathan and Ramakrishnan 2007). All these reasons support the careful selection of features. When humans select features they introduce their own biases of what will be useful features. The process of selecting features can be automated which could improve accuracy by removing human biases and increase the speed of feature selection (Brill, Brown et al. 1992).

Several approaches for finding useful features rely on creating a trained neural network from all the available features. One approach varies each feature over its range and computes the changes in the error of the output. Features that have little effect on the error when varied can be removed (Priddy, Rogers et al. 1993). Another approach judges the sensitivity of the network's output to the various features by examining the network's weights. The feature that affects the output the least can be removed (Ruck, Rogers et al. 1990).

A different approach is to add a noise feature to the available features and using the same procedure of inspecting weights to compute the features' effect on the trained network's output, a signal to noise ratio can be computed for each feature. Features that do not affect the output much more than the noise feature are removed (Bauer Jr, Alsing et al. 2000). By using the signal to noise ratio technique the features relevant to mental workload measured by EEG were determined to be power in the 31-40 Hz range (Laine, Bauer et al. 2002). Instead of using a signal to noise ratio comparison, a confidence interval can be constructed for the noise feature. Features that lie outside the confidence interval

can be removed (Belue and Bauer Jr 1995). Another variation is to use a Bonferroni-type test statistic, features that are less useful compared to the noise feature can be excluded (Steppe and Bauer Jr 1996).

Another way to find which features are most useful is to use a neural network and a pruning algorithm. The network is trained on all possible features, with a special penalty term included to make small weights even smaller. The features are ranked by performance of the network when the feature is set to zero. The feature that causes the largest increase in performance when set to zero is removed. The whole process is repeated until the performance of the network degrades beyond a threshold (Setiono and Liu 1997). Forcing small weight to become even smaller makes the output less sensitive to changes in the input, which makes finding useful features more difficult. To solve this problem, the algorithm can be modified by changing the transfer function to force processing elements to work in the saturation regions (Verikas and Bacauskiene 2002).

Instead of pruning features, features can be added to the feature set. Offline testing can be done on the different features using the “leave-one-out” method, N-1 samples are used to train the classifier and the remaining sample is used to test the classifier. The feature that shows the greatest classification accuracy is added to the previous features already used (Whitney 1971). An extension of the this algorithm is the ability to add and remove features called “floating search” instead of monotonically adding features (Pudil, Novovičová et al. 1994). The degree to which sets of features do not improve classification

accuracy over subsets of those features is mutual information. Mutual information can be used to determine which features are useful (Battiti 1994).

Another approach uses a genetic algorithm to find useful features. Different sets of features are ranked based on the accuracy of a neural network trained on each set of features. Poor performing sets are eliminated, remaining sets are mixed to have some of the features of other sets, and random mutations of the addition or removal of a feature occur. This process is repeated several times, with the best performing set used at the end (Brill, Brown et al. 1992; Yang and Honavar 1998).

Another type of algorithm that can be used is Ant Colony Optimization (ACO). An ACO algorithm finds the best path through a graph by simulating ants searching for the shortest path to food sources. By organizing the possible sets of features into a graph, with each node representing the addition of a feature, an ACO algorithm can be used to find the best feature set for classification. When the ACO algorithm visits each node of the graph the feature set represented with the node is evaluated by training a neural network and computing its accuracy. The ACO algorithm learns the path between feature sets that have high accuracy to find the set with the highest accuracy (Sivagaminathan and Ramakrishnan 2007). This approach has been used to find features for text classification (Aghdam, Ghasem-Aghaee et al. 2009). Instead of only using the classification accuracy of the network other measures can also be used to judge the feature sets, such as mutual information (Al-Ani 2006; Kabir, Shahjahan et al. 2012).

4.2 FEATURE SELECTION IN DYNAMIC FEATURE ADDITION

The dynamic feature addition in chapter 3 added features in a predetermined order. Features were added according to their frequency from 1 to 50 Hz. Adding features in a different order might improve the performance of the dynamic feature addition.

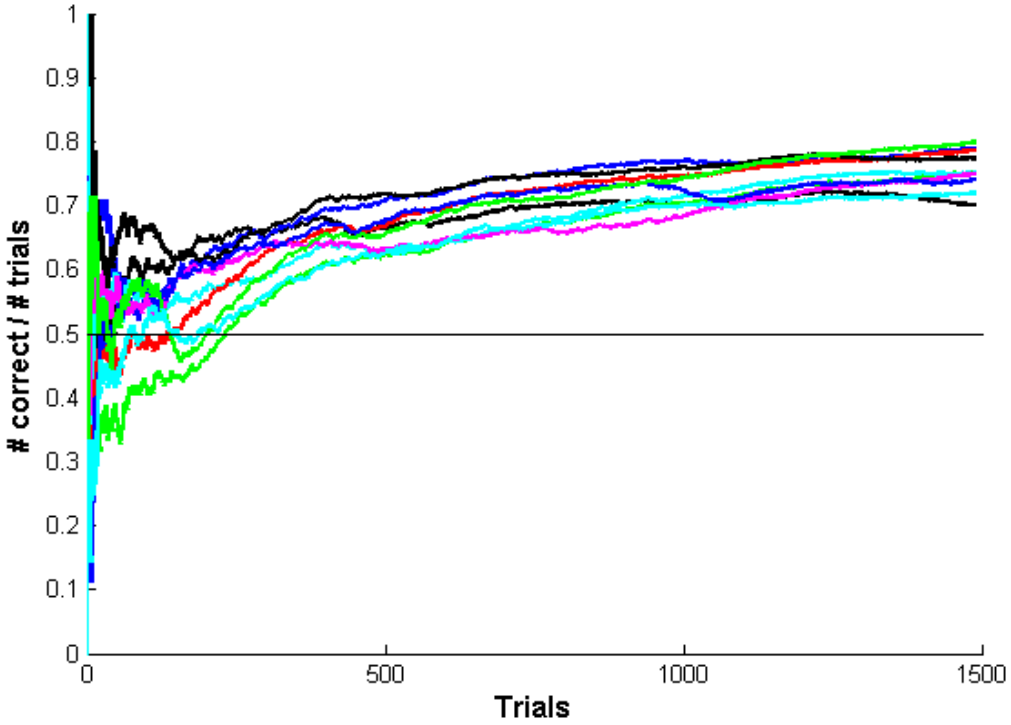

Figure 4.1 The performance of a network starting with 5 inputs (1-5 Hz) and adding 5 inputs every 150 trials in order of frequency bin, 5 to 50 Hz, during ten Monte Carlo simulations of randomized initial weight values and trials order. The change in number of inputs ameliorates the bias-variance tradeoff; however, adding features in this order reduces the potential performance of the network.

Using the data from the SCI subject in chapter 2, Figure 4.1 shows the performance of a network starting with 5 inputs (1-5 Hz) and adding 5 inputs

every 150 trials in order of frequency bin, 5 to 50 Hz, during ten Monte Carlo simulations of randomized initial weight values and trials order. The order in which features are added in Figure 4.1 does not take into account differences between features. These differences could be used to improve the order in which features are added.

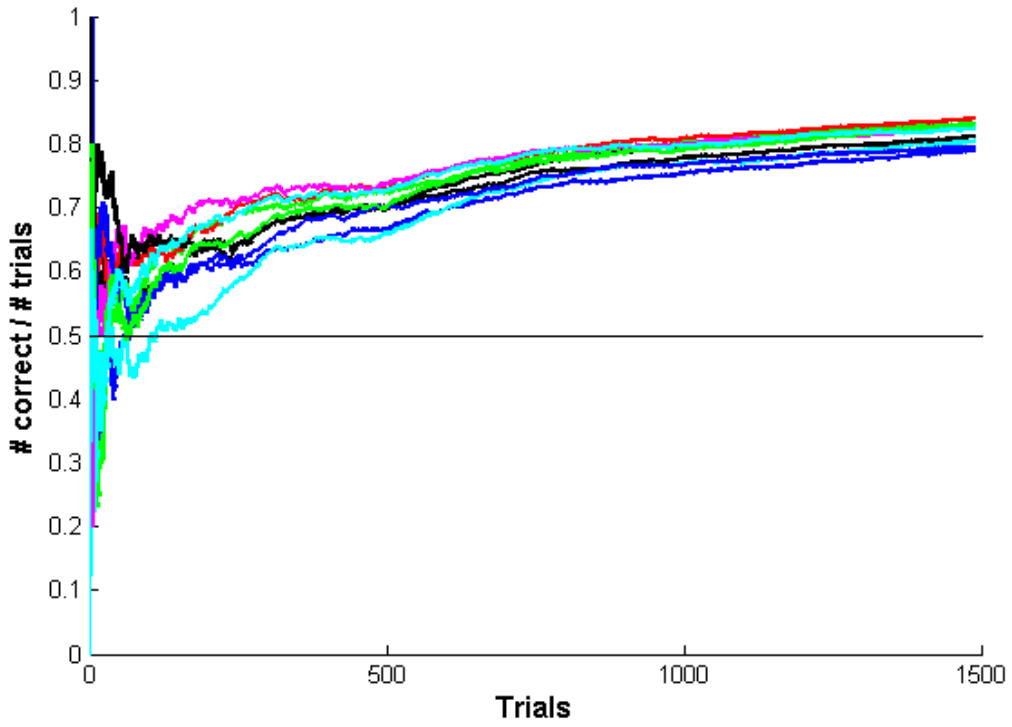

Figure 4.2 The performance of a network starting with 5 inputs and adding 5 inputs every 150 trials in order of largest difference in features between cues during ten Monte Carlo simulations of randomized initial weight values and trials order. The change in number of inputs ameliorates the bias-variance tradeoff; and, adding features in this order improves the performance of the network compared to other addition orders.

One way to measure the difference between features is to use z-scores to quantify the relative difference in average power of frequencies between cues during sessions. Large z-scores correspond to large differences between cues meaning those frequencies will be more useful for classification than other

frequencies. By using information about the relative usefulness of features, the order in which features are added can be improved. Figure 4.2 shows simulations where features added during a session showed the greatest difference between cues in the previous session. The features that showed the greatest difference between cues in the preliminary recording session were used as the initial features set. After 150 trials in the closed-loop sessions data, 5 additional features were added that showed the greatest difference between cues for the last 150 trials. This process was repeated every 150 trials until all the features from 1 to 50 Hz were added.

Table 4.1 shows that adding features to try to maximize the difference between cues in early trials results in simulations with higher end accuracy, mean performance of 81.76%, and smaller variations between simulations, SD of 1.6%.

Simulation	Largest Difference First	Smallest Difference First	By Frequency	Static Feature Set
1	84.03%	71.07%	79.93%	87.85%
2	83.22%	70.00%	78.93%	86.17%
3	82.89%	69.87%	78.72%	85.17%
4	82.81%	69.26%	77.38%	83.09%
5	82.40%	68.52%	75.10%	79.06%
6	82.31%	68.46%	75.03%	77.58%
7	81.14%	68.12%	74.90%	76.51%
8	80.40%	68.05%	74.09%	76.50%
9	79.46%	66.91%	71.95%	76.11%
10	78.98%	64.70%	70.07%	72.21%
Mean	81.76%	68.50%	75.61%	80.03%
SD	1.60%	1.69%	3.00%	4.93%

Table 4.1 Comparison of sorted performance of simulations from different feature addition orders and static feature set. Using the data from the SCI subject in chapter 2, the performance of a network was simulated during ten Monte Carlo simulations of randomized initial weight values and trials order. Adding features with the largest difference between cues first produces simulations with the highest mean performance and smallest SD between simulations.

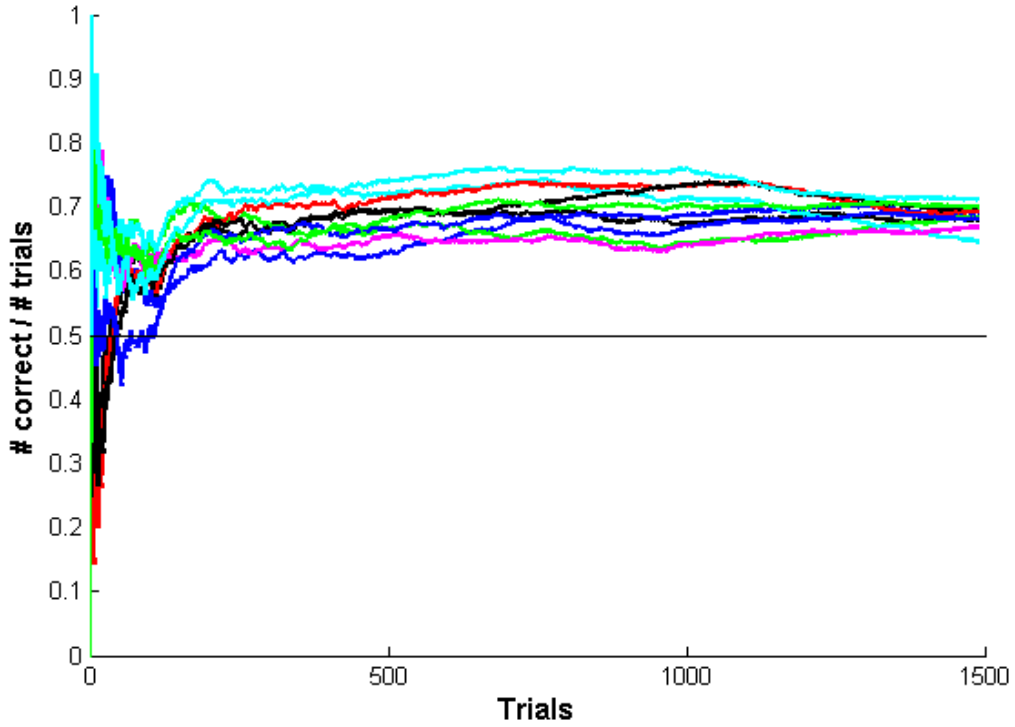

Figure 4.3 The performance of a network starting with 5 inputs and adding 5 inputs every 150 trials in order of smallest difference in features between cues during ten Monte Carlo simulations of randomized initial weight values and trials order. The change in number of inputs ameliorates the bias-variance tradeoff. However, this counterexample again shows the importance of addition order as adding features in this order results in the lowest performance of all the examples.

If the features are added in a sequence to try to minimize the difference between cues in early trials, the performance goes down compared to all other techniques. Features that provide the most information to differentiate between the two cues are added in later trials, too late to help develop a classifier. Figure 4.3 shows the performance of ten Monte Carlo simulations where the initial five features where the five 1 Hz frequency bins that showed the smallest difference between cues in the preliminary recording session. After the first 150 trials of the closed-loop sessions data, the next five cues added were the 1 Hz frequency bins that showed the smallest difference between cues in the previous 150 trials.

Features were added every 150 trials using the same procedure until all features from 1 to 50 Hz were included. The simulations using this method had lower performance at the end of the session than all other methods, again showing the importance of the order in which features are added.

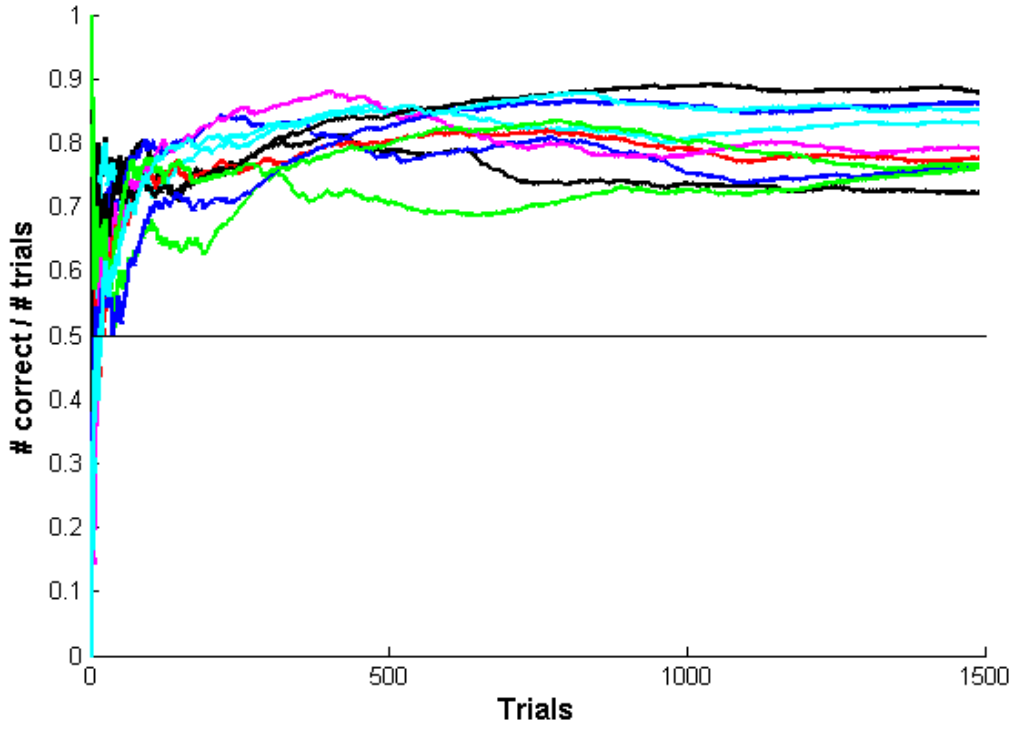

Figure 4.4 The performance of a network with a constant number of inputs, 50, during ten Monte Carlo simulations of randomized initial weight values and trials order. The constant number of inputs makes the network susceptible to the bias-variance tradeoff.

4.3 DISCUSSION

The order in which features are added during dynamic feature addition can affect the performance of the system. The simple method of adding features

by frequency order reduces the potential performance of the system. Adding features that behave similarly for each class does not help the classification as much as adding features that have distinct behaviors for each class. By taking into account how useful features are for discriminating between different cues and adding features that are more useful in early trials, the performance of the system can be improved. The features that contribute the most during classification would be added first so the weights could be adapted to these features first. Features that provided smaller clues to the class of the trial would be added later and fine-tune the weights. This approach should generalize to all BCIs and all classification problems that use a neural network during online learning. Major factors that affect this result are variability in noise that affect different features differently, how well the available features differentiate between the classes, the ability of the algorithm to determine which features have good differentiation, and the pace at which features are added.

Chapter 5: VARIABLE LEARNING RATE

5.1 OVERVIEW

The goal of the rehabilitation system is to use RL in a BCI so that the BCI adapts to the user over the course of rehabilitation. Using a static learning rate in the BCI could reduce the ability of the BCI to adapt to the user. A learning rate that adjusts to facilitate the progression of the user during rehabilitation could lead to a better rehabilitation system.

The RL algorithm tries to find the minimum of the error surface. The error surface of a neural network describes the relationship between the network's weights and error at the network's output. The properties of the error surface for real world applications have been studied by several authors (Widrow and Lehr 1990; Hush, Horne et al. 1992; Yu 1992; Hush and Horne 1993). The error surfaces of neural networks are composed of large regions with small gradients and long narrow regions with steep gradients. A weights update algorithm that uses a fixed learning rate will be inefficient because of these characteristics of neural networks' error surfaces. An algorithm using a large learning rate will progress rapidly over the region with a small gradient; however, steep gradients will cause the algorithm to over adjust and produce oscillations in the weights. A small learning rate will make the algorithm more stable in regions with steepgradients; however, the algorithm will move slowly through regions with small gradients (Gori and Tesi 1992). Algorithms with a constant learning rate are

not guaranteed to converge in general (Kuan and Hornik 1991). To alleviate these problems an adaptive learning rate can be used instead of a constant learning rate. By using an adaptive learning rate, the probability for poor performance because of a poor choice for an initial learning rate are reduced (Yu and Chen 1997). An adaptive learning rate reduces the need for batch learning and improves first-time learning, a important consideration for online learning algorithms (Weir 1991). Many different approaches to adjust the learning rate have been developed. One approach is to compare the error for learning rates slightly higher and slightly lower than the current learning rate, and choose the learning rate that produces the lower error (Solomon and Leo van Hemmen 1996). Another approach replaces the normal cost function used in back propagation with the Lyapunov function; with this change, the back propagation algorithm converges faster. By substituting in the Lyapunov function, the algorithm has the same behavior as back propagation with an adaptive learning rate (Behera, Kumar et al. 2006). If the gradient of the error surface is known the learning rate can be adjusted appropriately. By using the forward and backward procedure during back propagation the derivative of the error surface can be calculated and the learning rate adjusted accordingly (Yu, Chen et al. 1995; Yu and Chen 1997). Estimates of the gradient can also be used to adjust the learning rate. The gradient can be estimated using the direction cosines of the steepest descent vector (Hsin, Li et al. 1995), secant equation based quasi-Newton method (Barzilai and Borwein 1988; Polak 1997; Plagianakos, Sotiropoulos et al. 1998), or by estimating the Lipschitz constant, the steepest

gradient of a function (Magoulas, Vrahatis et al. 1997). Another approach uses a combination of the height of the error surface, the current error, and maximum gradient to compute a step length of the quotient height over maximum gradient (Weir 1991). These approaches do not apply to online RL with a neural network for several reasons: the computation time required to compare different learning rates and the dynamic nature of the error surface between trials. During early trials, the error surface is especially likely to fluctuate because of the small size of the sample set. A different approach uses the height of the error surface alone to increase or decrease the learning rate by a fixed factor (Jacobs 1988). The advantages of this approach are the simplicity of measurement and computation. The height of the error surface is also less susceptible to the dynamic nature of the error surface than the slope of the error surface.

5.2 METHODS

To test the rehabilitation system's ability to adapt several different update rules were investigated. The traditional method to update the weights of a multilayer perceptron (MLP) is with back propagation. Back propagation can be broken down into several steps. Compute the output of the MLP:

$$r_h = \varphi\,(w_h * r_i)$$

$$r_o = \varphi\,(w_o * r_h)$$

where r_i is the input to the MLP, r_h is the output of the hidden layer, r_o is the output of the MLP, φ is the activation function, w_h are the weights of the hidden layer, and w_o are the weights of the output layer. Compute the error of the MLP:

$$d_o = \left(r_0(1 - r_0)\right) * \ (r_d - r_0)$$

$$d_h = \left(r_h(1 - r_h)\right) * \ (w_o * d_0)$$

where r_d is the desired output of the MLP, d_o is the error at the output layer, and d_h is the error at the hidden layer. Then update the weights of the MLP:

$$\Delta w_h = \gamma(r_i * d_h)$$

$$\Delta w_o = \gamma(r_h * d_o)$$

Where Δw_h is the change of the weights in the hidden layer, Δw_o is the change of the weights in the output layer, and γ is the learning rate. While back propagation is the most common method, other approaches exist. Another update method is based on Hebbian style learning (Mazzoni, Andersen et al. 1991). Hebbian style learning can be written mathematically as:

$$\Delta w_{ij} = \begin{cases} \gamma(x_i\left(p_j - \ x_j\right)), \ f_t = 1 \\ -\gamma\left(x_i\left(p_j - \ x_j\right)\right), \ f_t = 0 \end{cases}$$

Where Δw_{ij} is the change of the weight between processing elements i and j, γ is the learning rate, x_i is the input to processing element j from processing element i, p_j is the thresholded output of processing element j, x_j is the output of processing element j, and f_t is the current feedback of the critic. Δw_{ij} is a function of the multiplication of the input and the output of processing element j, which is Hebbian style learning. Depending on the current feedback of the critic the weight w_{ij} either increases for positive feedback, 1, or decreases for negative feedback, 0.

The change in weight values can also be scaled as a function of how the system is performing. A more biologically plausible way to think of variable weight value change is that the weight change scales to how often feedback happens. For example, if errors are frequent, negative feedback will not create a large weight change but positive feedback will create a large weight change. By using this approach when the system is performing well and errors are few the system will not make dramatic changes when the output is correct, preserving the classification ability of the system. However, when an error occurs the system will make large changes to learn from that trial and error. Conversely when the system is performing poorly and errors are common, the system will make large changes for a correct output to learn from the trial and output. A way to implement this mathematically is:

$$r = (\sum_{i=1}^{t} f_i)/\#f$$

Where r is the scale factor, f_i is the feedback at trial i, and $\#f$ is the number of trials. The scale factor r can be incorporated into the Hebbian style learning as:

$$\Delta w_{ij} = \begin{cases} (1-r) * \gamma(x_i\,(p_j - x_j)), & f_t = 1 \\ (r) * -\gamma(x_i\,(p_j - x_j)), & f_t = 0 \end{cases}$$

The scale factor r can also be incorporated into back propagation as:

$$\Delta w_h = (1-r) * \gamma(r_i * d_h)$$

$$\Delta w_o = (1-r) * \gamma(r_h * d_o)$$

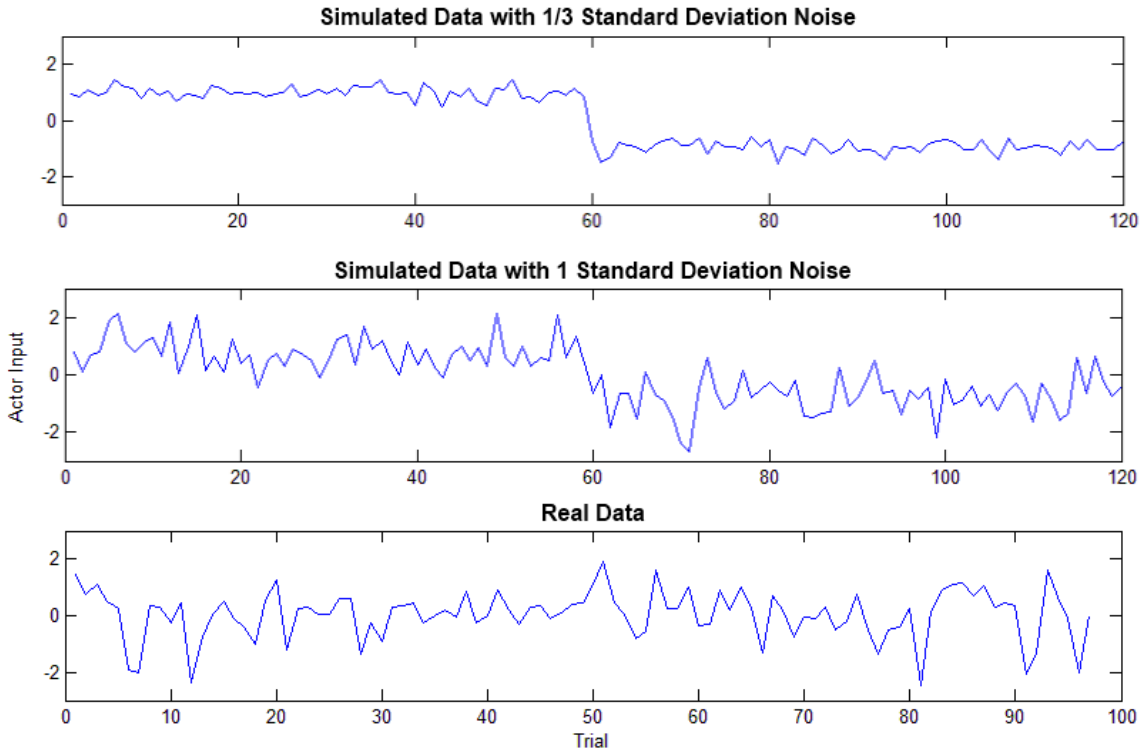

Figure 5.1 One input given to a neural network during testing of the different update rules: 1/3 standard deviation noise test data, 1 standard deviation noise test data, and real data.

Since the data from real recordings has a large amount of noise, comparing the performance of the different update rules with real data would be challenging. To test the various update rules, test data sets were created. The SD of a real recording was used to estimate the amount of noise to include in the test data sets. The first test data set has Gaussian noise with a SD equal to 1/3 the SD of the real recording. The second test data set has Gaussian noise with a SD equal to the SD of the real recording. Figure 5.1 shows the two test data sets created next to a recording from subject 1.

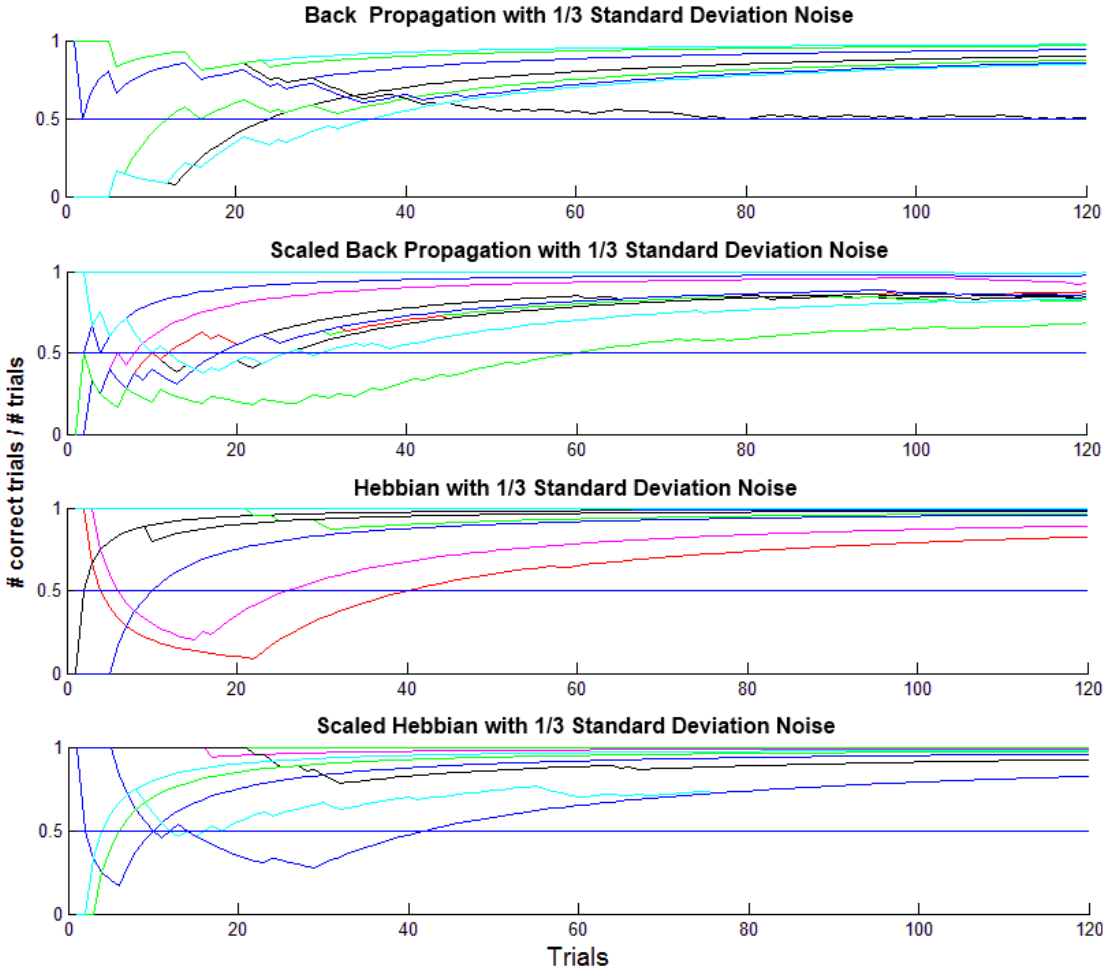

Figure 5.2 Comparison of a neural network's performance using back propagation, scaled back propagation, Hebbian style learning, and scaled Hebbian style learning using 1/3 standard deviation noise test data and 70% critic accuracy. Performance of the different update rules is comparable.

The accuracy of the critic was set at 70% to approximate the classification accuracy of the critic on real error data. Ten simulations were performed with each update rule. Performance was judged by cumulative classification accuracy of the actor on motor data, with chance equal to 50% for this two class test.

The results of using a 1/3 SD noise test data set with the various update rules is shown in Figure 5.2. Overall the results between the various update rules are very similar. The signal to noise ratio is so high that the update rule becomes less important. Occasionally back propagation will not be able to advance beyond a local minimum, which can be seen in the first graph as the simulation marked in black approaches 50% classification accuracy. Hebbian style learning does not seem to have this problem.

The increased noise in the 1 SD noise test data set starts to show the difference between the update rules, as seen in Figure 5.3. To quantify these results the update rules were compared by the number of simulations to reach 60% classification accuracy at the end of 120 trials, table 5.1. Hebbian style learning had better overall performance compared to back propagation and performed better than chance earlier. Similarly, scaling the back propagation or scaling the Hebbian style learning increased overall performance and helped the system reach a performance better than chance faster. The results of using scaled Hebbian style learning with real data are shown in Figure 5.4. The real data results are similar to the 1 SD noise test data set.

Simulations Above 60% Classification Accuracy	
Update Method	**Simulations (10 Total)**
Back Propagation	3
Scaled Back Propagation	8
Hebbian Style Learning	5
Scaled Hebbian Style Learning	10

Table 5.1 Number of simulations above 60% classification accuracy for each update rule. Scaled Hebbian Style Learning outperforms the other update rules.

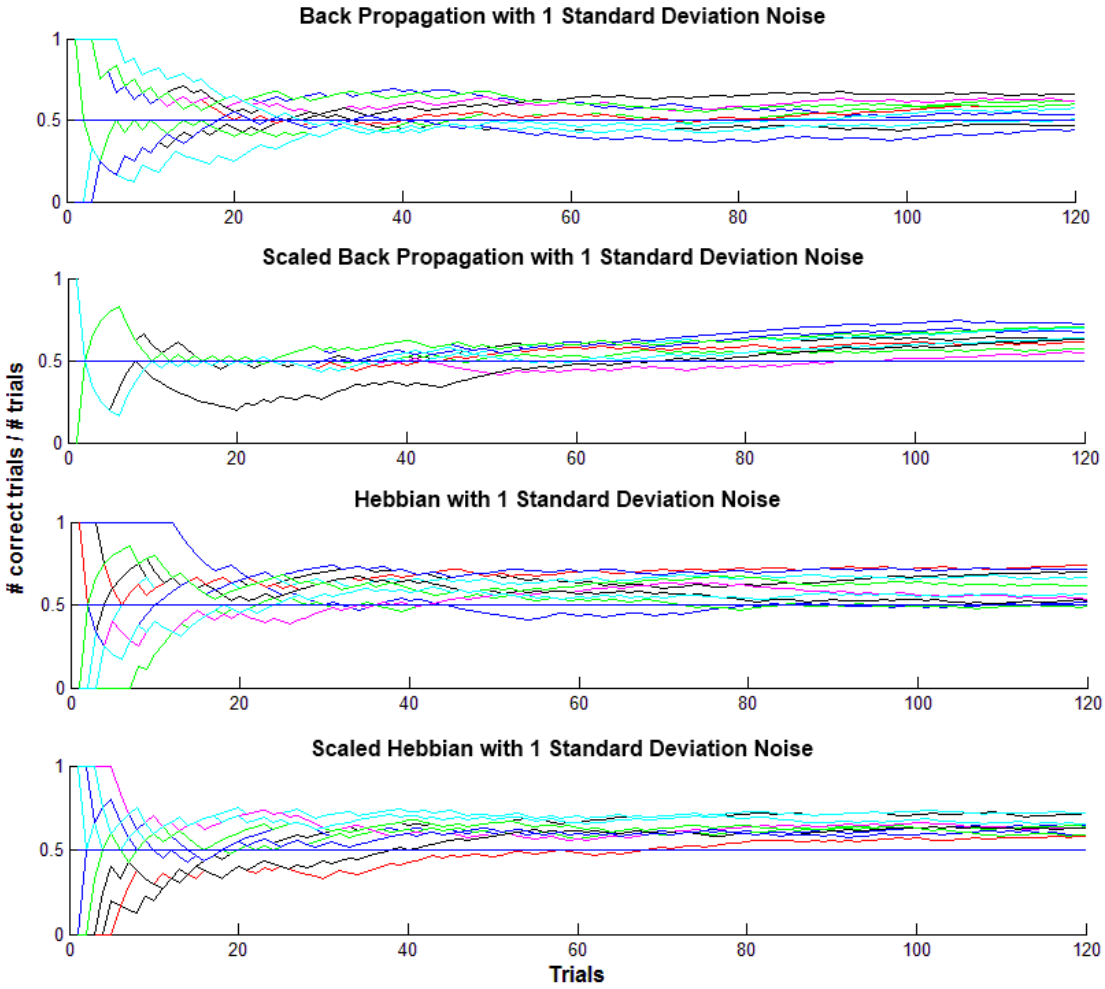

Figure 5.3 Comparison of a neural network's performance using back propagation, scaled back propagation, Hebbian style learning, and scaled Hebbian style learning using one standard deviation noise test data and 70% critic accuracy. The network performed the best using scaled Hebbian style learning.

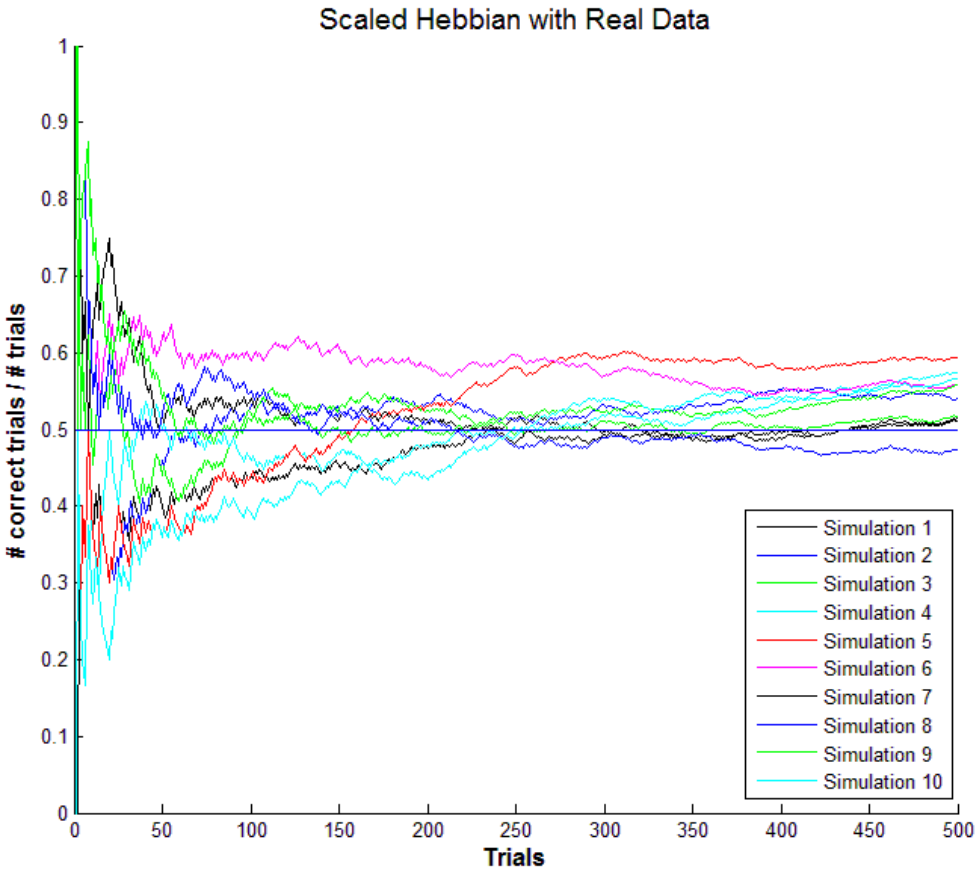

Figure 5.4 Using real data as input, the actor's performance when critic is 70% accurate with scaled Hebbian style learning update.

All the techniques of the current and previous chapters can be combined to improve the results even further. The scaled Hebbian style learning update can be combined with dynamic feature addition and adding features based on the largest difference between cues. Figure 5.5 shows the results of combining the techniques for ten Monte Carlo simulations using the SCI subject data from chapter 2. The simulations of the combined techniques performed better than the

other simulations. These results show the techniques can be used together and the improvements from the techniques are additive when combined.

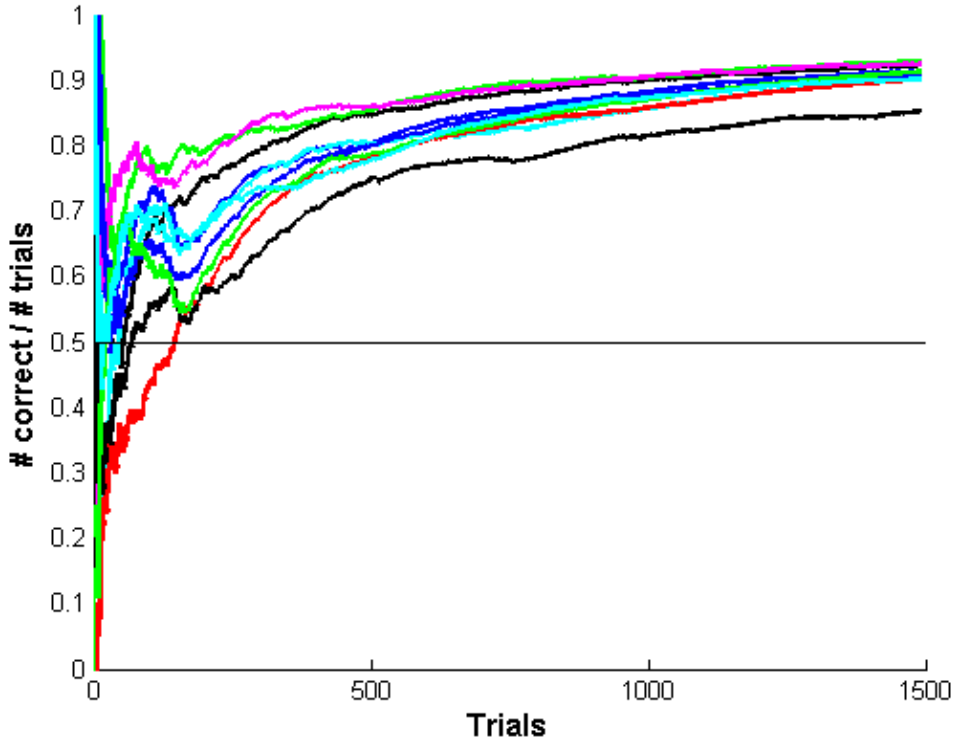

Figure 5.5 Using the combined techniques (scaled Hebbian style learning update, dynamic feature addition, and adding features based on the largest difference between cues) and the SCI subject data as input, the actor's performance during ten Monte Carlo simulations of randomized initial weight values and trials order.

5.3 DISCUSSION

Various update rules could be used in the rehabilitation system: back propagation, scaled back propagation, Hebbian style learning, and scaled Hebbian style learning. In simulations Hebbian style learning performed better than back propagation. While scaled Hebbian style learning performed better

than Hebbian style learning. Scaled Hebbian style learning also takes advantage of the online nature of RL used in the system. By adjusting the learning rate, the algorithm moves more quickly in areas where the slope of the error surface is small and converges on a minimum more quickly in areas where the slope is high. The adjustable learning rate also makes the choice of learning rate at the beginning of the algorithm have less of a negative effect on the performance of the algorithm. This approach should generalize to all classification problems using a neural network with an online learning algorithm.

A weakness of this approach is that it only relies on the height of the error surface. The slope of the error surface would give a more direct measure of what the learning rate should be set at, however determining the slope of the error surface is more computationally expensive and the slope is more likely to change between trials than the height. Even though the learning rate is adjustable the range of possible learning rates and size of the change in learning rate have to be set at the beginning of the experiment. Given the range of the learning rates and the size of the adjustments are preset, the learning rate cannot always be ideal for the error surface. Other factors that affect this result are how much noise is in the recordings and the dimensionality of the error surface.

This approach is best suited for classification problems using neural networks in an online learning algorithm. The range of learning rates should be chosen to accommodate reasonable maximum and minimum slopes of the error surface. The size of the adjustments to the learning rate can be smaller when the range between the maximum and minimum slope of the error surface is smaller

and the size of the adjustments to the learning rate can also be smaller when more trials are available.

Chapter 6: CONCLUSION

The research presented in this dissertation demonstrated a new EEG based BCI system using RL. A classifier was created for each user to detect ErrPs during closed-loop sessions. By using RL and detecting the user's ErrPs, the system was able to adapt to the user throughout the experiment without offline training. Over the course of the experiment, the classification accuracy for motor potentials increased each session until the accuracy reached the performance level of the ErrP classifier. The system was able to find motor potentials associated with cues using RL and feedback from the user in the form of ErrPs. The system adapted to the control and SCI subject and achieved comparable performance levels for both. By using the system, the user was able to control a FES. The system could be used in future work as a testbed for augmenting rehabilitation with a BCI.

By adjusting the way the system decodes motor potentials the RL BCI can be further improved beyond static BCI's. Neural networks suffer from a dilemma known as the bias-variance tradeoff. Networks too small suffer from poor performance in later trials while networks too large have poor performance in early trials. Choosing a size for a neural network using RL is difficult because the data set is always increasing. However by starting with a small network and adding inputs using dynamic feature addition, the performance of a RL BCI can be improved.

In addition to adding new features over time, the order in which features are added was also investigated. Instead of adding features in a fixed order, adding features that showed the greatest difference between cues in earlier trials improved performance. This change in the order in which features are added further improved the dynamic feature addition and the RL BCI.

RL also offers the opportunity to change the learning rate online during the experiment. To test how to best utilize this opportunity various update rules were investigated: back propagation, scaled back propagation, Hebbian style learning, and scaled Hebbian style learning. Hebbian style learning performed better than back propagation; and by taking advantage of the online nature of RL, scaled Hebbian style learning performed better than Hebbian style learning.

In future work when the system starts to perform at a level that subjects would be willing to use it every day for rehabilitation, techniques could be introduced to modify the performance of the system to ensure continued progress of the rehabilitation. The threshold for an action could be increased or the inputs to the actor could be limited to certain frequencies or electrodes, which could be helpful in guiding the continued rehabilitation of the subject. The ideal brain activity for a subject undergoing rehabilitation could also be investigated and might not necessarily match an able-bodied subject.

The task used during rehabilitation could also be changed. Tasks that are more engaging could be used and standard rehabilitation test like the Jebsen hand function test could be integrated into the system. The cues and feedback could be switched to auditory signals freeing the subject's visual attention.

Auditory cues and feedback could also be embedded in music making them more pleasant for the user.

These findings have important implications for BCI's and rehabilitation. As rehabilitation induces positive cortical reorganization in the user, frequent adjustments to the system might become necessary because of the user's changing motor potentials. A system that adapted to the user as quickly as their motor potentials changed could reduce the need for recalibration enabling longer rehabilitation sessions. The system was able to use the weights from the last session to decode the user's motor potentials at the start of a new session. As the system learned the user's motor potentials, changes in the system's weights became smaller meaning the user would experience consistent performance. Since the system does not need to be recalibrated by a scientist, the user could take the system home and use it continuously.

REFERENCES

Abu-Mostafa, Y. S. (1989). "The Vapnik-Chervonenkis Dimension: Information Versus Complexity in Learning." Neural Computation **1**(3): 312.

Aghdam, M. H., N. Ghasem-Aghaee, et al. (2009). "Text Feature Selection Using Ant Colony Optimization." Expert Systems with Applications **36**(3): 6843.

Al-Ani, A. (2006). "Feature Subset Selection Using Ant Colony Optimization." International Journal of Computational Intelligence **2**(1).

Anderson, K. D. (2004). "Targeting Recovery: Priorities of the Spinal Cord-Injured Population." Journal of Neurotrauma **21**(10): 1371.

Ash, T. (1989). "Dynamic Node Creation in Backpropagation Networks." Connection Science **1**(4): 365.

Bartlett, E. B. (1994). "Dynamic Node Architecture Learning: An Information Theoretic Approach." Neural Networks **7**(1): 129.

Barzilai, J. and J. M. Borwein (1988). "Two-Point Step Size Gradient Methods." IMA Journal of Numerical Analysis **8**(1): 141.

Battiti, R. (1994). "Using Mutual Information for Selecting Features in Supervised Neural Net Learning." Neural Networks, IEEE Transactions on **5**(4): 537.

Bauer Jr, K. W., S. G. Alsing, et al. (2000). "Feature Screening Using Signal-to-Noise Ratios." Neurocomputing **31**(1): 29.

Baum, E. B. and D. Haussler (1989). "What Size Net Gives Valid Generalization?" Neural Computation **1**(1): 151.

Beekhuizen, K. S. and E. C. Field-Fote (2008). "Sensory Stimulation Augments the Effects of Massed Practice Training in Persons with Tetraplegia." Archives of Physical Medicine and Rehabilitation **89**(4): 602.

Behera, L., S. Kumar, et al. (2006). "On Adaptive Learning Rate That Guarantees Convergence in Feedforward Networks." Neural Networks, IEEE Transactions on **17**(5): 1116.

Belue, L. M. and K. W. Bauer Jr (1995). "Determining Input Features for Multilayer Perceptrons." Neurocomputing **7**(2): 111.

Bishop, C. M. (1995). Neural Networks for Pattern Recognition, Oxford university press.

Brill, F. Z., D. E. Brown, et al. (1992). "Fast Generic Selection of Features for Neural Network Classifiers." Neural Networks, IEEE Transactions on **3**(2): 324.

Cramer, S., E. Orr, et al. (2007). "Effects of Motor Imagery Training after Chronic, Complete Spinal Cord Injury." Experimental Brain Research **177**(2): 233.

Cramer, S. C., L. Lastra, et al. (2005). "Brain Motor System Function after Chronic, Complete Spinal Cord Injury." Brain **128**(12): 2941.

Daly, J. J., R. Cheng, et al. (2009). "Feasibility of a New Application of Noninvasive Brain Computer Interface (BCI): A Case Study of Training for Recovery of Volitional Motor Control after Stroke." Journal of Neurologic Physical Therapy **33**(4): 203.

Daly, J. J. and J. R. Wolpaw (2008). "Brain–Computer Interfaces in Neurological Rehabilitation." Lancet neurology **7**(11): 1032.

DiGiovanna, J., B. Mahmoudi, et al. (2009). "Coadaptive Brain-Machine Interface Via Reinforcement Learning." Biomedical Engineering, IEEE Transactions on **56**(1): 54.

Elbert, T., C. Pantev, et al. (1995). "Increased Cortical Representation of the Fingers of the Left Hand in String Players." Science **270**(5234): 305.

Falkenstein, M., J. Hoormann, et al. (2000). "ERP Components on Reaction Errors and Their Functional Significance: A Tutorial." Biological Psychology **51**(2-3): 87.

Ferrez, P. W. and J. Millan (2008). "Error-Related EEG Potentials Generated During Simulated Brain-Computer Interaction." Biomedical Engineering, IEEE Transactions on **55**(3): 923.

Frean, M. (1990). "The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks." Neural Computation **2**(2): 198.

Gehring, W., M. Coles, et al. (1990). "The Error-Related Negativity: An Event-Related Brain Potential Accompanying Errors." Psychophysiology **27**(4): S34.

Geman, S., E. Bienenstock, et al. (1992). "Neural Networks and the Bias/Variance Dilemma." Neural computation **4**(1): 1.

Gori, M. and A. Tesi (1992). "On the Problem of Local Minima in Backpropagation." IEEE Transactions on Pattern Analysis and Machine Intelligence **14**(1): 76.

Green, J., E. Sora, et al. (1998). "Cortical Sensorimotor Reorganization after Spinal Cord Injury an Electroencephalographic Study." Neurology **50**(4): 1115.

Gruau, F., D. Whitley, et al. (1996). A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks. Proceedings of the First Annual Conference on Genetic Programming, MIT Press.

Hebb, D. (1949). The Organization of Behavior: A Neuropsychological Theory New York, John Wiley.

Hirose, Y., K. Yamashita, et al. (1991). "Back-Propagation Algorithm Which Varies the Number of Hidden Units." Neural Networks **4**(1): 61.

Hoffman, L. R. and E. C. Field-Fote (2006). "Cortically-Evoked Potentials of Muscles Distal to the Lesion Are Posteriorly Shifted and of Lower Amplitude in Individuals with Tetraplegia Due to Spinal Cord Injury." Journal of Neurologic Physical Therapy **30**(4): 202.

Hoffman, L. R. and E. C. Field-Fote (2007). "Cortical Reorganization Following Bimanual Training and Somatosensory Stimulation in Cervical Spinal Cord Injury: A Case Report." Physical Therapy **87**(2): 208.

Hoffman, L. R. and E. C. Field-Fote (2010). "Functional and Corticomotor Changes in Individuals with Tetraplegia Following Unimanual or Bimanual Massed Practice Training with Somatosensory Stimulation: A Pilot Study." Journal of Neurologic Physical Therapy **34**(4): 193.

Hohnsbein, J., M. Falkenstein, et al. (1989). "Error Processing in Visual and Auditory Choice Reaction Tasks." Journal of Psychophysiology **3**: 32.

Holroyd, C. B. and M. G. H. Coles (2002). "The Neural Basis of Human Error Processing: Reinforcement Learning, Dopamine, and the Error-Related Negativity." Psychological Review **109**(4): 679.

Hsin, H.-C., C.-C. Li, et al. (1995). "An Adaptive Training Algorithm for Back-Propagation Neural Networks." Systems, Man and Cybernetics, IEEE Transactions on **25**(3): 512.

Hua, J., Z. Xiong, et al. (2005). "Optimal Number of Features as a Function of Sample Size for Various Classification Rules." Bioinformatics **21**(8): 1509.

Hummel, F. C. and L. G. Cohen (2006). "Non-Invasive Brain Stimulation: A New Strategy to Improve Neurorehabilitation after Stroke?" The Lancet Neurology **5**(8): 708.

Hush, D. R., B. Horne, et al. (1992). "Error Surfaces for Multilayer Perceptrons." Systems, Man and Cybernetics, IEEE Transactions on **22**(5): 1152.

Hush, D. R. and B. G. Horne (1993). "Progress in Supervised Neural Networks." Signal Processing Magazine, IEEE **10**(1): 8.

Jacobs, R. A. (1988). "Increased Rates of Convergence through Learning Rate Adaptation." Neural Networks **1**(4): 295.

Jian, F. and X. Yugeng (1997). "Neural Network Design Based on Evolutionary Programming." Artificial Intelligence in Engineering **11**(2): 155.

Kabir, M. M., M. Shahjahan, et al. (2012). "A New Hybrid Ant Colony Optimization Algorithm for Feature Selection." Expert Systems with Applications **39**(3): 3747.

Kleim, J. A., S. Barbay, et al. (1998). "Functional Reorganization of the Rat Motor Cortex Following Motor Skill Learning." Journal of Neurophysiology **80**(6): 3321.

Kokotilo, K. J., J. J. Eng, et al. (2009). "Reorganization and Preservation of Motor Control of the Brain in Spinal Cord Injury: A Systematic Review." Journal of Neurotrauma **26**(11): 2113.

Kuan, C.-M. and K. Hornik (1991). "Convergence of Learning Algorithms with Constant Learning Rates." Neural Networks, IEEE Transactions on **2**(5): 484.

Laine, T. I., K. Bauer, et al. (2002). "Selection of Input Features across Subjects for Classifying Crewmember Workload Using Artificial Neural Networks." Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on **32**(6): 691.

LeCun, Y., L. Bottou, et al. (1998). Efficient Backprop. Neural Networks: Tricks of the Trade, Springer**:** 9.

Liepert, J., H. Bauder, et al. (2000). "Treatment-Induced Cortical Reorganization after Stroke in Humans." Stroke **31**(6): 1210.

MacLeod, C., G. Maxwell, et al. (2009). "Incremental Growth in Modular Neural Networks." Engineering Applications of Artificial Intelligence **22**(4–5): 660.

MacLeod, C. and G. M. Maxwell (2001). "Incremental Evolution in Anns: Neural Nets Which Grow." Artificial Intelligence Review **16**(3): 201.

Magoulas, G. D., M. N. Vrahatis, et al. (1997). "Effective Backpropagation Training with Variable Stepsize." Neural Networks **10**(1): 69.

Mahmoudi, B., J. DiGiovanna, et al. (2008). Neuronal Tuning in a Brain-Machine Interface During Reinforcement Learning. International Conference of the IEEE Engineering in Medicine and Biology Society 2008, IEEE.

Mahmoudi, B. and J. C. Sanchez (2011). "A Symbiotic Brain-Machine Interface through Value-Based Decision Making." PLoS ONE **6**(3): e14760.

Marino, R., T. Barros, et al. (2003). "International Standards for Neurological Classification of Spinal Cord Injury." The Journal of Spinal Cord Medicine **26**: S50.

Masters, T. (1993). Practical Neural Network Recipes in C++, Morgan Kaufmann.

Mazzoni, P., R. A. Andersen, et al. (1991). "A More Biologically Plausible Learning Rule for Neural Networks." Proceedings of the National Academy of Sciences **88**(10): 4433.

McFarland, D. J. and J. R. Wolpaw (2011). "Brain-Computer Interfaces for Communication and Control." Communications of the ACM **54**(5): 60.

Middendorf, M., G. McMillan, et al. (2000). "Brain-Computer Interfaces Based on the Steady-State Visual-Evoked Response." Rehabilitation Engineering, IEEE Transactions on **8**(2): 211.

Millan, J. R., F. Renkens, et al. (2004). "Noninvasive Brain-Actuated Control of a Mobile Robot by Human EEG." Biomedical Engineering, IEEE Transactions on **51**(6): 1026.

Müller-Putz, G. R., R. Scherer, et al. (2005). "EEG-Based Neuroprosthesis Control: A Step Towards Clinical Practice." Neuroscience Letters **382**(1–2): 169.

Neuper, C., M. Wörtz, et al. (2006). ERD/ERS Patterns Reflecting Sensorimotor Activation and Deactivation. Progress in Brain Research. N. Christa and K. Wolfgang, Elsevier. **Volume 159:** 211.

Nudo, R. J. and G. W. Milliken (1996). "Reorganization of Movement Representations in Primary Motor Cortex Following Focal Ischemic Infarcts in Adult Squirrel Monkeys." Journal of Neurophysiology **75**(5): 2144.

Nudo, R. J., B. M. Wise, et al. (1996). "Neural Substrates for the Effects of Rehabilitative Training on Motor Recovery after Ischemic Infarct." Science **272**(5269): 1791.

Ochoa, J. M., M. Listenberger, et al. (2011). Use of an Electromyographically Driven Hand Orthosis for Training after Stroke. International Conference on Rehabilitation Robotics 2011, IEEE.

Pal, S. K., R. K. De, et al. (2000). "Unsupervised Feature Evaluation: A Neuro-Fuzzy Approach." Neural Networks, IEEE Transactions on **11**(2): 366.

Pascual-Leone, A., A. Cammarota, et al. (1993). "Modulation of Motor Cortical Outputs to the Reading Hand of Braille Readers." Annals of Neurology **34**(1): 33.

Pascual-Leone, A., J. Grafman, et al. (1994). "Modulation of Cortical Motor Output Maps During Development of Implicit and Explicit Knowledge." Science **263**(5151): 1287.

Pfurtscheller, G., G. R. Müller, et al. (2003). "'Thought' – Control of Functional Electrical Stimulation to Restore Hand Grasp in a Patient with Tetraplegia." Neuroscience Letters **351**(1): 33.

Plagianakos, V., D. Sotiropoulos, et al. (1998). "Automatic Adaptation of Learning Rate for Backpropagation Neural Networks." Recent Advances in Circuits and Systems: 337.

Pohlmeyer, E. A., B. Mahmoudi, et al. (2012). Brain-Machine Interface Control of a Robot Arm Using Actor-Critic Reinforcement Learning. International Conference of the IEEE Engineering in Medicine and Biology Society 2012, IEEE.

Polak, E. (1997). Optimization: Algorithms and Consistent Approximations, Springer-Verlag New York, Inc.

Prechelt, L. (1998). Early Stopping - but When? Neural Networks: Tricks of the Trade. G. Orr and K.-R. Müller, Springer Berlin / Heidelberg. **1524:** 553.

Priddy, K. L., S. K. Rogers, et al. (1993). "Bayesian Selection of Important Features for Feedforward Neural Networks." Neurocomputing **5**(2): 91.

Pudil, P., J. Novovičová, et al. (1994). "Floating Search Methods in Feature Selection." Pattern Recognition Letters **15**(11): 1119.

Qin, L., L. Ding, et al. (2004). "Motor Imagery Classification by Means of Source Analysis for Brain–Computer Interface Applications." Journal of Neural Engineering **1**(3): 135.

Ruck, D. W., S. K. Rogers, et al. (1990). "Feature Selection Using a Multilayer Perceptron." Journal of Neural Network Computing **2**(2): 40.

Setiono, R. and L. C. K. Hui (1995). "Use of a Quasi-Newton Method in a Feedforward Neural Network Construction Algorithm." Neural Networks, IEEE Transactions on **6**(1): 273.

Setiono, R. and H. Liu (1997). "Neural-Network Feature Selector." Neural Networks, IEEE Transactions on **8**(3): 654.

Sivagaminathan, R. K. and S. Ramakrishnan (2007). "A Hybrid Approach for Feature Subset Selection Using Neural Networks and Ant Colony Optimization." Expert Systems with Applications **33**(1): 49.

Śmieja, F. J. (1993). "Neural Network Constructive Algorithms: Trading Generalization for Learning Efficiency?" Circuits, Systems and Signal Processing **12**(2): 331.

Solomon, R. and J. Leo van Hemmen (1996). "Accelerating Backpropagation through Dynamic Self-Adaptation." Neural Networks **9**(4): 589.

Stanley, K. O. and R. Miikkulainen (2002). Efficient Reinforcement Learning through Evolving Neural Network Topologies. In Proceedings of the Genetic and Evolutionary Computation Conference 2002, Morgan Kaufmann.

Steppe, J. and K. Bauer Jr (1997). "Feature Saliency Measures." Computers & Mathematics with Applications **33**(8): 109.

Steppe, J. M. and K. W. Bauer Jr (1996). "Improved Feature Screening in Feedforward Neural Networks." Neurocomputing **13**(1): 47.

Sutton, R. S. and A. G. Barto (1998). Reinforcement Learning: An Introduction, Cambridge Univ Press.

Vapnik, V. N. and A. Y. Chervonenkis (1971). "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities." Theory of Probability & Its Applications **16**(2): 264.

Verikas, A. and M. Bacauskiene (2002). "Feature Selection with Neural Networks." Pattern Recognition Letters **23**(11): 1323.

Vidal, J. J. (1977). "Real-Time Detection of Brain Events in EEG." Proceedings of the IEEE **65**(5): 633.

Weir, M. K. (1991). "A Method for Self-Determination of Adaptive Learning Rates in Back Propagation." Neural Networks **4**(3): 371.

Weng, W. and K. Khorasani (1996). "An Adaptive Structure Neural Networks with Application to EEG Automatic Seizure Detection." Neural Networks **9**(7): 1223.

Whitney, A. W. (1971). "A Direct Method of Nonparametric Measurement Selection." Computers, IEEE Transactions on **100**(9): 1100.

Widrow, B. and M. A. Lehr (1990). "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation." Proceedings of the IEEE **78**(9): 1415.

Wyndaele, M. and J. J. Wyndaele (2006). "Incidence, Prevalence and Epidemiology of Spinal Cord Injury: What Learns a Worldwide Literature Survey?" Spinal Cord **44**(9): 523.

Yang, J. and V. Honavar (1998). Feature Subset Selection Using a Genetic Algorithm. Feature Extraction, Construction and Selection, Springer**:** 117.

Yu, X.-H. (1992). "Can Backpropagation Error Surface Not Have Local Minima." Neural Networks, IEEE Transactions on **3**(6): 1019.

Yu, X.-H. and G.-A. Chen (1997). "Efficient Backpropagation Learning Using Optimal Learning Rate and Momentum." Neural Networks **10**(3): 517.

Yu, X.-H., G.-A. Chen, et al. (1995). "Dynamic Learning Rate Optimization of the Backpropagation Algorithm." Neural Networks, IEEE Transactions on **6**(3): 669.

Zhang, B.-T. and H. Muhlenbein (1993). "Evolving Optimal Neural Networks Using Genetic Algorithms with Occam's Razor." Complex Systems **7**(3): 199.