2015-05-05

Synchronous Control of a Reinforcement Learning Based Brain-Machine Interface With Biological Feedback

Noeline Prins
University of Miami, noeline.prins@gmail.com

UNIVERSITY OF MIAMI

SYNCHRONOUS CONTROL OF A REINFORCEMENT LEARNING BASED BRAIN-MACHINE INTERFACE WITH BIOLOGICAL FEEDBACK

By

Noeline W. J. A. L. Prins

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
May 2015

UNIVERSITY OF MIAMI

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

SYNCHRONOUS CONTROL OF A REINFORCEMENT LEARNING BASED BRAIN-MACHINE INTERFACE WITH BIOLOGICAL FEEDBACK

Noeline W. J. A. L. Prins

Approved:

Justin C. Sanchez, Ph.D. Associate Professor of Biomedical Engineering	Ozcan Ozdamar, Ph.D. Professor and Chairman of Biomedical Engineering
Abhishek Prasad, Ph.D. Assistant Professor of Biomedical Engineering	Jorge E. Bohorquez, Ph.D. Associate Professor in Practice of Biomedical Engineering
Christopher Bennett, Ph.D. Research Assistant Professor of Music Engineering Technology	M. Brian Blake, Ph.D. Dean of the Graduate School

PRINS, NOELINE W. J. A. L. (Ph.D., Biomedical Engineering)
Synchronous Control of a Reinforcement Learning Based Brain-Machine Interface with Biological Feedback (May 2015)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Justin C. Sanchez.
No. of pages in text. (131)

Brain-Machine Interfaces (BMIs) have the potential of restoring functionality of persons suffering from paralysis and amputations. At present, BMIs have been developed to use cortical neural signals and control prosthetic devices or to stimulate paralyzed limbs. However, these BMIs rely on an external training signal (usually desired kinematics) as a reference to infer an error signal to be able to adapt the decoder appropriately and learn the task. For amputees and paralyzed persons, a desired kinematic cannot be measured directly. We propose to acquire an error or reward signal from the brain itself as a training signal for motor decoders. For this, we adopt Actor-Critic Reinforcement Learning (RL) paradigm to use as a BMI. There are several challenges associated with obtaining an error signal from the brain. One of the challenges is due to the unstationary nature of neural signals, the classification of the error being low and there being no indication as to the level of accuracy of the signal. If an indication can be extracted as to the accuracy of the signal, we propose that such a system can maintain performance of the BMI even when the error signal is less than perfect. This is done by incorporating a confidence metric in the weight update rule, where the confidence metric indicates the accuracy of the signal. We propose a synchronous BMI where the forward path is provided by the motor

cortex and the feedback path is provided by the striatum. Computer simulations on synthetic data were performed to test the architecture. The confidence metric mentioned above can be obtained by different methods; the distance to the boundary and a probabilistic measure were implemented. The confidence arrived from the different classification methods (distance/ probability) was thresholded to give three output classes indicating rewarding, non-rewarding or ambiguous. As the threshold increased from zero, the performance increased and as the threshold increased further, the performance dropped. By this we conclude that there exists an optimum threshold for the Critic data where even though the Critic feedback is noisy, the Actor can maintain its performance. The said system was implemented in closed-loop with a monkey using a probabilistic classifier, where the probability of the Critic output belonging to one class or the other was used as the confidence measure. With using the confidence measure the performance of the system was improved.

This dissertation is dedicated to

my Lord and Savior, Jesus Christ for the freedom of life and the freedom to explore life,
and my parents, who instilled a sense of curiosity and exploration in me,

both of which inspired me to be the Engineer I am today.

Acknowledgements

Acquiring a Ph.D. is an endeavor one does not embark alone. There were several people without whose scientific guidance and personal influence, this task would not have been completed. First and foremost I want to thank my advisor and Principal Investigator of the Neuroprosthetics Research Group (NRG) Justin Sanchez, for his continuous support throughout my Ph.D. career. His motivation, enthusiasm, wisdom and guidance in directing me assisted not only during research and writing, but also to make me the independent researcher I am today. I would also like to thank Abhishek Prasad, co-P.I. of NRG for his untiring efforts and mentoring throughout my time as a Ph.D. student. His counsel and advice and dedication has inspired me. I would like to extend my gratitude to the rest of my committee, Ozcan Ozdamar, Jorge Bohorquez and Christopher Bennett for their insightful comments and hard questions to assure that the dissertation met the scientific rigor, and for keeping me on my toes. I believe I have tried their patience on more than one occasion, but they have been very enduring.

I wish to also thank Dr. Daniel Rothen, Veterinarian, the staff of Division of Veterinary Resources (DVR) and husbandry staff at University of Miami for the assistance with animal surgery and animal care. My gratitude goes to the administrative staff of Biomedical Engineering at University of Miami, Angie Del-Llano, Melissa Dietrick and Vivian Figueredo, for all the assistance given in

matters related to documentation and sending out reminders, which meant I had one less thing to worry about every week.

A word of thanks to the Defense Advanced Research Projects Agency (DARPA) for the funding support for the project I was involved in (N66001-10-C-2008). My thanks is extended to the collaborators at University of Florida (Principe Lab), State University New York (Lytton Lab and Francis Lab) and others in the, DAPRA Reorganization and Plasticity to Accelerate Injury Recovery (REPAIR) program.

It is good to have living examples of those who are a step ahead of oneself, close at hand. My experience in the lab was greatly enhanced by such post-doctoral associates at NRG; Eric Pohlmeyer, Babak Mahmoudi and Rowshanak Hashemiyoon. I wish to thank them for their mentoring in different aspects of my research and their life advice in general which will not be taken lightly.

I want thank my fellow lab mates Katie Gant and Qi Yu. I wish to extend my thanks to Shubham Debnath, a fellow Ph.D. student with the help in animal experiments. Fellow Ph.D. student and programming wiz in the group, Ziqian Xie and Charles Prins for help with C++ and Python coding; both of them spent many a night trying to teach a MATLAB programmer to start counting from zero. A very special thanks to Shijia Geng, fellow lab mate and friend for the stimulating discussions, sleepless nights before deadlines, experimental work and all the fun we have had bonding over monkeys. They say the best way to learn is teach,

whether it be mentoring undergraduates or teaching neurosurgery to other graduate students; I wish to thank them for their assistance in lab matters, the opportunity to teach, train and mentor them, and their feedback on my performance as a mentor.

A personal note of thanks to my roommates, Sheba Johnson, Saptaparni Bandyopadhyay and Vy Vo, for making the last few years a pleasant experience at home and putting up with the lifestyle of a Ph.D. student; Anuradha Gunathilake, Biology Ph.D. student and my dissertation writing buddy, for not making me feel alone during the writing process; the Kalbac family and Cameron family for providing "home(s) away from home" and adopting me into their families wholeheartedly, their spiritual insights and being available at any time day or night as a family would; and my church pastor, Joe Lortie for spiritual guidance and the long hours of helping through decision making. Lastly, I wish to thank my parents and brother, Charles for the immense patience and the love extended *daily* across 9800 miles and 10 ½ hour time difference. I am truly blessed to have such a family.

Noeline Prins

University of Miami

May 2015

"Worthy art thou, our Lord and God, to receive glory and honor and power, for thou didst create all things, and by thy will they existed and were created." Revelation 4:11

Contents

List of Tables

List of Figures

Abbreviations

AD	:	Alzheimer's Disease
ADL	:	Activities of Daily Living
ALS	:	Amyotrophic Lateral Sclerosis
ANN	:	Artificial Neural Network
AP	:	Anterior-Posterior
BCI	:	Brain Computer Interface
BG	:	Basal Ganglia
BMIs	:	Brain-Machine Interfaces
CA	:	Computer Agent
CAR	:	Common Average Reference
CL	:	Closed-Loop
CRI	:	Constant Rate Infusion
DA	:	Dopamine
DBS	:	Deep Brain Stimulation
ECoG	:	Electrocorticography
EEG	:	Electroencephalography
FES	:	Functional Electrode Stimulation
FFT	:	Fast-Fourier Transform
HRL	:	Hebbian Reinforcement Learning
IA	:	Interaural
IM	:	intramuscular
k-NN	:	k-Nearest Neighbor
LDA	:	Linear Discriminant Analysis
LFP	:	Local Field Potentials

LIS	:	Locked-In Syndrome
MI	:	Primary Motor Cortex
ML	:	Medial-Lateral
MLP	:	Multi-Layer Perceptron
MSP	:	Medium Spiny Projection
NAcc	:	Nucleus Accumbens
NHP	:	Non-Human Primates
OL	:	Open Loop
PARC	:	Perception Action Reward Cycle
PCA	:	Principal Component Analysis
PCs	:	Principal Components
PE	:	Processing Element
PVA	:	Population Vector Algorithm
RL	:	Reinforcement Learning
RLBMI	:	Reinforcement Learning Brain-Machine Interface
RM	:	Robot Movement
SCI	:	Spinal Cord Injury
SL	:	Supervised Learning
SNR	:	Signal to Noise Ratio
SUA	:	Single Unit Activity
SVM	:	Support Vector Machine
TD	:	Temporal Difference
TDNN	:	Time Delay Neural Network
VFE	:	Value Function Estimator
wrt	:	with respect to

Chapter 1 Introduction and Background

1.1 Motivation

According to 2009 statistics published by Christopher Reeve Foundation, paralysis affects 1.9% (5.596 million people) of the U.S. population. Various types of accidents (motor vehicle, work place, and falling) accounted for the great majority of spinal cord injuries. Additionally, stroke (29%), spinal cord injury (SCI) (23%), and multiple sclerosis (17%) were the other causes. 250,000 Americans are spinal cord injured with approximately 11,000 new injuries occurring each year. The annual health care, living expenses, and estimated lifetime costs that are directly attributable to SCI vary greatly according to severity of injury. Average lifetime costs for quadriplegics are estimated at $1.35 million (age of injury 25). The percentage of SCI individuals unemployed eight years after injury is 63%. For amyotrophic lateral sclerosis (ALS) patients, annual incidence rate is roughly 2 people per 100,000 [1-3].

For some kinds of paralysis like locked-in syndrome (LIS), there are therapies such as functional neuromuscular stimulation, which may help activate some paralyzed muscles. Another approach for treating loss of sensorimotor function is bypassing damaged areas with electronics, which is known as Brain-Machine Interface (BMI). Several research groups have attempted to control external devices (computer cursor or robotic arm) or patient's own limbs using functional electrode stimulation (FES) [4-7]. BMI research, including this study, is

motivated by the need to help such people with sensorimotor loss. This chapter gives an introduction to the work in the field of BMI.

1.2 Background of BMI Work

BMIs attempt to link the brain to the external environment. The BrainGate neural interface system showed that people with tetraplegia were able to use a neural interface system to control a robotic arm to perform three-dimensional reach and grasp movements. Subjects were able to control the robotic arm and hand over a broad space without explicit training, using signals decoded from a small population of motor cortex (MI) neurons recorded from a 96-channel microelectrode array [8-10]. While a simple task like this is possible, research has not yet been able to give a solution in terms of adapting BMI for activities of daily living (ADL). This section will discuss BMI classification and the work done in BMI decoders at present.

1.2.1 BMI Classification

Sensory, Motor and Cognitive BMIs

BMIs depending upon the application are broadly divided as sensory, motor (neural prosthesis), or cognitive BMIs. Sensory BMIs like cochlear implants restore sensory function, while motor BMIs restore motor functions. This takes place either by stimulating a person's own limb, as in the case of FES, or through control of an external device (computer cursor, prosthetic device, robotic arm), like the BrainGate system in humans [4-8]. Cognitive BMIs restore neural interactions within damaged internal networks while incorporating perception.

Therapy for Alzheimer's Disease (AD) by Deep Brain Stimulations (DBS) is an example of cognitive BMIs [11].

Adaptive Decoders and Static Decoders

A decoder in BMI is what interprets the neural signals and converts them to an executable action(s). Decoders can be divided broadly into static and adaptive decoders based on the weight (parameter) update. Adaptive decoders change the decoder parameters (weights) to make adjustments to changes in the neural input, while static decoder weights are fixed and do not update. In research, animals have learned to gain control of static decoders over longer periods of time. However, adaptive decoders are more popular as they are able to reorganize and update themselves amidst large input perturbations [12-14].

Supervised, Semi-Supervised and Unsupervised Learning

Machine learning techniques are divided based upon the assistance needed for learning; supervised, unsupervised, and semi-supervised learning. Supervised Learning (SL) infers a function from labeled training data, which in BMI applications can either be real or inferred kinematic signals. Supervised training requires an explicit training signal, whereas, in cases of severe paralysis or amputation of bilateral limbs, it may not be possible to collect these training signals. Therefore, there is a need to develop other means of acquiring training signals and using them to adapt neural decoders. In contrast, unsupervised learning techniques do not rely on external training signals, but only on the patterns in the input data. Semi-supervised learning techniques are a compromise between supervised and unsupervised learning.

Synchronous and Asynchronous Control

Synchronous control systems are synchronized to an external reference, usually a go cue given by the experimenter indicating the beginning of 'trial'. In synchronous BMIs, the decoded neural data is channeled to the actuator only during these trial periods. In contrast, asynchronous BMIs can be connected to the actuator continuously and the control can be intermittent, starting whenever the user wishes to operate the actuator. Most of the work done in these self-paced controllers is on non-invasive controllers or Brain Computer Interfaces (BCI).

1.2.2 Decoders Used in BMIs

The initial animal BMIs were open loop (OL), with no feedback provided on the accuracy of the decisions made [15]. Subsequent closed-loop (CL) experiments were performed where the animal had an audio and/or visual feedback (Figure 1.1 A), which changed the brain states related to perception. But these brain states were not a part of the BMI [15]. This section explains briefly the common decoders used in research at present. The equations in each sub-section is independent from other sub-sections.

Linear Regression

A robotic arm was controlled in real time by predicting the arm trajectory of an owl monkey from neuronal activity of multiple cortical areas using both linear regression (Wiener filter – Figure 1.1 B) and artificial neural network (ANN) algorithms. This was shown both when the monkey had visual feedback and when he did not. The linear model used to predict hand position is; $Y = XA;$

where Y is the vector of kinematic and dynamic variables and X is the input vector of neuronal firing rates with time lags [15-17].

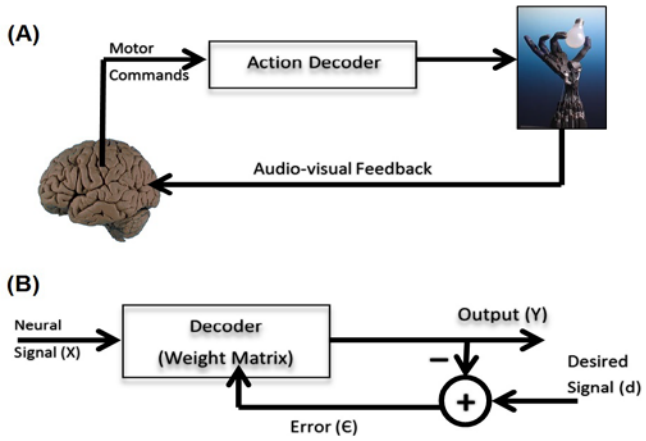

Figure 1.1: (A) Closed-Loop BMI. (B) Block diagram of the Wiener filter

Matrix A is solved by

$$\boldsymbol{A} = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T Y} \qquad \text{Eq. 1.1}$$

The error ϵ used to adapt the weight matrix is the difference between the desired signal d and the output of the decoder y (Figure 1.1 B).

$$\epsilon_i = d_i - y_i \qquad \text{Eq. 1.2}$$

Multiple regression can be used for joint kinematics and end point control of computer cursor/ robotic arm for fast and accurate control of cursor/ robotic arm, for spatial locations. The estimate of the position R at time t is given by

$$R(t) = \sum_i \sum_j a_{i,j} N(t+i,j) \qquad \text{Eq. 1.3}$$

where i is the time index and j is the cell number index. $N(i,j)$ is the activity of cell j at time i and $a_{i,j}$ represents the corresponding 'weight' [18-21].

Linear regressions rely heavily on the error signal, require extensive training and are confined to familiar movements.

Population Vector Algorithm

The population vector algorithm (PVA) assumes a cell's firing rate is a function of the velocity vector associated with the movement performed by the individual. This is based on the use of tuning curves (tuning curve relates the mean of movement-related cell activity to movement direction), which provides a statistical relationship between neural activity and behavior. The tuning (or preferred direction) of each cell in the ensemble conveys the average firing of a cell given a particular movement direction [6, 11, 22-26]. The PVA model relating the tuning to kinematics is given by

$$s_n(V) = B \cdot V = |B||V|cos\,\theta \qquad \text{Eq. 1.4}$$

where $s_n(V)$ gives the firing rate for n^{th} neuron with velocity vector V. The preferred direction is given by the weight vector, B. θ is the angle between the cell's preferred direction and movement direction. The magnitude of the vector contribution of each neuron in the direction of P is given by

$$w_n(V,t) = s_n(V) - b_0^n \qquad \text{Eq. 1.5}$$

where b_0 is the mean firing rate for n^{th} neuron. The population vector P is given by

$$\mathrm{P(V,t)} = \sum_{n=1}^{N} w_n(V,t)\frac{B_n}{||B_n||} \qquad \text{Eq. 1.6}$$

Kalman Filters

Kalman filter is a special condition of Bayesian recursive filter with the assumption of linearity and normal distribution (linearity and normality not assumed in the case of extended and unscented kalman filters). Several groups have adopted Kalman filter to predict movement trajectories where the system is assumed to be a linear dynamical system [27-33]. The Kalman filter estimates the next state, $\bar{x}(t)$, of a linear dynamical system based on the previous state. In BMI applications, the states are the hand position, velocity, and/or acceleration. The next state is given by

$$\bar{x}(t) = A\,x(t-1) + u(t) \qquad \text{Eq. 1.7}$$

and the output mapping (measurement prediction) is given by

$$\bar{z}(t) = C\,\bar{x}(t) + v(t) \qquad \text{Eq. 1.8}$$

where $u(t)$ is zero-mean Gaussian noise, $v(t)$ is zero-mean Gaussian measurement noise and $x(t-1)$ is a vector of neural firing rates. The update equation is given by

$$X_{est} = \bar{x}(t) + \mathrm{K}\,(z(t) - \bar{z}(t)) \qquad \text{Eq. 1.9}$$

where $z(t)$ is the actual measurement and $z(t) - \bar{z}(t)$ gives the correction term which is the difference between the actual and predicted measurement. K is the Kalman gain.

Recalibrated Feedback Intention-Trained Kalman Filter (ReFIT-KF) is a modification of the Kalman Filter to include a feedback perspective introducing a

causal intervention. The result alters the modeling assumptions and the training method [34-36].

1.3 Incorporating Feedback

For decoders used in BMI applications, we need to incorporate more biologic realism in order to have a completely autonomous system. The need for a more biologically realistic BMI is to have the ability to interact with the human brain, not only on a motor level, but also on a cognitive level. A paradigm for mutual adaptation (or co-adaptation) between humans and machines is important for neural rehabilitation and will open a new window for symbiotic human machine research [37]. This section focuses on using reinforcement learning (RL) as a way of incorporating motor and cognitive interactions in the BMI decoders.

1.3.1 Reinforcement Learning in the Brain

The perception–action–reward cycle (PARC) is the circular flow of information from the environment to sensory and motor structures and back to the environment completing the cycle during the processing of goal-directed behavior (Figure 1.2 A). All forms of adaptive behavior require PARC and the processing takes place both in series as well as parallel [38, 39]. The control of the goal-directed actions relies on the operation of an information-movement cycle. Every movement gives rise to a specific flow. This specificity translates into the existence of a continuous signal to inform the Actor about the validity of the produced movements with respect to the task at hand [40].

Different computations hypothesized to be occurring in separate brain regions are described in Figure 1.2 B: the cerebellum for SL (guided by the error signal), the basal ganglia (BG) for RL (guided by the reward signal) and the cerebral cortex for unsupervised learning (guided by the statistical properties of the input signal itself) [41, 42]. The principal components of the BG are the striatum, the pallidum, the substantia nigra and the subthalamic nucleus. Research has shown that the BG is involved in various aspects of psychomotor behavior even though it is not a major sensory relay nor a coordinating neuronal system [43].

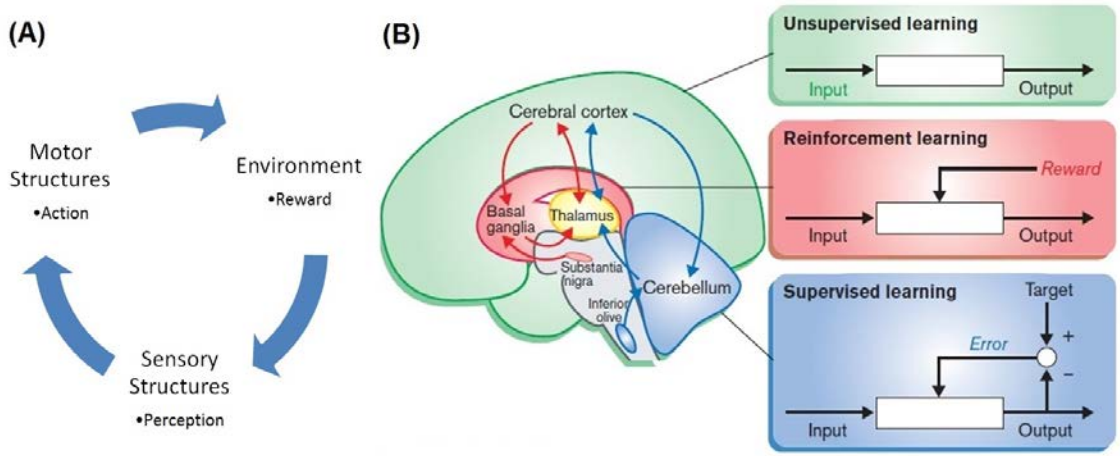

Figure 1.2: (A) Perception – Action – Reward Cycle (PARC). (B) Different Learning Architectures in the Brain: cerebellum for Supervised Learning (guided by the error signal), the basal ganglia (BG) for Reinforcement Learning (guided by the reward signal) and the cerebral cortex for Unsupervised Learning (guided by the statistical properties of the input signal itself) [42].

This kind of RL in the brain has motivated researchers to find alternative approaches to building BMIs. When mimicking RL in the brain, we need to find a structure in the brain that will give us a reward signal. Obtaining reward information has a variety of challenges associated with it. Much research has gone into identifying reward centers in the brain [44-47]. Of these centers, the nucleus accumbens (NAcc) is a main region in the ventral striatum and plays a

key role in the linking of reward to motor behavior and has been hypothesized to give the error signal for RL based BMIs [48-53].

1.3.2 Engineering a Biologically Realistic BMI

The PARC in goal-directed behavior provides key concepts in developing a new framework for BMI. The PARC relies on continuous processing of sensory information that adapts behavior and is used to guide goal-directed actions. This entire process is regulated by external environmental and internal neural feedback, which in turn guides the adaptation of computation and behavior [37, 38, 40, 54, 55]. The intention is to establish a direct communication channel between the user's brain and the machine with the goal of sharing the PARC with the user [37]. However, unlike the PARC that is central to animal interaction with the world, the PARC in a co-adaptive BMI will be distributed between the user and the computer agent (CA), thus incorporating two intelligent entities [37]. External training signals are not needed for the BMI, if the sensorimotor process interacts with the movement trajectories. The hypothesis is that the neural activity represents some form of evaluative feedback of the actions and can contribute to shaping future motor behaviors [38, 40, 54, 55]. To come up with a biologically realistic decoder, we need to incorporate both the action and reward as seen in Figure 1.3.

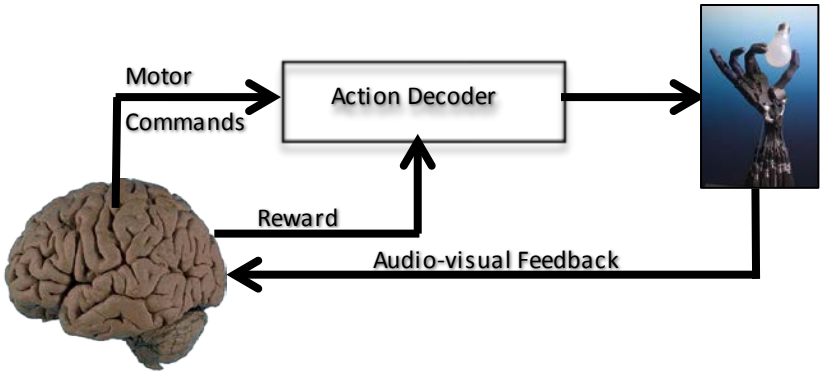

Figure 1.3: A more Biologically Realistic Architecture incorporating a reward signal from the brain

1.3.3 Reinforcement Learning Decoders

Conventional RL Decoders

RL provides a method of biological and computational learning that does not depend on an explicit training signal as in SL [56, 57]. The conventional RL paradigm involves an "agent" and an "environment" [58]. The agent is an intelligent being (e.g. computer algorithm) and the environment is anything that the agent interacts with and is able to influence through its actions (Figure 1.4 A). The agent makes an action at time t, which changes the state of the environment from S_t to S_{t+1} and receives the reward r_{t+1}. The goal of the agent is to maximize the cumulative reward (or return) R_t

$$\boldsymbol{R_t = \sum_{n=t+1}^{\infty} \gamma^{n-t+1} r_n} \qquad \text{Eq. 1.10}$$

where r_n is the reward earned at time n. The future rewards are discounted by the discount factor γ (≤ 1). The agent does not have knowledge if the selected action is optimal at the time the decision is made. This is only known later. By selecting suboptimal actions, the agent is "exploring". The agent must

"exploit" the situation by making a decision when he thinks it is optimal. There is always a dilemma between "exploration" and "exploitation" in RL.

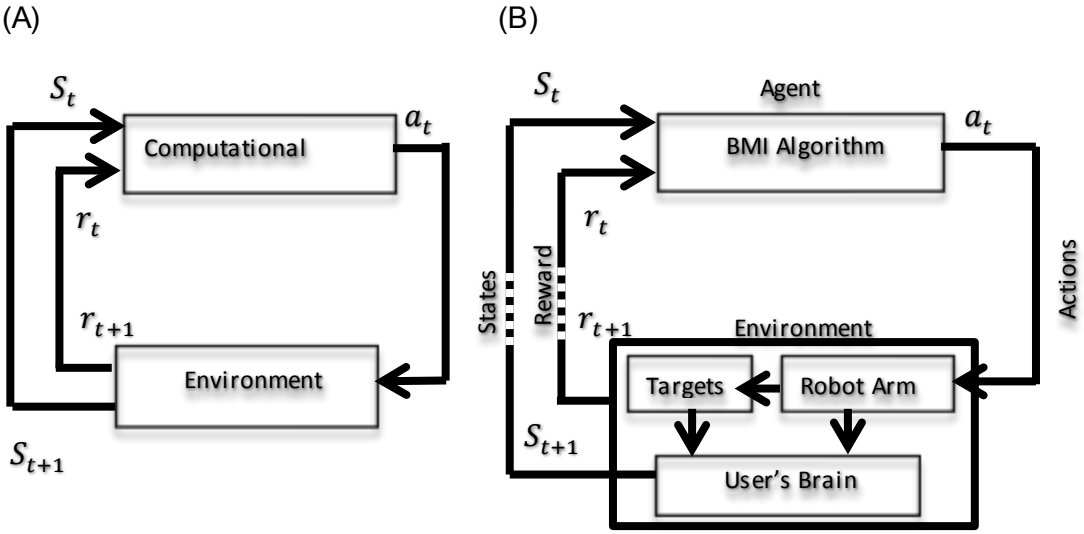

Figure 1.4: (A) Classical Reinforcement Learning Architecture. (B) Reinforcement Learning Architecture as applied to Brain-Machine Interfaces

The first BMI application of this architecture is shown Figure 1.4 B. The Agent is the BMI decoding algorithm and the Environment comprises of the user's brain, robot arm and the targets. The user acts through the BMI to accomplish tasks in the environment. The positions of the prosthetic and the target are the states of the environment. Since the user cannot move, their actions are a high level dialogue (neural modulations) with the BMI. The user seeks to learn a value for each action (neural modulation) given the relative position of the prosthetic (state) and the goal in order to achieve rewards [59].

This was experimentally shown to be a success for a rat involved in a 2 choice decision making task controlling a robot [60]. The value of the action selected needed to be estimated, and the value function estimator (VFE) was a fully connected multilayer perceptron (MLP) with three layers and hyperbolic

tangent as the activation function at the hidden layer nodes (Figure 1.5) [61]. The VFE $Q_k(S_t)$ is given by

$$Q_k(S_t) = \sum_j \tanh\left(\sum_i s_i w_{ij}\right) w_{jk} = \sum_j net_j(s_t) \cdot w_{jk} \qquad \text{Eq. 1.11}$$

where S_t, the neural state vector is the input to the MLP, and each output layer processing element (PE) estimates one action value given S_t and each PE also has a bias input.

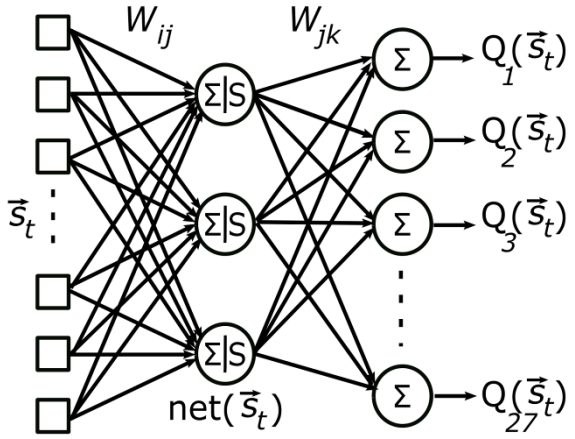

Figure 1.5: Value Function Estimation (VFE) network. [61]

Actor-Critic RL Paradigm

The next step to the RLBMI is to design a paradigm to incorporate the state signals from the brain and translate the reward directly from the user. For this we used the traditional Actor-Critic RL BMI architecture as shown in Figure 1.6 A. In this architecture, the Actor is used to select actions, and the VFE is known as the Critic, because it criticizes the actions made by the Actor [58]. After each action selection, the Critic evaluates the new state to determine whether the action selected led to a better state or worse state. This is given by the temporal-difference (TD) error δ_t.

$$\delta_t = r_{t+1} + \gamma V(S_{t+1}) - V(S_t) \qquad \text{Eq. 1.12}$$

where $V(\cdot)$ is the value function implemented by the Critic, r_{t+1} is the expected value of the reward at time $t+1$ and γ is the discount factor to reduce the weight of $V(S_{t+1})$ (i.e. future points). If the TD error is negative it suggests that the action led to a worse state [58].

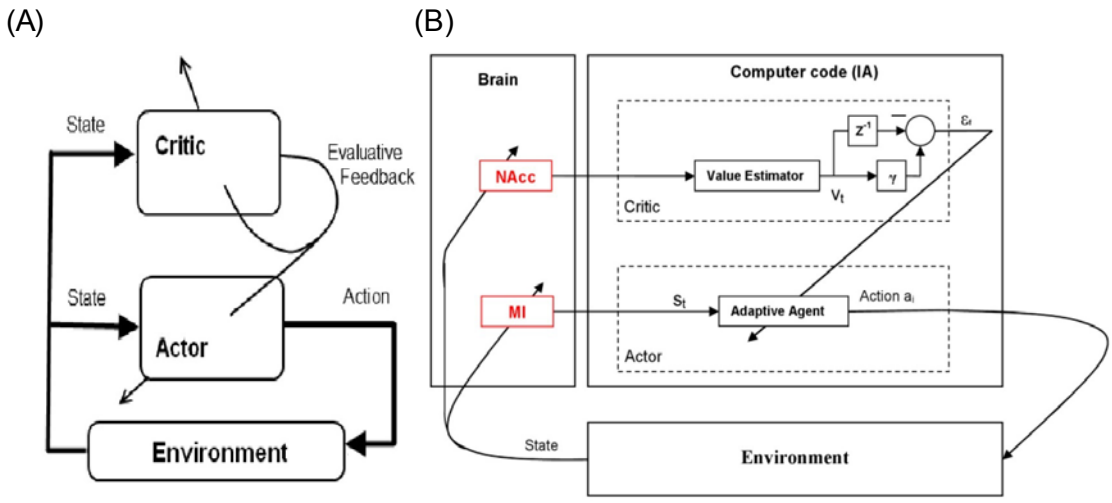

Figure 1.6: (A) Classical Actor Critic Reinforcement Learning Architecture. (B) Block diagram of the symbiotic BMI Controller. Actor driven by the MI and Critic driven by the NAcc

We adopted this Actor-Critic RL approach to develop a new method of decoding for the application in BMI [54, 61, 62]. In this approach, the Actor is driven by motor neural inputs and translates them into behavioral actions. The role of the Critic is to adapt the Actor based on experience. The only feedback the Critic requires is about the appropriateness (correct/ rewarding or incorrect/ non-rewarding or penalizing) of the action selected. This feedback signal can be obtained from the external environment or from the brain itself. The architecture in Figure 1.6 B is adopting the conventional RL in Figure 1.6 A to a biological learning paradigm of Figure 1.2 B. This was a first step in integrating robot action with biologic perception [60].

The first step in this new Actor-Critic RL paradigm was to have an adaptive agent (or “Actor”) that is able to make decisions based on a biological source. This was successfully implemented for a two-target choice task where a rat operated a robot arm. The adaptive algorithm was a time delay neural network (TDNN) with back-propagation [60]. The Critic was a value estimator, which used the entire trial labeled as either +1 or –1. The performance of the Critic in this case was slightly above chance. Another finding of this was that the Actor was able to maintain overall performance up to 10% inaccuracy of Critic, but the performance dropped drastically as the Critic accuracy reduced further [60].

If signals from this structure are to be used to adapt BMI decoders, a first step is to determine how to process and extract reward signals from it. The second step is, given the characteristics of the information related to reward that we can extract, to ascertain how best to use it in a biologically realistic BMI architecture. In the chapters to follow, we will present the processing and extraction of these reward signals and an engineering solution of how to incorporate these in to BMIs.

1.4 **Asynchronous Brain-Machine Interfaces**

Present BMIs, which are time-locked to a trial start time (synchronous BMIs), instruct the algorithm output when to be connected to the external device or prosthesis. Real life applications for such well-defined tasks are limited. When the subject is not engaged in the task, these synchronous BMIs are manually disengaged from the actuator. Having an autonomous gating mechanism to know

when the person is engaged and disengaged in the task will be helpful to have the BMI continuously ready but only active when the person is interested in performing the task. This will enable the BMI to be connected to the actuator 24/7, even when the subject returns home after the initial setup. A BMI that is not time-locked is an asynchronous BMI. Furthermore, such asynchronous BMIs can be used as a gating mechanism or switch to activate a call button for a nurse or helper when the patient requires attention. These asynchronous BMIs have two requirements; to be available (onset) to control the actuator at any given time as needed by the user and to recognize rest phase (offset) from the movement initiation in order for the user to control it effectively.

The first step in developing an asynchronous BMI is to differentiate initiation phase from the rest and movement phases. Several groups have done this using beta signals (13-30 Hz) of electroencephalography (EEG) [63, 64] and electrocorticography (ECoG) [65-68] signals. One method is to use the phase prior to movement onset, perform Fast-Fourier Transform (FFT) of each channel and use the averaged signal spectrum to classify states using a support vector machine (SVM); this process yielded an accuracy of 71.7% [67]. One of the challenges faced during decoding movement onset is the imbalance in positive and negative examples. A weighted SVM can been used to overcome this. Using an empirically determined threshold value the movement onset has been classified with true negative (specificity) rates 73%-94% while the true positive (sensitivity) rates are 26%-73% [66]. Another method of analyzing the ECoG is using wavelet transform (time-frequency representation of the ECoG signals for

each electrode) and predicting the movement onset using a least-square regression [65]. Least-squares regression was used with leave-one-out cross validation for training the decoder, and overall accuracy levels of 98-99% was reported [68]. However, it must be noted that the overall accuracy rates will be higher due to the imbalanced trial numbers. The group also reported that the false positive states occurred when the animal was resting [68].

Few groups have attempted to implement the asynchronous control. Linear discriminant analysis (LDA) is used to classify the states of EEG-based controllers. In this controller, the output is a posterior probabilistic distribution, which will give the probability that each trial belongs to each state (rest or move). The average accuracy of this system was 86.7% and 66.7% for 2 subjects [69]. A normalized low frequency asynchronous switch has been developed for EEG applications which incorporates different classification techniques; k-means, learning vector quantization, fuzzy adaptive resonance theory and Karhunen-Loeve transform. The group reported an overall accuracy level of 97% with 68% false positive rate [70]. During self-paced BCI control, the algorithm can classify 3 states: baseline, plan and go, for the state estimation with accuracies above 80% [71].

The basal ganglia have been shown to be involved in gating sensory information in the motor loop [72-75]. Abnormalities in the ventral striatum have been shown to affect limbic regions in sicknesses like the Tourette's syndrome where there is sensorimotor dysfunction [76]. We investigate the NAcc which is a

nuclei in the ventral striatum to assess if the gating signal can be obtained to control the flow of information in a BMI.

1.5 Specific Aims

1.5.1 Investigate role of Striatum during a Reaching Task and Test the feasibility of Using Striatal Signals as Feedback for a BMI

This aim was designed to investigate the feasibility of utilizing striatal signals as feedback reward/ error signals for a Reinforcement Learning based BMI. Microwire arrays were implanted each into the MI and the NAcc of a marmoset monkey. Animals were trained to control a robotic device by reaching to targets. Striatal neural activity was analyzed when the animal interacted with a robot during the reaching task. Studies were done to investigate reward/error representation and how it can be extracted from the striatum. It was also examined if movement onset can be extracted from the striatum during the reaching task. The results of this aim demonstrate the feasibility of using signals from the striatum as reward/ error signals and/or trial onset that can be used to design an intelligent closed-loop BMI control architecture.

1.5.2 Development of an Synchronous Closed-Loop BMI Control Algorithm

In this study, a biological feedback from the brain was used for a RL based BMI. Simulated and surrogate data sets were used to design the control architecture and test the feasibility.

1.5.3 Use the Striatum in Conjunction with MI to Control a Closed-Loop BMI

Once the closed loop BMI control paradigm was developed, it was tested on a marmoset monkey for a two choice robotic arm control task. The system was tested for the effects of reliability of feedback.

1.6 Outline of Dissertation

The motivation of the current work is to bring research a step closer in to adapting BMI for ADL. The current chapter gave the motivation and background of the BMI work at present. The second chapter is on experimental work for analyzing the striatal data for reward signal and trial onset. The third chapter focuses on computer simulations using synthetic data in order to test the hypothesis that if a reward signal can be extracted, using this information intelligently, can improve performance. In the fourth chapter we implement a closed-loop BMI experiment with a marmoset monkey and compare three scenarios: perfect Critic feedback, real Critic feedback with and without confidence measure. In the fifth and final chapter are the present challenges and future work in this area. During the analysis of the striatal data, an interesting phenomenon was discovered which is relevant to BMI field. We were able to cluster left and right robot movement with opposing natural hand movements. These findings are reported in the appendix.

Chapter 2 Experimental Data Analysis

This chapter explains the data acquisition methods including the surgical procedure, experimental paradigm and the data analysis of the experiments.

2.1 Data Acquisition and Surgical Procedure

In order to acquire signals from the brain required to drive the BMI, microelectrode arrays were implanted surgically in a common marmoset (*Callithrix jacchus*). Two neural signals were required to control the Actor-Critic RL paradigm; the Actor driven by the cortical structures (MI) and the Critic driven by the subcortical structures (NAcc). All surgical and animal care procedures were consistent with the National Research Council Guide for the Care and Use of Laboratory Animals and were approved by the University of Miami Institutional Animal Care and Use Committee.

2.1.1 Animal Model

The common marmoset is a small New World primate, native to the forests in Brazil. Historically the common marmoset has been used in neuroscience, reproductive biology, infectious disease, and behavioral research and more recently, in drug development and safety assessment [77]. Their size, availability, cost, husbandry, biosafety and unique biological characteristics may represent an alternative species to more traditional non-human primates (NHP) [77-79]. The average height of an adult marmoset is 20–30 cm and average weight is 350 – 400g [78]. Their endocrinological and behavioral similarity to humans have in addition drawn a lot of attraction in the field of neuroscience [78].

The marmoset has advantages over the macaques in terms of animal welfare and practicality. They are available for laboratory use from well-established captive colonies in national primate research centers, academic institutions, and commercial breeding facilities. Unlike macaques, marmosets do not carry herpes b virus (Macacine herpesvirus 1) [77], which is beneficial for their handlers. Their small relative size can also translate into lower caging and feeding costs and reduced floor space [78].

In addition, several marmoset brain atlas have been developed for neuroscience research [80-87] and extensions for stereotaxic equipment has been developed to make referencing easier for surgical procedures. In addition to all of these advantages, the smaller size of animal, leads to a more compact brain (8g on average) [88], and make it is easier to reach deep brain structures in comparison to larger NHP models which is one of the primary reasons we used this particular animal model.

2.1.2 Electrodes

Two microelectrode arrays were surgically implanted, each being a 16-channel tungsten microelectrode array as shown in Figure 2.1 A, (Tucker Davis Technologies, FL) with a differential reference and a ground wire. Each microwire electrode was blunt cut with Polyimide insulation and 50µm in diameter. The 16 electrodes were arranged in 2x8 configuration with row separation of 500µm and electrode separation of 250µm. For the final subject, the NAcc array was a 4x4 configuration with electrode separation of 250µm.

2.1.3 Surgical Procedure

For acquiring neural signals for arm reach, we targeted the hand and arm region of the primary motor cortex (MI). Reward information is represented in deep brain structures and therefore we implanted the second array in the striatum targeting the NAcc. To get the locations for each of the implants, we consulted different marmoset atlas [80-87] and selected the target locations as following with reference to the interaural (IA) plane. The implant for the MI was 9mm AP and 4.5mm ML relative to the IA center with an insertion depth of 1.8-2.0mm. The NAcc implant was 11.5mm AP and 2.3mm ML relative to IA center with a depth of 8.0mm. Figure 2.1 B shows the target locations with reference to the skull. Histological results show that in previous monkeys we were in the striatum. We will continue to refine targeting to get the best signals that could be used for a biological feedback. Since the two implants were close together, we used one craniotomy. Six screws were used for supporting the implants while one of these was used as the ground and reference point. Figure 2.1 C shows the depth for each implant.

The animal was anesthetized with ketamine (10 mg/kg IM) and aseptically prepped for surgery. Constant Rate Infusion (CRI) ketamine was used throughout the surgical procedure to maintain anesthesia at a rate of 10-15 mg/hr. The animal's head was shaved prior to midline incision, after which the skin was retracted to expose the skull and the surface cleaned with hydrogen peroxide. A craniotomy was made on the right hemisphere with the underlying dura opened as well. Electrical mapping stimulation was done to identify

arm/hand motor regions. An electrode-recording array targeting the primary motor cortex was implanted, with electrophysiology measurements being made during the implantation to ascertain placement. A second array was then implanted targeting the nucleus accumbens, again with electrophysiology measurements being made to ascertain position. The craniotomy was then sealed and the recording electrode arrays anchored to the skull (leaving the array connectors accessible) using genta cement and several anchoring screws. Prior to recovery, the animal was given 0.02 ml buprenorphine (0.3 mg/ml) IM.

2.1.4 Signal Processing

Neural recordings were collected using a Tucker Davis Technologies RZ2 system sampling at 24,414Hz and a band-pass filter 300-5000Hz. Local Field Potentials (LFPs) acquired with a 1-500Hz band-pass filter. Figure 2.1 D shows the spike interface during recording. A common average reference (CAR) was used to improve signal quality. The CAR was set up in such a way that if the signal to noise ratio (SNR) needed to be further improved, any of the electrodes could be removed from the CAR. This was necessary as the SNR varies with the electrode longevity and encapsulation [89-91]. Figure 2.1 D(a) is the signal after band-pass for one channel while Figure 2.1 D(b) shows the same for 16 channels. Figure 2.1 D(c) shows the 16 channels after CAR.

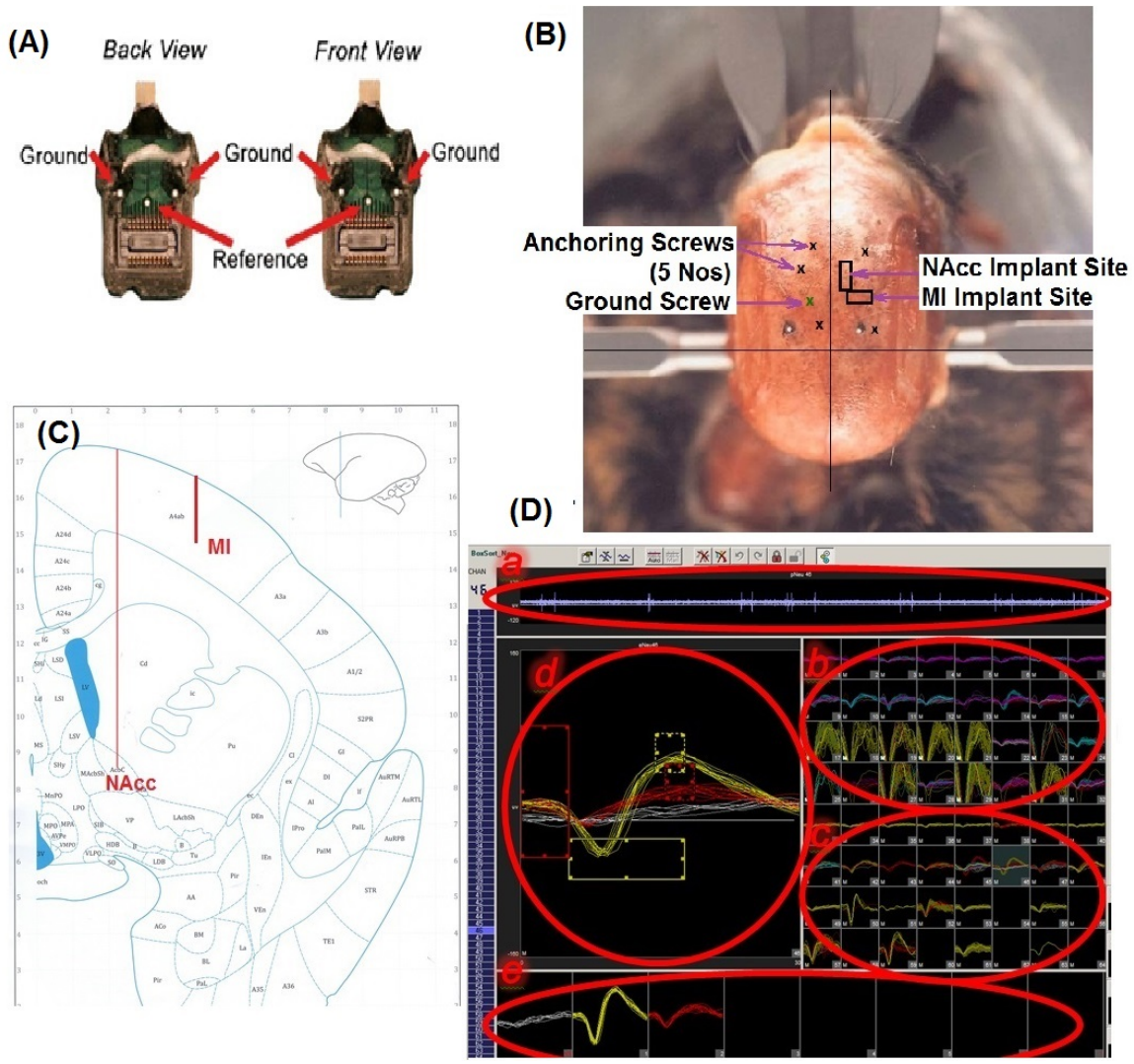

Figure 2.1: (A) – Microelectrode Arrays. (B) – Target Locations with reference to the skull. (C) - Target Depths. (D) – Recording Interface. (a) Filtered Raw Signal for each channel. (b) Snippets after threshold for each channel. (c) CAR of each channel. (d) Spikes for one channel with two sorts – red and yellow. Background activity given no sort is shown in grey. (e) Sorted Spikes and background activity.

Isolating of spikes was done in real-time using standard spike sorting methods [92]. These were based on the shape and amplitude of action potential waveforms. A manual threshold level was set for each channel through visual inspection. Figure 2.1 D(d) and Figure 2.1 D(e) show the spike sorting for one channel. There is a larger amplitude waveform given a yellow sort and a smaller amplitude waveform given a red sort and background activity given no sort

(grey). Both multiunit and single unit neurons were recorded and used equivalently in all applications. Multiunit activity and single unit activity (SUA) collectively are referred to as neuronal signals. These neuronal signals (RAW or CAR) are used as input to the decoders [93, 94].

2.2 **Experimental Design**

To use the NAcc in BMI context, we needed first gauge how the NAcc behaves during a robotic task. Neural data was analyzed when an animal is engaged with a robotic task. Two tasks (Go-No-Go and two-target reach tasks) were designed and the data analyzed with the intention of extracting the reward/error representation. 3 subjects were used for the 2 different experimental paradigms. Subject 1 (“Princeton”) was only for Go-No-Go task, while subject 2 (“Duke”) was trained for both tasks and subject 3 (“Don”) was only trained for the two-target reach task.

2.2.1 Go-No-Go Paradigm

We designed a two-choice decision making task which included perturbations to test if robot incorrect (non-rewarding) actions would create a different neural response from the robot correct (rewarding) actions. The task was a simple go-no-go as shown in Figure 2.3 (A) and described below. The task was designed to investigate the representation of the NAcc during the robot movement. The animal was trained to move a robot arm to one of two targets to receive a food reward. The experimental timeline is shown in Figure 2.3 (B). The animal initiated trials by placing its hand on a touchpad for a random (0.7-1.2 seconds) hold period. At the onset of the trial, an audio go signal was

administered that corresponded to a robot arm moving upwards, out from behind an opaque shield, and presenting its gripper. The gripper held either a desirable (waxworm or marshmallow, ‘A’ trials) or undesirable (wooden bead, ‘B’ trials) object. Simultaneously, the A (red) or B (green) spatial target LED corresponding to the type of object in the gripper was illuminated [95].

Depending on the type of trial, the animal was required to respond within a time limit. Each type of trial required a different action; for A trials, the monkey had to reach a second sensor within 2 second reach time limit and the robot would move to A target; for B trials, it was required to keep its hand motionless on the touchpad for 2.5 seconds and the robot would move to B target. It was necessary that the time for B trials (no move) were slightly longer than for A trials (move) so that the animal was forced to understand the difference between the two types of trials and respond accordingly. The robot arm would move to the location indicated by the animal’s response. For both A and B trials, if the robot moved to the target indicated by the LED, the monkey was given a food reward. The actions and the category of trials are given in Table 2-1. Trials where the animal either did the wrong action or was not interacting with the task were removed from the analysis [95].

To create robot perturbations that contrast with reward trials, the robot was occasionally overridden and moved in the direction opposite to that of the action commanded by the monkey. These trials where the monkey sees an undesirable action in the environment (evoking negative response in the brain) were considered “catch” trials. From the animal’s perspective, the catch trials are

those in which the robot moved in the wrong direction, even though he performed the action correctly. The percentage of catch trials varied with the animal's behavior. The trials where the robot moved to the intended target and the animal received a food reward were called "standard" trials (Table 2-2).

Table 2-1: Trial Type And Different Actions Of The Monkey. (Success And Failure Trials)

Category	Trial Type	Object on Gripper	Required Action	Time Limit	Monkey's Action	Robot Action	Reward
A success	A	Desirable (waxworm or marshmallow)	Touch & trigger reach sensor	2 sec	Reaches & triggers reach sensor	Move to target A (left)	Receive Treat on Gripper
A failure					Does not trigger reach sensor	Move to target B (right)	No Reward
B success	B	Undesirable (wooden Bead)	Keep hand in touch pad	2.5 sec	Keeps hand in touch pad	Move to target B (right)	Receive Food Treat
B failure					Takes hand out of touch pad	Move to target A (left)	No Reward

Table 2-2: Robot Action For Different Types Of Trials. (Standard And Catch Trials)

Category	Trial Type	Object on Gripper	Required Action	Time Limit	Monkey's Action	Robot Action	Reward
A standard	A	Desirable (Food Treat)	Touch & trigger reach sensor	2 sec	Reaches & triggers reach sensor	Move to target A (left)	Receive Treat on Gripper
A catch						Move to target B (right)	No Reward
B standard	B	Undesirable (wooden Bead)	Keep hand in touch pad	2.5 sec	Keeps hand in touch pad	Move to target B (right)	Receive Food Treat
B catch						Move to target A (left)	No Reward

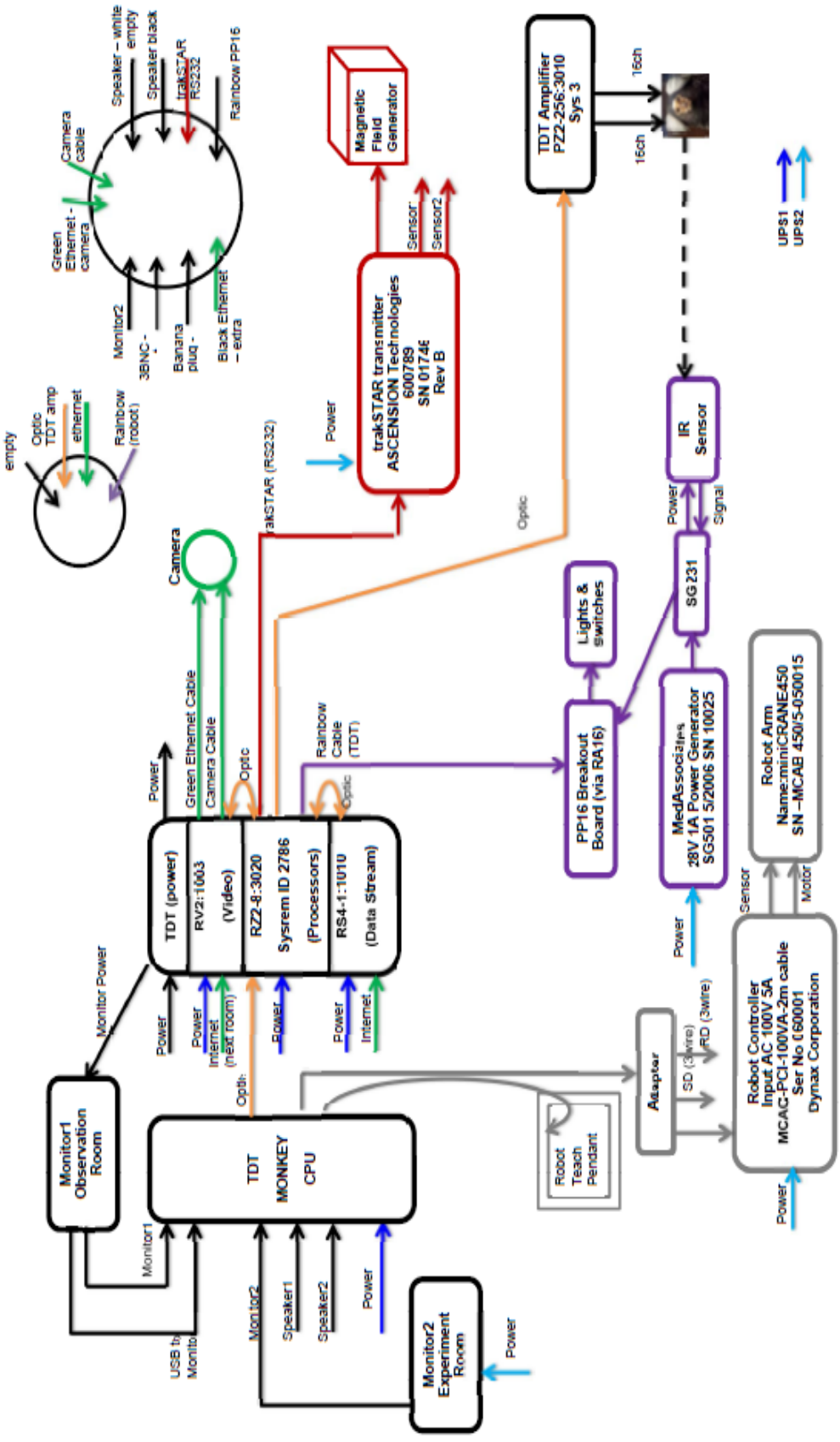

Figure 2.2: Experiment Setup with the data acquisition system.
Black: Central Processing Unit and TDT System for data processing and streaming. Grey: Robot Controller and Robot related hardware.
Purple: Sensors and sensor related hardware. Red: Arm motion tracking system. Two circles on right top indicate the physical communication pathways between the two rooms

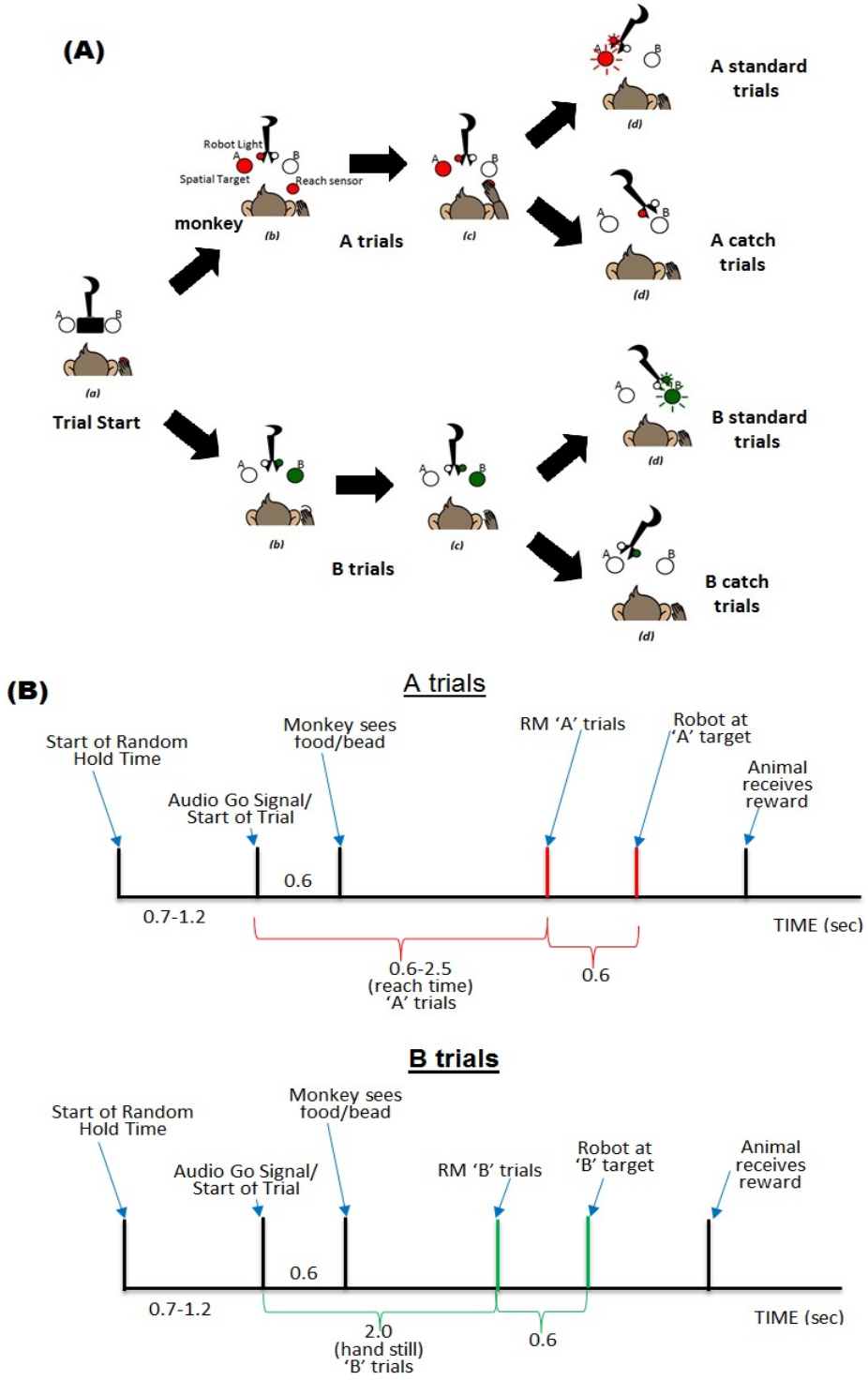

Figure 2.3: (A) Experiment Setup. (a) Trial Start – Animal triggers trial (b) Robot moves out from opaque shield, target A/B lights up (c) Animal makes arm movement to reach sensor for A trials/ keep hand still for B trials (d) Robot moves to correct target (standard trials) or incorrect target (catch trials). (B) Time line for the trials. TOP: A trials. BOTTOM: B trials

2.2.2 Two-Target Reach Paradigm

The paradigm described in above (go-no-go) had two types of trials with two different reach states (reach/non-reach), two time limits (2sec/ 2.5sec), two light colors (red/ green) and two things on the robot gripper (desirable and undesirable objects). The next experiment was designed to reduce some of these variables and focus on the robot movement. A trial was initiated similar to the go-no-go experiment when the animal placed his hand on a touchpad for a random (0.7-1.2 seconds) hold period, at the end of which the audio go cue was given with the robot arm moving upwards from behind an opaque shield and presenting its gripper which held a food reward. Since the robot always held a food treat (waxworm, mushroom or marshmellow) and it had no informative value, thus controlling that variable.

Simultaneously, to the robot arm moving upwards, a spatial target for the robot and an IR reach sensor for the animal were both illuminated. Both of these targets were on the animal's left ('A' trials), and the animal had been trained to touch the left IR sensor in order to move the robot to the left spatial robot target. Similarly, during 'C' trials, the animal would move the robot to a spatial robot target on its right, by reaching and touching an IR sensor that was also on the animal's right. For each type of trial, the animal had 2 seconds to make an arm movement, and there were additional LEDs mounted on the left and right of the robotic gripper which illuminated in parallel with the robot spatial target lights (Figure 2.4 A and Figure 2.4 B). If he reached for the IR sensor that was

illuminated the robot similarly moved in the correct direction and the animal received the food reward at the end of the robot gripper (standard trials). Those trials where the animal either did the wrong action or was not interacting with the task were removed from the analysis [96].

To ensure that the monkey attended to the robot arm movements, occasionally (varied between 20%-40%), the robot was overridden to go to the wrong target (and thus the monkey received no reward). These “catch” trials, controlled for the effects of the monkey’s physical arm movements which may have otherwise skewed the results; for example, ‘A’ catch trials corresponded to a left physical arm movement, but the robot moved to the right and similarly, ‘C’ catch trials had a right arm movement, but the robot moved to the left. During catch trials the monkey’s behavior indicated that he was aware of the robot moving in the wrong direction. Catch trials resulted in non-rewarding instances which the animals did not like. Therefore, the catch trials percentage was kept low so as to keep the monkey engaged in the task [96].

For analysis, neural data during the hold time, before and during the arm movements as well as during the robot movement were considered. The experimental timeline for the two-target reach task is given in Figure 2.4 C.

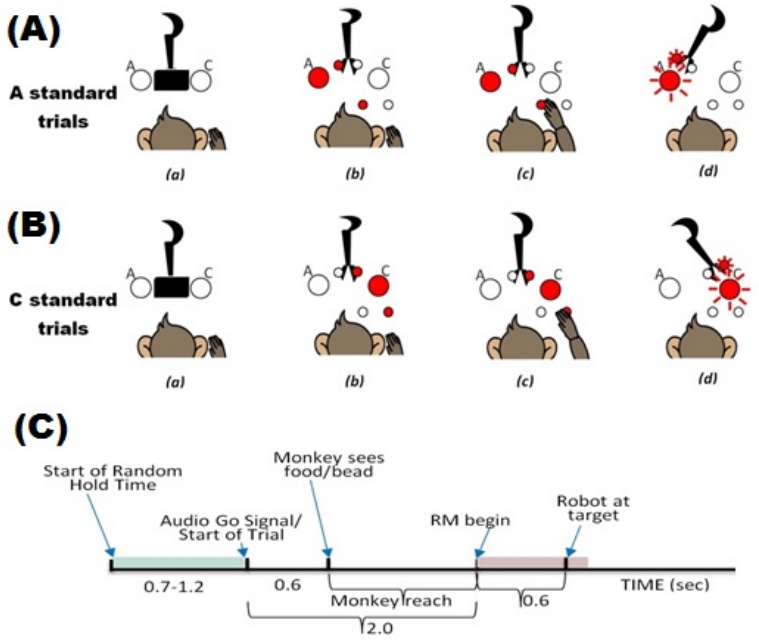

Figure 2.4: Standard Trials. (A): A trials: (a) Animal triggers trial (b) Robot moves out from opaque shield, target A lights up (c) Animal makes arm movement to left reach sensor (d) Robot moves to target A. (B): C trials: (a) Animal triggers trial (b) Robot moves out from opaque shield, target C lights up (c) Animal makes arm movement to right reach sensor (d) Robot moves to target C. (C): 'Timeline for trials in black. Hold time shown in green and RM shown in red. RM = Robot Movement

2.2.3 Experiment Variable Summary

The summary of the variables during the animal's hand reach (time of trial start, Table 2-3) and during the time of the robot movement (Table 2-4) are given in the tables below. As seen, two confounding variables (treat type and color of light) have been controlled in the two-target reach task.

Table 2-3: Summary Chart Of Experiment Variables At The Time Of The Trial Start

Experiment Name	Go-No-Go Task		Two-Target Reach Task	
Trial	A	B	A	C
Treat	Food	Bead	Food	Food
Color	Red	Green	Red	Red
Spatial Target Location	Left	Right	Left	Right
Monkey Action	Reach	Still	Reach left	Reach Right
Correct Robot Movement	Left	Right	Left	Right

Table 2-4 Summary Chart Of Experiment Variables At The Time Of Robot Movement

Experiment Name	Go-No-Go Task		Two-Target Reach Task	
Type	Standard	Catch	Standard	Catch
Treat	Food/ Bead	Food/ Bead	Food	Food
Color	Red/ Green	Red/ Green	Red	Red
Spatial Target Location	Left/ Right	Left/ Right	Left/ Right	Left/ Right
Monkey Action	Reach/ Still	Reach/ Still	Reach	Reach
Robot Movement	Correct	Wrong	Correct	Wrong
Lights	Flash	Off	Flash	Off

2.3 **Data Analysis and Results**

To use NAcc neuronal activity for closed-loop control of a robotic arm, we need to first understand how the NAcc behaves while the animal is engaged in a robotic task. This section describes the analysis techniques to understand how the NAcc projects to a prosthetic limb or a robotic arm and how to extract reward information from the NAcc. We present histograms and data reduction methods first. There are many methods we can use to classify our data; both supervised and unsupervised methods can be used with advantages and disadvantages for both. Results from both experiments (Go-No-Go – red/green and two-target reach – red/blue) are presented in this section. The two main questions are: how does the animal perceive the robot and how is the neuronal firing affected by perturbations in the environment.

2.3.1 Neural Firing Patterns and Histograms

Next how the robot movement was captured in the neural firing pattern was studied using histograms. For this, the time window during the robot movement was analyzed. There were neurons that showed difference in modulation for the movement of robot (either in the correct or incorrect direction). Figure 2.5 (A) shows the neural modulation during the time the robot is moving for one unit during the Go-No-Go task. For A standard trials (red), the activity increases as it approaches 0.5 seconds (see Figure 2.5 (A)), but for A catch (black), the activity decreases in the same time period. For B trials, we see the opposite effect; the activity increases for B catch (black) trials and decreases for B standard (green) trials.

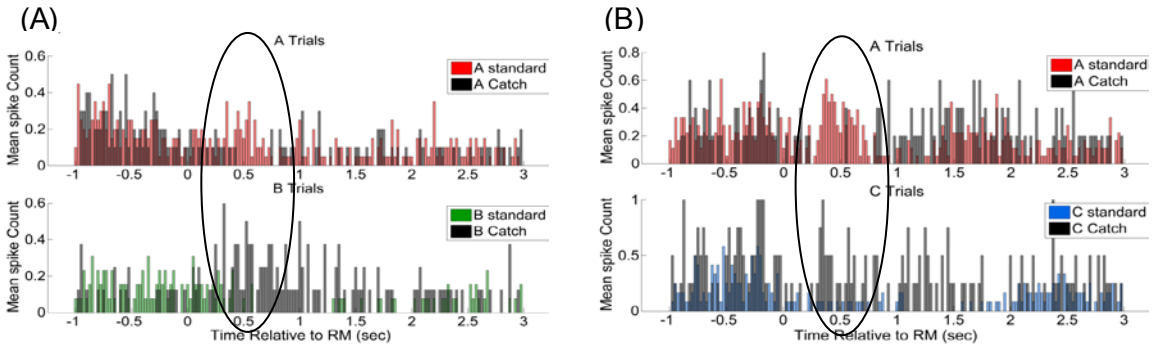

Figure 2.5: Mean Spike Count for standard and catch trials with relative to the Robot Movement (RM).
Window of interest 0.2-0.7sec after the RM. (A) Go-No-Go Task. Standard: A (red), B (green). Catch: black. A catch and B standard trials show inhibition. (B) Two-Target Reach Task. Standard: A (red), C (blue). Catch: black. A catch and C standard trials show inhibition.

Table 2-5 gives the number of significant units (t-test, alpha = 0.1) for each task broken down by the different categories. There are a larger number of units that fire differently for left vs right. The total window length was 0.7 sec which was arrived at using the animal's reach time, robot movement time and the best accuracy for the classifying the two different classes.

Table 2-5: Number Of Significant Units For Each Type During Robot Movement (0.7sec Of Data) Alpha = 0.1

Go-no-go Task (Trial Type A/B). Total units = 29						
Session	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
A stand/catch	8	4	9	8	6	9
B stand/catch	7	6	7	5	12	6
stand/catch	6	6	5	3	5	6
left/right	14	8	8	8	10	10
2 target Task (Trial Type A/C). Total units = 27						
Session	**S1**	**S2**	**S3**	**S4**	**S5**	
A stand/catch	2	5	1	4	6	
C stand/catch	3	5	6	3	5	
stand/catch	1	1	4	3	3	
left/right	5	8	4	5	7	

Here we analyzed SUA during the time when the animal is observing the robot movement. There is a statistically significant difference during the robot movement; high number of units is significantly different (t-test, alpha = 0.1) for left and right robot movements. However, there are a few units that are significantly different for rewarding vs non-rewarding trials.

2.3.2 Neural Population Dynamics – Principal Component Analysis (PCA)

While it is important to study the firing patterns of individual neurons, in the context of brain machine interfaces, we need to study populations of neurons. There is redundancy in the brain in case a few neurons stop functioning we are still able to capture the information by using the population. We are able to capture multiple information from a population of neurons that we cannot from single neurons. Individual neurons can code single information. However, when considering the population of neurons together, the population may code something different from the individual neuron. For example, this has been

shown in neuronal population coding related to movement [25, 97]. Therefore it is important that we study the neural population as a whole.

When considering the neural population, one of the challenges is to extract the appropriate features for the application. A key characteristic in the neural modulation is variance. We used principal component analysis (PCA) as the method to convert the data into a low-dimensional space for both ranking the relative importance of the neurons as well as visualizing the data. PCA has been widely used for spike sorting and dimensionality reduction of neural data [98, 99]. PCA also gives the direction of maximal variance, which helps in extracting relevant features and in dimensionality reduction, which is helpful in BMI applications. For all the sessions analyzed, the first 9 principal components (PCs) accounted for at least 80% of the variance, while the first 15 PCs accounted for at least 90%. The first two PCs contained 48% of the variance and showed best separability. Hence, the first 2 PCs were selected as the features for analysis. The separability of the data was inspected in two dimensional space with the first two PCs. A visual separation between standard (‘+’) and catch (‘o’) trials was seen when each type of trial was analyzed. The results from all the sessions analyzed concluded that once the trial type information was identified, a separation can be seen between standard and catch trials in the PC space. Further, when the trial types were combined, a left vs right separation was visible.

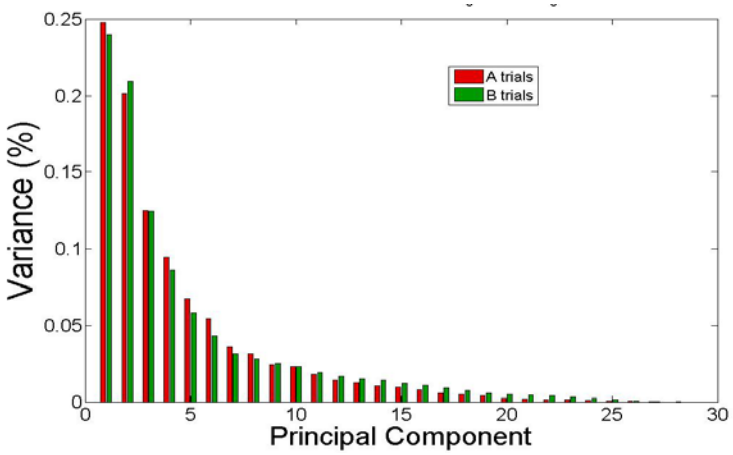

Figure 2.6: Variance of data relative to RM. Red: 'A' trials . Green: 'B' trials. Window 0.2-0.7sec (S1+S2+S3)

The analysis done was using a 0.5 second sliding window (0.1 second overlap to capture any temporal features) with the sum of firing rate within the given window of each of the 29 neuronal signals as the feature space. This goal was to find the optimal window that correlated with the robot moving to or away from the desired target.

2.3.3 Unsupervised Clustering

In unsupervised clustering, the data will be separated according to the similarity within a cluster and dissimilarity from the other cluster. An advantage in our application is that the number of clusters is known to be 2. We can assign label +1/-1 for the clusters obtained. In supervised classification, the label must be known *a priori* and the decoder needs to be trained with a sufficient amount of examples for good performance and robustness.

After features were extracted using PCA, we used a simple unsupervised methods (k-means) to cluster the data and compared those clusters with the class labels we required and calculated the accuracy. k-means is used to classify

n objects of input space $I\{i_1 i_2 \dots i_n\}$, each having measurements on p variables $i_j\{x_{j1}x_{j2} \dots x_{jp}\}$, into k clusters with cluster centroid $C\ (c_1 c_2 \dots c_k)$. In this case, $n = number\ of\ trials, p = number\ of\ principal\ components\ and\ k = 2$.

The algorithm was set to start by setting C to an initial value (randomly picked from I). The centroid value for cluster c_k is given by:

$$c_k = \frac{1}{n_k}\sum_{j=1}^{n_k} i_j;\ \forall\ i_j\{x_{j1}x_{j2} \dots x_{jp}\}\epsilon c_k \qquad \text{Eq. 2.1}$$

where n_k is the number of objects in k.

Next, clustering is done based on minimizing the cost function which is a measure of the distance between each data point and the centroid. Three different cost functions were used: squared Euclidean distance, sum of absolute differences and one minus the cosine of the included angle between points (treated as vectors). The results are of squared Euclidean distance are presented as the clusters aligned better with this criterion. However, there is much room for improvement as shown in figures below.

For each $i_j \epsilon\ I$, the squared Euclidean distance (d) between i_j and its centroid, c_k was calculated.

$$\mathrm{d}(\mathrm{i_j}, \mathrm{c_k}) = (\mathrm{i_j} - \mathrm{c_k})^2;\ \ \forall\ \mathrm{i_j}\ \{\mathrm{x_{j1}x_{j2}} \dots \mathrm{x_{jp}}\}\ \epsilon \mathrm{c_k}, \mathrm{j} = 1,2 \dots \mathrm{n} \qquad \text{Eq. 2.2}$$

The objects of I were moved to the cluster whose centroid was closest, until d was minimum [100].

$$\underset{C}{\operatorname{argmin}}\left(\sum_{j=1}^{k} d(i_j, c_k)\right) \quad \text{Eq. 2.3}$$

The two clusters obtained from k-means clustering were assigned labels (standard and catch) manually and compared against the class labels standard ('+') and catch ('o') categories in the experiment. The manual labeling of the clusters was always done to maximize the resulting classification accuracy. The classification accuracy was the number of trials correctly classified (True Positive + True Negative) out of the total number of trials.

For A trials in the go-no-go task (Figure 2.7A), the clusters given by k-means (blue/yellow) do not overlap accurately with the '+' and 'o' clusters in the experiment, however, for B trials (Figure 2.7B), the clusters given by k-means coincide better than the A trials. Accuracy can further be improved by better clustering as there is one misclassified 'o' trial in the blue cluster. For the two-target reach task, in A trials (Figure 2.7C), the clusters given by k-means are a good representation of the '+' and 'o' classes; there is one outlier 'o' in the yellow cluster. For C trials (Figure 2.7D), the clustering accuracy can further be improved with better separation of clusters as there is one misclassified 'o' trial in the green cluster [95].

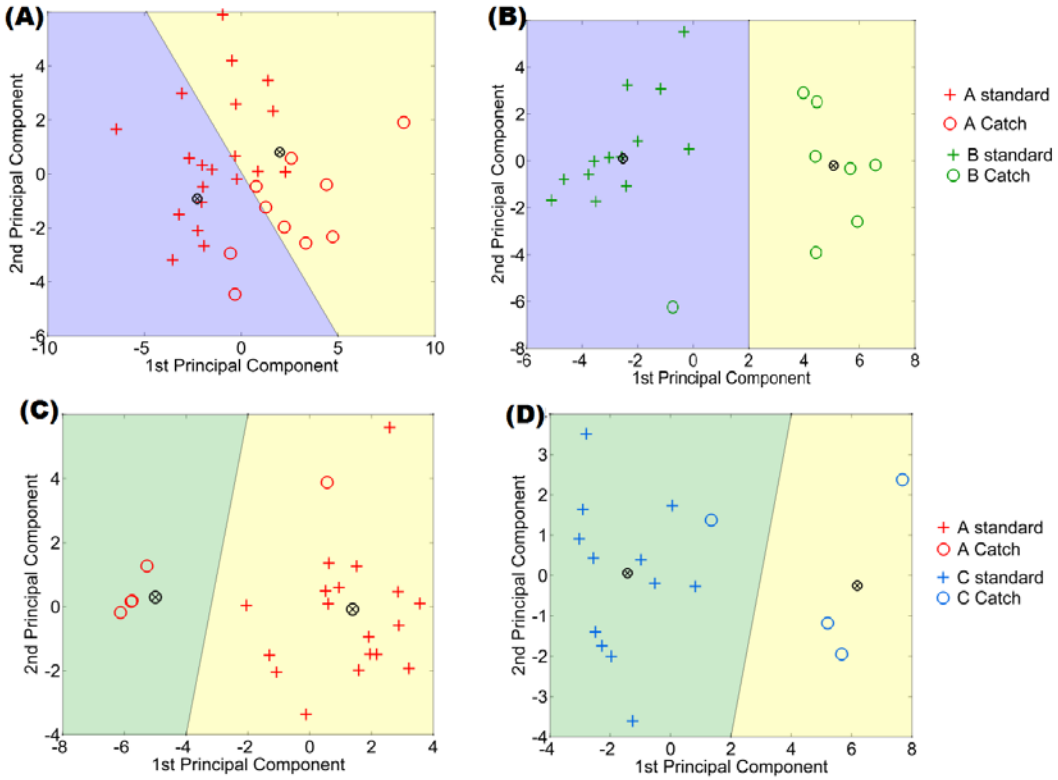

Figure 2.7: Data clustered in PC space using k-means. Blue/ Green: Cluster 1. Yellow: Cluster 2. '+': standard. 'o': catch and ⊗: cluster centers. Window 0.2-0.7sec.
(A) A trials (go-no-go task). (B) B trials (go-no-go task). (C) A trials (two-target task). (D) C trials (two-target task).

Next we combined the trial types for each experiment and recalculated the PC space and clustered using k-means as before. Figure 2.8 shows how the k-means clustering partitioned the space based on the minimizing the Euclidean distance. There are a few outliers (two points in (A) one point in (B)) that may have caused the clustering to be skewed from what the ideal situation is.

In Figure 2.8A, the blue cluster includes both B standard and A catch trials while the yellow cluster show A standard and B catch trials. The blue cluster represents right robot movement and the yellow cluster represents left robot movement. In Figure 2.8B, the green cluster includes both C standard and A catch trials while the yellow cluster show A standard and C catch trials. The

green cluster represents right robot movement and the yellow cluster represents left robot movement. There is one misclassified A standard trial (red '+') in the green cluster.

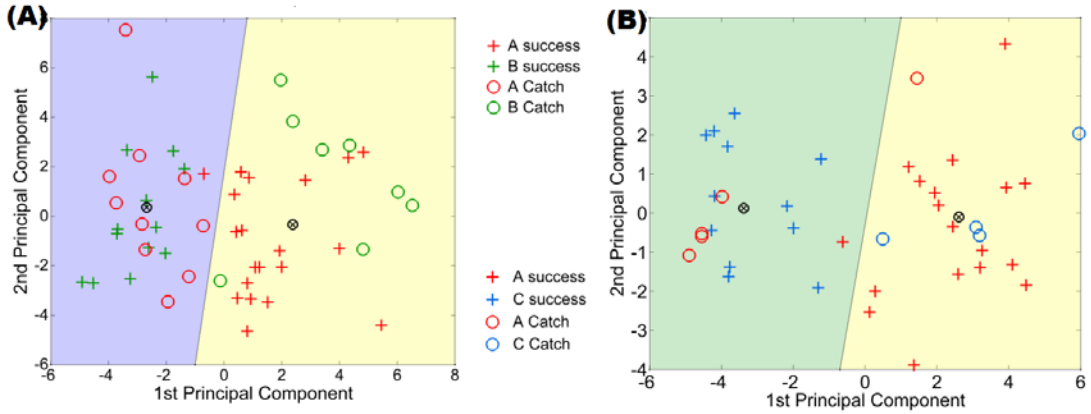

Figure 2.8: Data from 1 session. '+' standard trials. 'o' catch trials.
(A) Blue: Cluster 1. Yellow: Cluster 2 and ⊗: cluster centers. Red – A trials. Green – B trials. The blue cluster represents right robot movement and the yellow cluster represents left robot movement, with different reward (standard/catch) and natural arm movement. (B) Green: Cluster 1. Yellow: Cluster 2 and ⊗: cluster centers. Red – A trials. Blue – C trials. The green cluster represents right robot movement and the yellow cluster represents left robot movement, with different reward (standard/catch) and natural arm movement.

Table 2-6 shows the clustering accuracy for each experiment when clusters were aligned with standard and catch as well as left and right robot movement. The clustering accuracy is higher with left vs right, but the accuracy is at chance level for standard vs catch.

Table 2-6: Accuracy Percentages When Aligning The K-Means Clusters With The Different Categories (Window 0.2-0.7sec Relative To Robot Movement) Two-Target Reach Task

Session	S1	S2	S3	S4	S5	Average
Standard vs Catch	59%	54%	60%	50%	57%	56%
Left Vs Right	85%	90%	56%	89%	89%	82%

We are able to conclude that robot movement direction (left/right) is well represented in the data and can be extracted with simple unsupervised clustering techniques for 2 subjects. Subject 1 ("Princeton") data was only from go-no-go task, subject 3 ("Don") data was only from two-target reach task and subject 2

("Duke") data were from both tasks. For further mathematical representation of the robot movement direction see Appendix A. However, for our application in RLBMI, we need a reward/error signal from the brain. The unsupervised methods perform at chance and are not suitable for extracting reward information. Since the histograms indicate there is some reward information, we move on to supervised techniques to extract this reward information.

2.3.4 Supervised Classification

In supervised classification techniques, the classifier needs to be trained by known data and the corresponding class labels. Traditionally, hundreds of data points are required to build a classifier. However, in our application, we only have a few trials for the entire session and even combining several sessions across days can give us 200+ trials.

Classifiers Used

Four methods have been used for the supervised classification in this section. These classifiers were based on advantages each method had to offer and the data that was collected. First is a Support Vector Machine (SVM). SVM is a discriminative classifier formally defined by a separating hyperplane between the two classes. The algorithm outputs an optimal hyperplane such that the margin of the training data is maximized. For data such as above, it is possible that in higher dimensional space, there exists such a hyperplane [101, 102].

Next a k-Nearest Neighbor (k-NN) algorithm used is a non-parametric method where the input consists of the k closest training examples in

the feature space. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If the data at hand is not separable even in higher dimensional space, this method is advantageous since it exploits the features of the neighboring data points. However, if this is to be used in real time, the computational complexity is a factor that needs consideration [103, 104].

Next method is a naive Bayes classifier, which is a simple probabilistic classifier based on Bayes' theorem with strong (naïve) independence assumption between the features. This classifier was implemented as it uses the distribution of the data classes. The data which was explored had multiple information encoded and an added advantage of Naïve Bayes is, it is not sensitive to irrelevant features [105, 106].

The last classifier is a tree-bagging algorithm, also known as random forests. These operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. The predicted class is decided by the probabilistic outcome. This classifier is similar to k-NN but the decision was made to implement this since the predicted class has a probability associated with it [107, 108].

For each of the classifiers above, training and testing data were balanced. This is important so as to not have a biased classifier.

Data Sets

Two marmoset monkeys were used for the two-target reach task: “Duke” (subject 2) and “Don” (subject 3). For Duke 11 sessions across 9 days were combined for a total of 468 trials (52, 58, 48, 32, 30, 41, 39, 41, 43, 38 and 46 trials in each session). For Don, 3 data sets across 5 days were combined for a total of 220 trials (48, 88 and 84 trials in each session). Table 2-7 shows the distribution of trials in both data sets.

Table 2-7: Distribution Of Trials In The Data Sets Analyzed

	Duke	Don
Total Sessions	11	3
Total Trials	468	220
MI units per day	10	11
NAcc units per day	27	28
Type A trials	250	109
Type C trials	218	111
Rewarding Trials	360	164
Non-Rewarding Trials	108	56

Different time windows were analyzed for both NAcc data and MI data. For MI data, the time window was with respect to (wrt) the start of the “Go Tone”. Both 500 msec and 1000 msec bins were analyzed. For NAcc data, the windows were wrt to the start of the robot movement (RM). 90% of data (balanced) were used for training and 10% (balanced) for testing. The reason for using balanced classes for testing was so that the accuracy given will be a true representation of classification, and due to classification of all trials as one type. Hence, randomly selected equal number of examples from both classes were given and 100 simulations were performed.

NAcc Data

This section analyses the success vs catch difference in the monkeys. Success trials were given a label '+1' suggesting rewarding action by the robot and catch trials were given a label '-1' indicating a non-rewarding action by the robot. All trials are synchronized to the start of the robot movement, which is the first indication of the correct or incorrect robot movement. However, depending on the training level of the animal, the animal may or may not grasp this immediately.

Figure 2.9 shows the accuracy for Duke (A/B) and Don (C/D) data from 100 simulations in classifying success vs catch trials from NAcc for different window sizes and sliding windows, for the four classifier types discussed above. Three of the classifiers performed poorly on Don's data on average (Refer to the figure below which show the standard deviation along with the average). Subplots (A) and (C) show 500 msec window size and subplots (B) and (D) show 1000 msec window size. Each colored trace gives a different classifier type. The accuracy with Naïve Bayes (green) classifier is low in general for all the windows and both sets of data. Similarly Random Forests (blue) tend to do better in general for most of the windows analyzed in both monkeys. For Duke, a later time window (500 msec or after) gives better results, while for Don, an earlier time window (500 msec or before) gives better results. This difference could be due to different animals perceiving the task differently. At the time of data collection, Don was more trained than Duke, and therefore recognized catch

trials at the start of the robot movement, whereas Duke took a few hundreds of milliseconds to realize this.

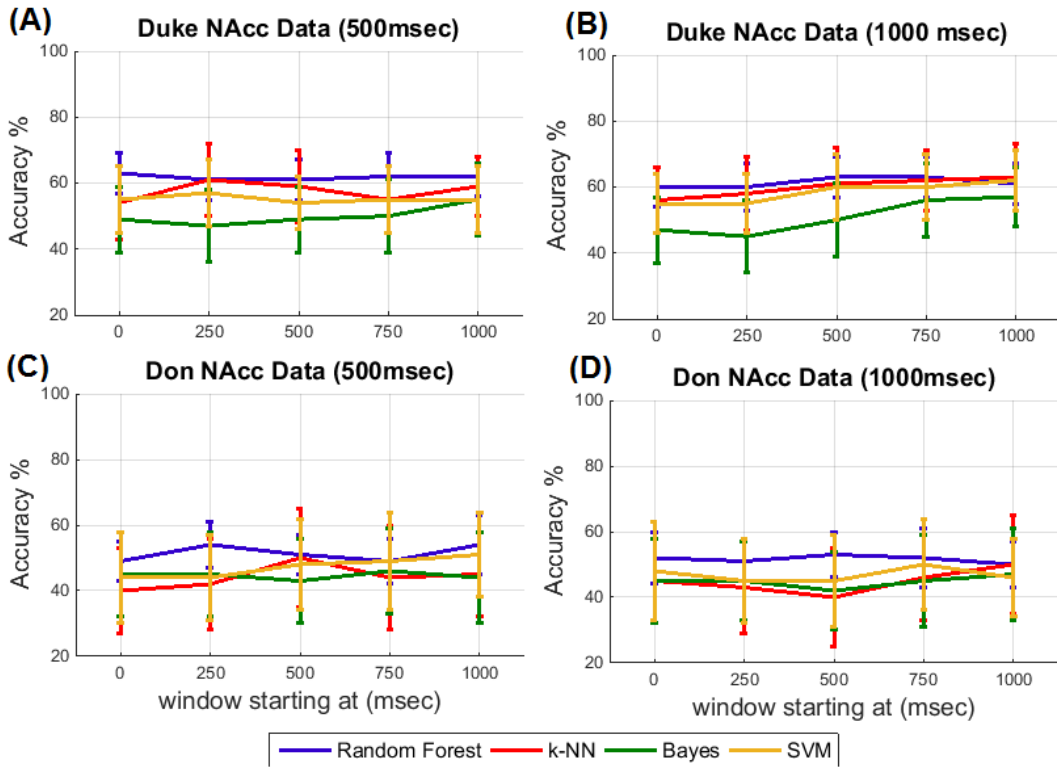

Figure 2.9: Classification accuracy (average accuracy of 100 simulation) for NAcc data success vs catch for Duke (A/B) and Don (C/D). 500msec bins (A/C) and 1000msec bins (B/D). x axis shows the start of the window wrt RM and y axis shows the accuracy percentage. Each colored trace shows a different classification method. (blue – RF, red – kNN, green – Naïve Bayes, Orange – SVM). Chance 50%

MI Data

Next we studied how well supervised classifiers are able to classify left and right hand movement of the animals. This was done by giving a class label of '1' for A trials (left movement) and a class label of '2' for C trials (right movement). Analysis was done for both 500msec bins and 1000msec bins. Figure 2.10 shows the classification accuracies for classifying left arm movement and right arm movement from MI data for Duke (A/B) and Don (C/D). For Duke, Random Forests perform better than the other three classifiers, and the SVM

performs the lowest (at chance). For Don, none of the classifiers outperform each other; all of the traces are between 40%-60%. When comparing the time window to use for a BMI, for Duke, a wider window (1000 msec – subplot B) with minimum 500 msec delay yields the best results. For Don, this is harder to interpret since the overall accuracy is low. However, in the case of Don, the best time delay is 500 msec; any longer time delays, reduce the accuracy. The 1000 msec bin performs slightly better than the 500 msec bin and we conclude to use a 1000 msec bin for the BMI experiments for Don. As mentioned before, since this animal was more trained than Duke, his reaction times were faster, and the reach times were lower.

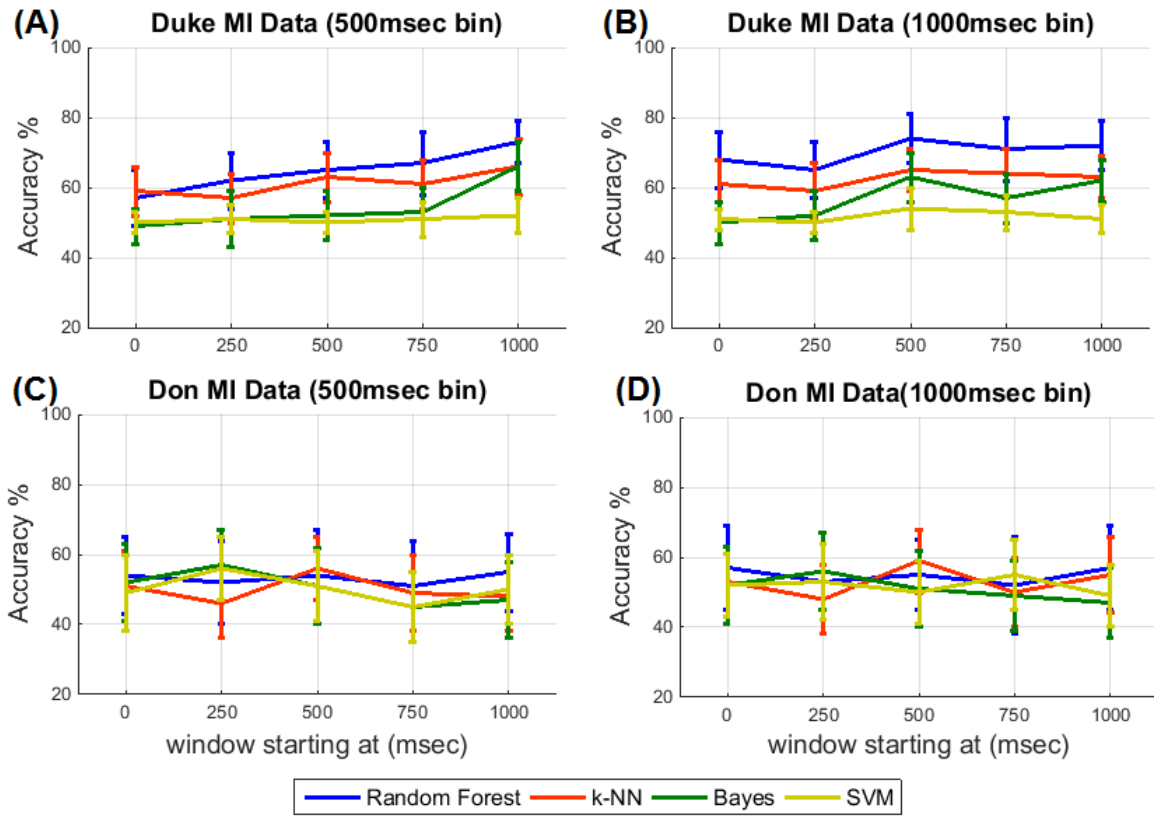

Figure 2.10: Classification accuracy (average accuracy of 100 simulation) for MI data left vs right arm movement for Duke (A/B) and Don (C/D).
500msec bins (A/C) and 1000msec bins (B/D).

x axis shows the start of the window wrt RM and y axis shows the accuracy percentage. Each colored trace shows a different classification method. (blue – RF, red – kNN, green – Naïve Bayes, Orange – SVM). Chance 50%

2.4 Trial Initiation from the Striatum

This section gives results of extracting trial initiation from the striatum data. The hypothesis is that the NAcc has trial initiation information. If this information can be reliably extracted, an architecture can be built to exploit this information in an asynchronous manner.

2.4.1 Filter Design and Preprocessing

LFPs from 15 channels were used for one monkey from one session. The LFPs were acquired at 2034.5Hz and down sampled to 1017.25Hz before preprocessing. Five band-pass filters were designed for the following frequency bands shown in Table 2-8. The frequency response of the filters are shown in the figure below. Once the signals were filtered for the respective bands, it was smoothed with a 100msec window with 50% overlap.

Table 2-8: Frequency Bands For Lfps

(1)	Delta 1-4 Hz
(2)	Theta 4-8 Hz
(3)	Alpha 8-13 Hz
(4)	Beta 13-30 Hz
(5)	Classical Gamma 30-60 Hz

The labels for the data were based on the start of the trial, where the animal's hand was stationary inside the touch pad for 0.7-1.2 sec. After this the robot came up from behind the shield – which took approximately 0.6 sec to complete. The animal's reaction time/ time of beginning of reach were as early as 0.5 sec and as late as 1.5 sec (average 0.8 sec). The earliest time was used at

0.5 sec. For each trial, 0-0.5 sec from the trial start was given a label of 1 and everywhere else was given a label of 0. The labels were also down sampled to the same frequency as the data above. With the filtered and smoothened data, 250 msec was used to predict one label

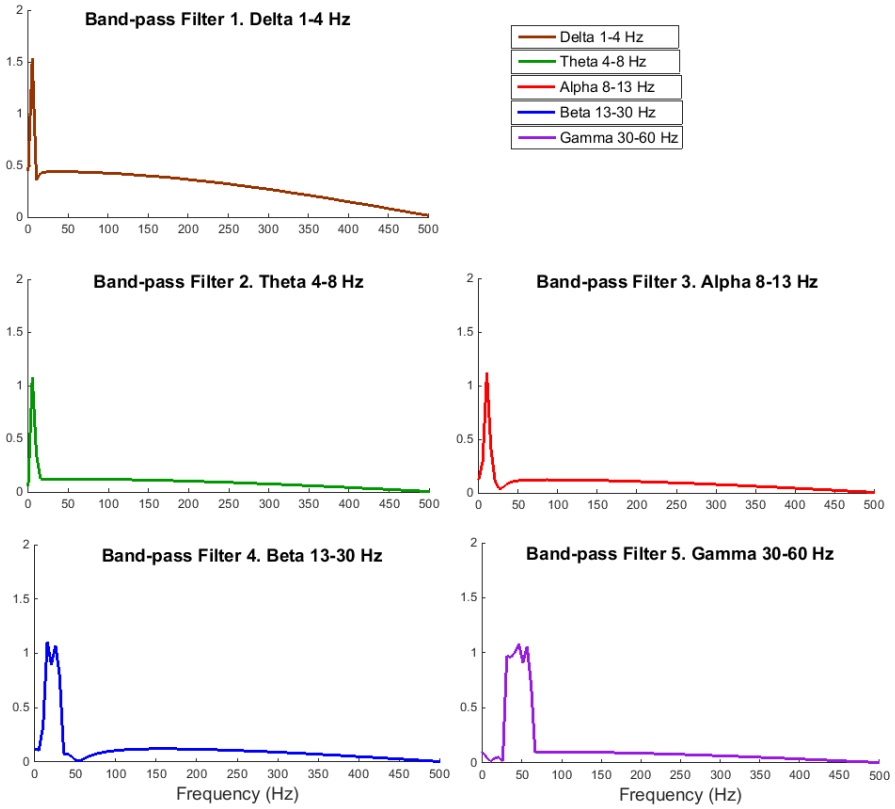

Figure 2.11: Frequency Response of the 5 different filters used in LFP pre processing

2.4.2 Classification

Two classifiers were used for classification. Since the integrity of the data depended upon the temporal sequence, the trials were not shuffled; instead the first 75% of data points used for training and the remaining 25% used for testing. Table 2-9 gives a summary of the results for SVM and Table 2-10 gives the summary for logistic-regression. The accuracy is very high since there are more

negative examples, but as discussed in the introduction, this is not a metric of performance in the field.

Table 2-9: Trial Initiation Classification From SVM

Session #	Total Trials	Trials in Test Data	Accuracy	Recall	Precision
1	88	22	97.13%	0.44%	16.67%
2	91	34	96.88%	0.29%	2.86%
3	84	16	98.27%	0.00%	0.00%
4	105	23	96.74%	0.85%	5.71%
5	119	24	97.62%	3.29%	28.57%

Table 2-10: Trial Initiation Classification From Logistic Regression

Session #	Total Trials	Trials in Test Data	Accuracy	Recall	Precision
1	88	22	94.80%	10.09%	9.62%
2	91	34	94.81%	4.93%	5.35%
3	84	16	96.75%	7.98%	6.50%
4	105	23	93.23%	9.36%	6.09%
5	119	24	91.02%	20.16%	5.98%

The recall (TP/(TP+FN)) and precision (TP/(TP+FP)) are also given in the tables. Recall indicates of the number of data points that were trial initiation, how many were actually classified as such. Precision indicates of the number of data points classified as trial start, how many actually were trial starts. Logistic Regression performed better than SVM, but the recall and precision were still very low.

Figure 2.12: Session 1 classification Results (Red – actual, Blue – predicted)

Figure 2.12 gives the performance of each classifier for the first session and Figure 2.13 gives same for the second session. Red * shows the actual and blue □ shows the points that were classified. The SVM in both sessions have less false positives, while the logistic regression gives a high number of false positives.

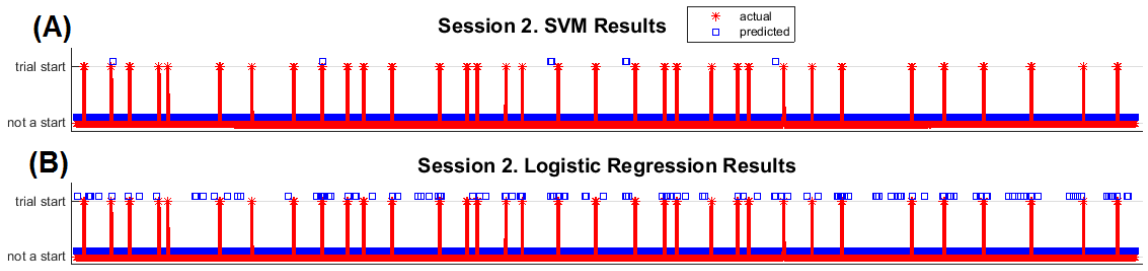

Figure 2.13: Session 2 classification Results (Red – actual, Blue – predicted)

The few data points (2 in session 1 and 5 in session 2) classified by the SVM as trial starts were also classified the same by logistic regression. These few data points are robust to both classifiers. The points classified as trials are clustered around the actual trial starts even though only a few are correctly classified. The temporal resolution needs to be explored here with the nature of the signal causing the predicted positive class to be clustered around the actual positive class. If it shows feasibility (i.e. high precision and high recall), it is possible to build an asynchronous BMI using the NAcc signal as a gate to initiate movement.

2.5 **Summary and Conclusions**

In this chapter, the choice of animal model was discussed and the surgical methods were looked at. The experimental paradigm for studying the NAcc during a goal-directed task was shown; go-no-go paradigm and the two-target

reach paradigm. Next, firing patterns and the histograms were studied to understand the behavior of the NAcc during the goal-directed task. It was seen that more number of units modulated with the robot movement direction, than with the reward, and therefore the information which can be extracted easily from unsupervised methods were not suitable for the BMI application. Next, four commonly used supervised classification techniques were applied to the data. For Don (the monkey that closed-loop was implemented on), the NAcc time window for Critic input was identified as starting either at 0 or 250 msec and ending at or before 1000 msec. Random Forests Classifier performed better than the other three overall and this will be used for the Critic in the closed-loop implementation. For the same monkey the motor cortex (MI) data is used for the Actor and the best time window is 500 msec after the Go Tone. These conclusions were used in designing the closed-loop experiments. The NAcc LFPs were studied briefly to test the feasibility of extracting a trial initiation signal to build an asynchronous BMI. The precision and recall from classification were very low for this signal to be used as expected.

Chapter 3 Development of the Control Architecture

3.1 Control Architecture for the Actor

In order to use the signals acquired after from Chapter 2, we need an architecture that can incorporate these signals in a closed-loop BMI. The findings from the previous chapter (reward/ error representation) can only be used if an architecture can be developed to handle the Critic uncertainty.

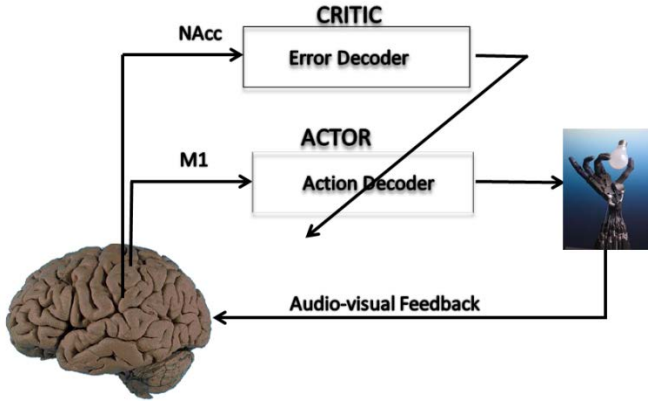

Figure 3.1: Architecture for Biological Actor-Critic Reinforcement Learning.
The Critic is controlled by the NAcc neural inputs and the Actor is controlled by the MI neural inputs. The Critic provides an evaluative feedback to the Actor.

In the present paradigm, x_i is the input to the Actor at i^{th} node and ω_{ij} is the weight of the network with input node i and output node j which is updated using the feedback from the Critic. The output state x_j is computed based on the net state (s_j) of the node and a tanh non-linear transfer function.

$$S_j = \sum_{i=0}^{N} \omega_{ij} x_i \qquad \text{Eq. 3.1}$$

$$P_j = tanh(s_j) \qquad \text{Eq. 3.2}$$

Hence the output of the Actor is given by

$$X_j = SGN(P_j) = \begin{cases} 1, & P_j > 0 \\ 0, & P_j = 0 \\ -1, & P_j < 0 \end{cases} \qquad \text{Eq. 3.3}$$

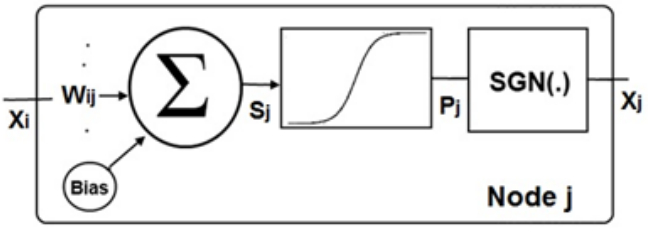

Figure 3.2: Node 'i' of the neural network for the Actor

The update equation is given by

$$\Delta\omega_{ij} = \mu^{+} r(x_j - P_j)x_i + \mu^{-}(1 - r)(1 - x_j - P_j)x_i \qquad \text{Eq. 3.4}$$

where the μ^{+} and μ^{-} are the learning rates for the reward and penalty respectively. The reinforcement feedback is given by r. $r = 1$ is a rewarding action and $r = -1$ is a non-rewarding action. If $r = 0$, then there is no weight update. This is the Hebbian reinforcement Learning (HRL) update equation. The first term corresponds to the reward and the second term corresponds to the penalty.

There are two unique cases for this equation. The first case is when $r = 1$, there is contribution only from the first term and the weight update equation above becomes:

$$\Delta\omega_{ij} = \mu^{+} r(x_j - P_j)x_i \qquad \text{Eq. 3.5}$$

This means that in rewarding trials $(r = 1)$, only the positive component contributes to the weight update. But in non-rewarding trials $(r = -1)$, both terms

contribute and the system is more sensitive to the negative feedback. The second case is when P_j approaches x_j there is contribution only from the second term, hence the weight update becomes:

$$\Delta\omega_{ij} = \mu^{-}(1 - r)(1 - x_j - P_j)x_i \qquad \text{Eq. 3.6}$$

In this case, the system will only adapt for negative feedback. When both the above conditions are achieved, ($r = 1\ and\ P_j \to x_j$), the weights will not update further. During instances where there is no weight update, the system has consolidated the functional relationship between input and output. Unless and until there is a negative feedback, the system will not update further.

3.1.1 Modifications to the Actor

Our previous analysis has shown that the overall system accuracy is limited by the Critic accuracy [109]. Hence, we updated the Actor to incorporate the Critic confidence level. The Critic determines the appropriateness of the action taken by the Actor, the Actor should be able to integrate the feedback even if it is not fully reliable (as is the case in many biological signals). The Critic will give the feedback along with the confidence it has on this feedback. Depending on the confidence given, the Actor weights will be updated only when the Critic confidence is high. More noisy data will result in lower levels of confidence and the Actor weights will not be updated as frequently. The assumption is that by not updating when the Critic feedback is wrong, and thus weights remaining same, has less negative effect on the system than when updating every time with an inaccurate Critic. The trade-off is that the learning

rate is much slower and the system will take much longer to learn the mapping between neural states and the output actions.

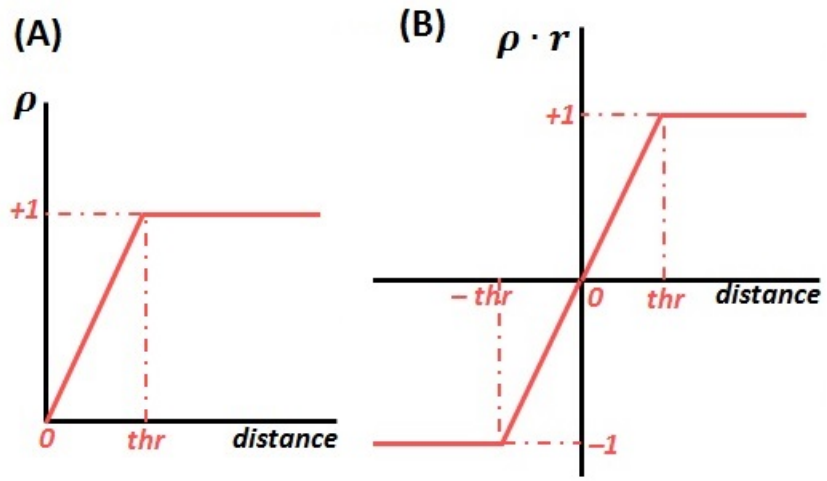

Figure 3.3: How the distance is converted to the confidence and reward. thr=threshold (A) Confidence Only. (B) Confidence and Reward. At lower confidence values, the Critic confidence is low while at higher confidence values, the Critic is 100% confident.

The Critic will give an output of ±1 indicating if it was an action to be rewarded or penalized. In addition, the Critic will also give a value of the confidence (ρ) it has on the feedback given. The update equation thus becomes

$$\Delta\omega_{ij} = \mu^{+}\rho\, r(x_j - P_j)x_i + \mu^{-}(1-\rho\, r)(1 - x_j - P_j)x_i \qquad \text{Eq. 3.7}$$

Where ρ is the confidence in the feedback, r. If both terms ρ and r can be determined from a single step, then the terms can be combined. However, the advantage of having two different terms ρ and r is the ability to acquire them from two different methods if the same method does not give both values. If the Critic value given was correct, the confidence will be increased and if the value is wrong, it will be decreased. There are different methods to derive this ρ value.

3.1.2 Confidence of the Critic

First method of these is to use a neural network. The input to the network is the firing rates form the NAcc. The network will be trained with labels +1 (rewarding) and -1 (non-rewarding/ penalizing) for x_j. During online testing, x_j will give values between -1 and +1. The sign will be given as the feedback r and the scalar value of the f function will be used as the confidence. If the output of the network exceeds ±1, the f function ensures that the confidence remains at 100%.

$$r = SGN(X_j) \quad \text{Eq. 3.8}$$

$$\rho = abs\left(f(X_j)\right) \quad \text{Eq. 3.9}$$

where

$$abs\left(f(X_j)\right) = \begin{cases} 1\,; & X_j > l \; or \; X_j < -l \\ X_j; & 0 < X_j < l \; or \; -l < X_j < 0 \end{cases} \quad \text{Eq. 3.10}$$

l represents the threshold whose values are to be determined by the nature of the Critic data and simulations.

Since this method needs extensive training, we used a second method (distance to the boundary) as the confidence. In the figure below, there are rewarding and non-rewarding training trials marked in black. The red trials are the ones classified. If the classified trial is closer to the decision boundary, it will give a lower confidence and if the distance is higher, then the confidence in the decision is also higher. ρ is the normalized absolute distance from the decision boundary.

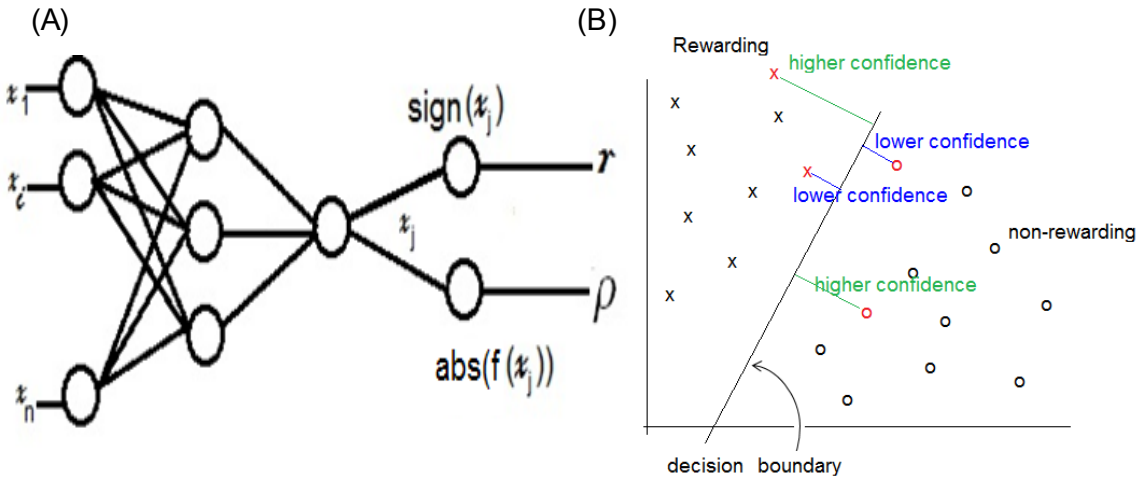

Figure 3.4: (A) An Artificial Neural Network for the Critic with the Reward Value and the Confidence. (B) An Alternate Method to Obtain the Critic Confidence Level. Data points further away from the decision boundary will have higher confidence and the points closer to the decision boundary have lower confidence.

If a probabilistic classifier is used, the probability of a data point being in the particular class can be used converted to the confidence measure. If the absolute value of the probability minus 0.5 is closer to 0, the confidence is low and if it is closer to 0.5, the confidence is high. Figure 3.4 gives an explanation of this. The two curves (red and blue) are the probability distributions for each class. For a given data point, if the probability for class 1 is higher it is classified as class 1, else class 2. Higher the difference of the probability values, greater the confidence. This method is implemented later in Chapter 4.

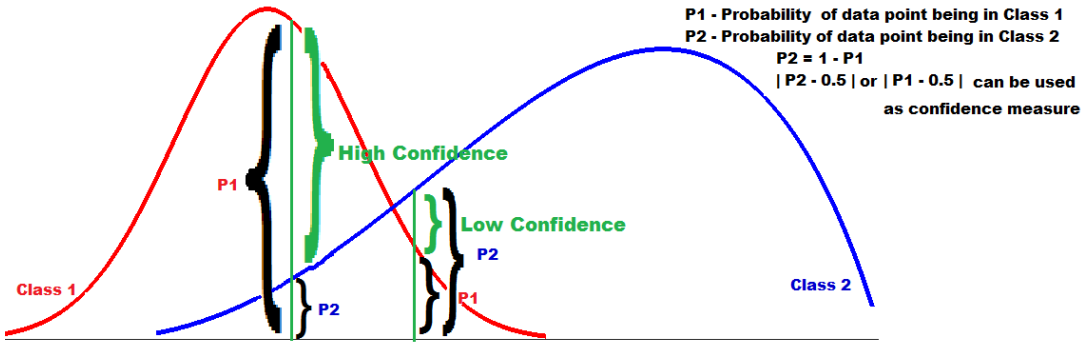

Figure 3.5: Using a probabilistic method to arrive at the confidence. P1+P2=1. abs(P1-0.5) or abs(P2-0.5) can be used as confidence measure.

3.2 **Data Generation for the Actor**

We generated synthetic data to simulate how the above proposed decoder will work with neural signals. Neural data was generated according to the Izhikevich model [110, 111] and added an additional probability component for the stimulus to make the data more noisy. This component ensured that a certain percentage of neurons will be harder to classify. This was verified using the first two PCs. Figure 3.6 shows an example of the generated data set. Figure 3.6 A shows the neural data generated by the standard method in the first two PCs. Even though there is noise inbuilt into the data generation, in the PC space, the data is separable easily. For data shown in Figure 3.6 A, the Actor will be able to classify with higher accuracy. Since we needed to be able to simulate a noisy data set, an additional probability component was added to reduce the stimulus in a certain percentage of neurons (varied from 0% to 100%). The PC plot is shown in Figure 3.6 B corresponds to 25%. The additional probability component was added to the stimulus generation and not to the neural data itself.

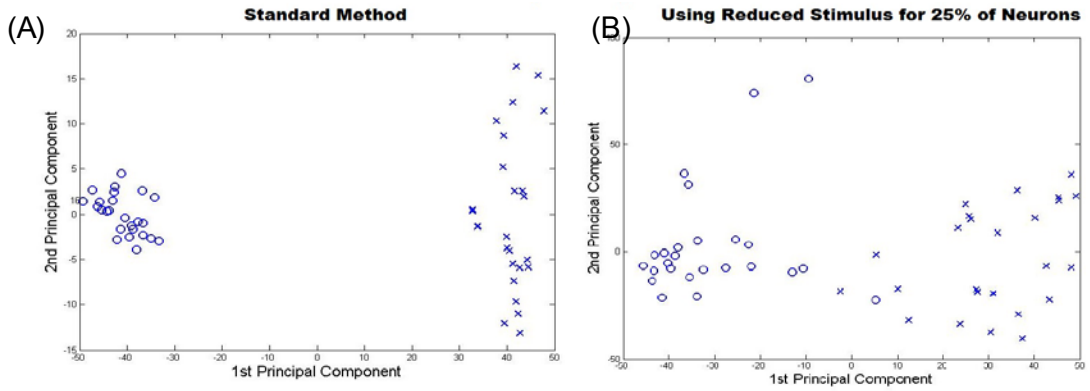

Figure 3.6: An Example of Synthetic Data for 2 states (o and x) in PC space. (A) Standard stimulation method. The PC space is able to discard the noise and give two clear clusters. (B) With Additional Probability Component in the Stimulation. The PC space is more overlapped.

The neural data was generated by the standard method [110] where the model is given by

$$v' = 0.04\,v^2 + 5v + 140 - u + I \quad \text{Eq. 3.11}$$

$$u' = a(bv - u) \quad \text{Eq. 3.12}$$

with the auxiliary after-spike resetting

$$if\ v \geq +30mV,\quad then \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \quad \text{Eq. 3.13}$$

Where v was the membrane potential of the neuron and u represents a membrane recovery variable, which accounted for the activation/inactivation of ionic currents, and it provided negative feedback to v. After the spike reached its apex (+30mV), the membrane voltage and the recovery variable were reset. The synaptic current is given by the variable, I, which was calculated from the stimulus of '1' for spike and '0' at all other times. For excitatory cells, $\mathrm{a} = 0.02$, $\mathrm{b} = 0.2$, $(\mathrm{c}, \mathrm{d}) = (-65{,}8) + (15, -6) \cdot \mathrm{e}^2$ where e is a random variable uniformly distributed, $\mathrm{e} \in [0{,}1]$ [110]. We generated two motor states (motor state 1 and motor state 2) using the above model to depict two actions. The neural data was generated in 3 ensembles, one ensemble each tuned to one state (activity of the particular ensemble correlated with one state) and the third ensemble not tuned to either state simulating noise in real neural data.

While the synthetic data was generated using a biologically realistic model, there are dynamic factors, which contribute to forms of noise not considered in the model. These are factors such as neurons dropping, electrodes deteriorating or breaking and encapsulation. Without making the model more

complicated to mimic the noisy physiological system, we introduced additional noise to the synthetic data by adding a probability component to the stimulus, which generated the I in Eq. 3.11. The actual value of noise in the stimulus was decided by a Gaussian distribution instead of the '1' or '0' as before. The number of neurons with this additional noise was varied from 0% to 100% in 10% increments. This additional probability component resulted in overlapping classes; the higher the probability component, more overlapping in the states generated. This was verified graphically using the first two PCs and confirmed that as the probability component to generate I was increased, the overlapping of the two classes also increased [112].

3.3 **Dealing with Inherently Slow Adaptation**

Real time 'epoching' of the data was used to speed the initial adaptation from the purely random initialization weights to functionally useful. Each trial used all of the past data ten times to rerun through the system. A pseudo-real time normalizing of the inputs was performed before feeding to the network. This was done by keeping a real time record of the highest firing rate detected for each input, and then used to continually update the normalization parameters throughout the session [54, 62].

3.4 Simulations for Dealing with Critic Uncertainty

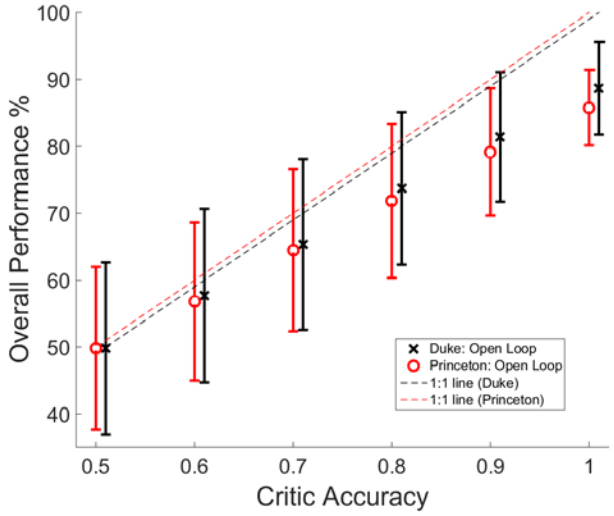

Figure 3.7: Performance of the BMI Vs the Critic accuracy during open loop simulations (mean ± standard deviation)
Duke: black X, 1000 simulations; Princeton: red O, 700 simulations) when the accuracy of the Critic feedback was varied (0.5 to 1.0). dotted lines give 1:1 relationship. The overall performance is limited by the accuracy of the Critic [113].

In previous closed-loop analysis it was concluded that the overall motor control accuracy can be limited by the Critic accuracy [113]. This property is illustrated in Figure 3.7; The results of the open loop simulations for one monkey (black crosses) as well as a second monkey (red circles), indicate that the overall performance of the system is limited by the Critic accuracy [109]. To overcome the overall system accuracy being limited by the Critic accuracy, we developed a new method to update the Actor weights only when the Critic had high confidence (ρ), in the feedback (r) provided. The Actor-Critic architecture was modified as shown in Figure 3.8 to incorporate the confidence term in addition to the already existing reward term.

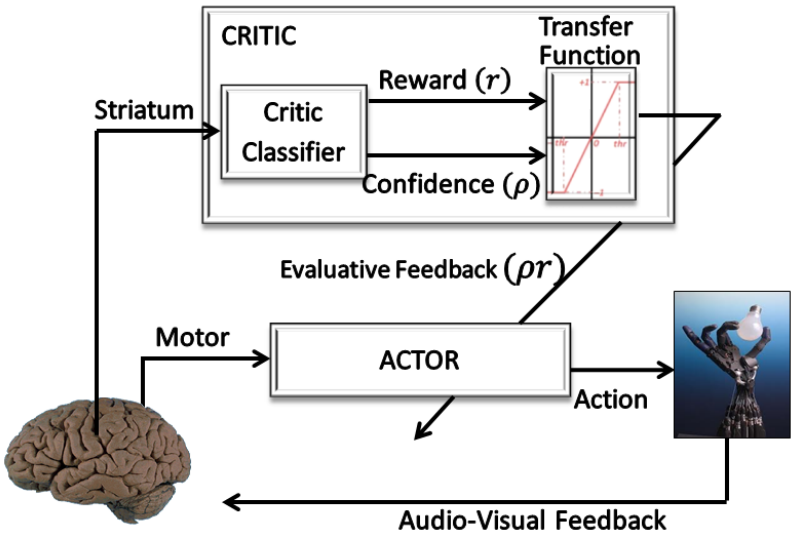

Figure 3.8: Modified Actor-Critic RL showing how Reward and Confidence terms were incorporated in the architecture.

3.5 Can Using the Feedback Intelligently Improve Performance?

As a first step, we tested the hypothesis, using the feedback intelligently can improve performance. These results have been adapted from published work in Frontiers in Neuroscience [112].

We tested the model using 3 different data sets in classification mode. Data sets used were: (1) synthetic data generated by an Izhikevich neural spiking model, (2) synthetic data with a Gaussian noise distribution, and (3) data collected from a non-human primate engaged in a reaching task. We varied the Critic accuracy from 50% to 100% and ran two sets of simulations (S1 and S2) for each of the three data sets; S1, updated the Actor at every trial and S2 updated only when the Critic feedback was correct (i.e. confidence high). This was performed to compare whether it was better to adapt after each trial or only when the Critic feedback was correct. For the purpose of these simulations, we used the correct Critic feedback to indicate a high confidence of '1' and an

incorrect Critic feedback to indicate a low confidence of '0'. This can be determined empirically by the Critic data that would require an in-depth evaluation, which was not the focus of this study. Since the decoder started at a naïve state, we used a pseudo-real time normalizing of the inputs before feeding to the network. This prevented any bias due to the difference in the magnitude of the inputs. This was done by keeping a real time record of the highest firing rate detected for each input, and then used to continually update the normalization parameters throughout the session [109].

3.5.1 Effect of confidence measure on Actor performance

Figure 3.9A shows how the performance level increased as the Critic accuracy increased. The Actor which was updated every time is shown in blue. The performance was always below the 1:1 curve showing how the Actor performance is limited by the Critic accuracy. However, the performance of the system where the Actor was updated only when the Critic was confident (shown in red) was able to perform above the Critic accuracy level as seen in the figure. The performance increased from 50% (±6.6%) to 70% (±8.8%) at Critic accuracy of 50% and further improved from 87% (±10.4%) to 92% (±6.9%) at Critic accuracy of 90%. A Critic accuracy of 90% means that the Critic gave a correct feedback 90% of the trials and wrong feedback 10% of the trials. For example, in our simulations each consisting of 100 trials, a 70% accurate Critic gave correct feedback in 70 trials and wrong feedback in 30 trials. If there was no confidence built-in, the Actor assumes that the value was always correct. In this new system with confidence built in, we reduced the confidence of the wrong feedback to

zero. At lower Critic accuracies (50%, 60% and 70%), the system with the confidence outperformed the system without the confidence by approximately 20%. The performance of the two systems showed significant difference for all Critic accuracy levels from 50% to 90% (Student's paired t-Test, with a two-tailed distribution, alpha 0.001 – shown with * in the figure). By updating weights accurately, the system learned optimal mapping and stabilized with time. Given that the system began with random initial conditions, there was no guarantee that the system would stabilize.

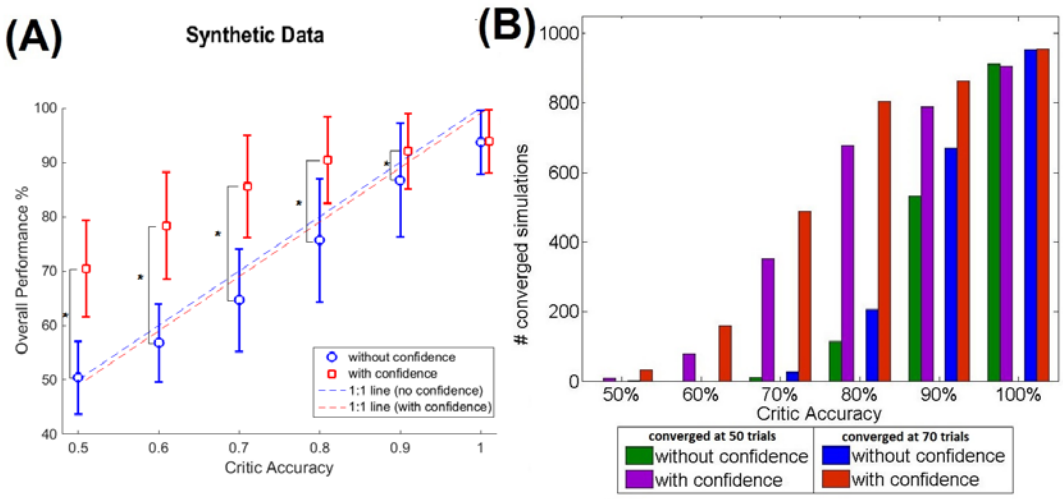

Figure 3.9: (A) Performance of the BMI Vs the Critic accuracy with and without confidence inbuilt. (mean ± standard deviation. 1000 simulations. 100 trials per simulation).
Red: New update rule with confidence. Blue: Previous method with no confidence. Black: 1:1 relationship. Critic accuracy was varied from 50% to 100% with 100% being the best. * shows the values which showed statistical significant difference (alpha 0.001). The overall performance of the blue curve is limited by the accuracy of the Critic but the overall performance of the red curve is able to go beyond the Critic accuracy, decoupling the performance from the Critic accuracy.
(B) Stability of the system without (green/blue) and with (purple/red) confidence. Plot shows the number of simulations that maintained 100% accuracy beyond 50 trials (green/purple) for convergence and beyond 70 trials (blue/red) for convergence [112].

Figure 3.9B gives a summary of the number of simulations out of 1000 that stabilized after 50 trials and 70 trials with and without the confidence. The convergence or stability was defined as maintaining 100% accuracy (last 50 trials

or last 30 trials). The number of simulations that did stabilize at lower Critic accuracies was higher for the system with the confidence measure. At higher Critic accuracy levels, the overall performance was no longer limited by the Critic accuracy but by the data itself. As the Critic confidence increased, the difference in performance between the two systems became smaller and converged to a single value (94% ±5.8%) since at 100% Critic accuracy, both systems effectively have the same update equation.

Figure 3.10 shows the details of the action selected in each trial and also the Critic values for that particular trial. Figure 3.10A has two sets of simulations S1 and S2 and Figure 3.10B also has two sets of simulations S1 and S2. Each simulation started with random initial conditions. Figure 3.10 (A and B) shows two such examples with two different Critic accuracy levels. The Critic accuracy was changed randomly based on the percentage given to the decoder. In Figure 3.10A, the Critic is 60% accurate and the top subplot shows the performance of the system if the Actor was updated every time (S1). The overall performance in this case is 47%. The first trial was correct, but the Critic gave a wrong feedback and the Actor weights were updated with this erroneous feedback causing the second trial to be wrong. When the Critic gave a correct feedback during the third trial, the system started performing correctly. However, due to the erroneous feedback the performance was not stable. Even when the Actor chose the correct action, if the Critic provided a wrong feedback, it decreased the performance. In contrast, the second subplot shows the performance when the Actor was updated with a confidence level (S2). For the same neural data, order

of trials and Critic feedback, the performance of the second system is 80%. Even though the Critic gave wrong feedback at first, the Actor learned to ignore this and was able to have a better outcome.

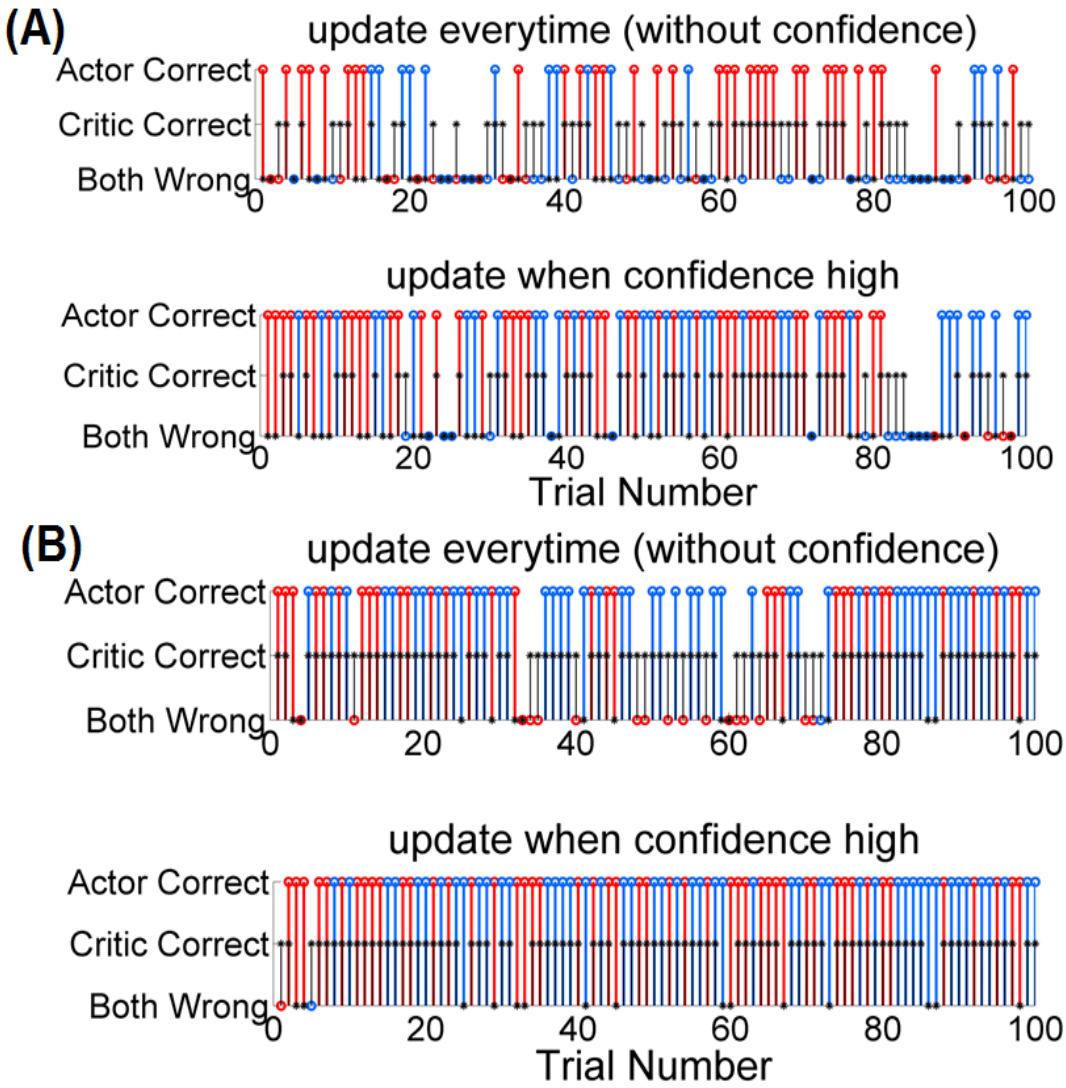

Figure 3.10: Performance of each decoder during the length of the experiment for one simulation starting at random initial conditions. 100 trials. Red: Action 1, Blue: Action 2, Black: Critic. (A) Critic accuracy 60%. Both decoders perform correctly in the first trial but the Critic gives a wrong feedback. The first system changes the weights causing the second trial to be wrong. Again, the Critic gives a wrong feedback causing the third trial also to be wrong. Since the system weights are updated every time, wrong Critic feedback causes the system to perform below the Critic accuracy. However in contrast even though the second subplot also starts the first trial the same way, the erroneous feedback does not affect it and the decoder is able to perform better than the first system. (B) Critic accuracy 80%. The first system starts with a correct action, but is very sensitive to wrong Critic feedback. The second system starts with a wrong action, but by the 6th trial is able to achieve good performance and maintain throughout the rest of the session [112].

Figure 3.10B shows the performance of the two systems when the Critic accuracy was 80%. The top subplot shows when there was no confidence measure and the Actor updated every time (S1). The bottom subplot shows the Actor updating only when the Critic was correct (S2). The Critic provided a similar output at the beginning. For the first system, the system started with random weights and continued to do well with correct Critic feedback at the beginning. However, an erroneuous Critic feedback at trial 3 caused the system to perform wrong in the next trial. In contrast, the second system started with random weights which caused the first trial to be wrong but the system received good feedback and was able to perform correctly in the subsequent trials. In the first 5 trials, the first system performed better than the second. However, since the second system Actor weights were only updated when the Critic feedback was good, it took longer for the second system to learn the ideal mapping.

3.5.2 Neural Perturbations – Additional Noise in Data

Figure 3.11A shows how the system with the Critic confidence level still performed better than the system which updates the Actor weights every time even with the additional noise. The system which updated at every trial performed at chance level (50% performance) at lower Critic accuracies, while the system with the Critic confidence performed better (at Critic accuracies 80% and below the difference in the performance was approximately 10%). However, as the Critic accuracy increased (beyond 70%), the system accuracy did not increase as expected in both curves (i.e. both systems stayed below the 1:1 curve). This was due to the limitations in the input data as the data to the

decoder was noisy and the states were not as clearly separable. As noted in the previous section, the performance of the two systems showed significant difference for all Critic accuracy levels from 50% to 90% (Student's paired t-Test, with a two-tailed distribution, alpha 0.001 – shown with * in the figure). In Figure 3.11A, the probability component used to generate I was 40%, which was most similar to the NHP data shown in the next section. Figure 3.11B shows how different noise levels affected the overall performance as the Critic accuracy increased. Each colored trace is a different noise level as shown in the legend. With low noise levels, the system was still able to perform amidst the Critic inaccuracies. However as the noise level increased, the system performed at chance (50%) at low Critic accuracy levels and performed marginally above chance even at higher Critic accuracy levels.

3.5.3 Simulations using NHP Data

These results are shown in Figure 3.11C where the blue trace shows the performance of the Actor updating every time and the red trace shows the Actor updating only when the Critic is confident. Similar to the results of the synthetic data, we can see an improvement (from 50% to 63% at Critic accuracy of 50% and from 77% to 83% at Critic accuracy of 90%) in the overall performance by adding the confidence measure in the update equation. This is more apparent in lower Critic accuracies (At alpha = 0.001 Critic accuracies 50% to 90% showed significant difference – shown with * in the figure). At higher Critic accuracies, the system which only updates when the Critic is confident is still able to do better but the difference in the percentages was smaller.

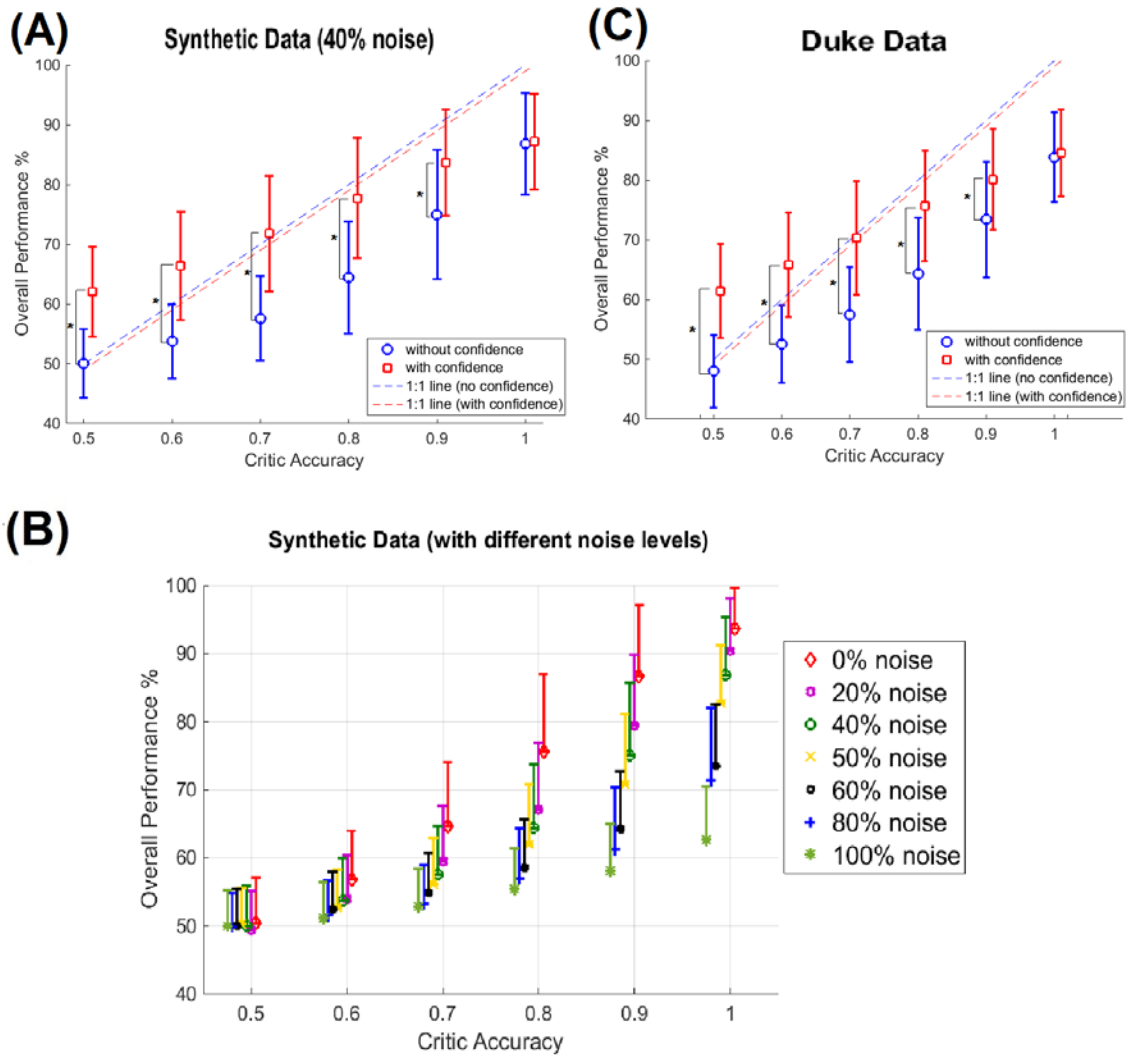

Figure 3.11: (A & B) Effect of noise on the overall performance. (C) Results of the simulations where the monkey controls the robot arm (offline simulations). Dotted: 1:1 relationship. (A) Performance of the BMI Vs the Critic accuracy with 40% of the neurons receiving a less stimuli than the standard (mean ± standard deviation. 1000 simulations. 100 trials per simulation). Red: New update rule with confidence. Blue: Previous method with no confidence. Critic accuracy was varied from 50% to 100% with 100% being the best. * shows the values which showed statistical significant difference (alpha 0.001). The overall performance of the blue curve is limited by the accuracy of the Critic but the overall performance of the red curve is able to go beyond the Critic accuracy. Hence, decoupling the performance from the Critic accuracy. (B) How the overall performance changes with the Critic accuracy (1000 simulations). Each curve gives a different noise level of the data set. (C) Results of the simulations where the monkey controls the robot arm. Performance of the BMI Vs the Critic accuracy with and without confidence inbuilt for data collected from monkey DU. (mean ± standard deviation. 1000 simulations). Red: New update rule with confidence. Blue: Previous method with no confidence. Black: 1:1 relationship. Critic accuracy was varied from 50% to 100% with 100% being the best. * shows the values which showed statistical significant difference (alpha 0.001). At lower Critic accuracies, the new update with confidence performs much higher than the one without the confidence measure. As the Critic accuracy increase, the plot with the confidence measure is able to outperform the curve without the confidence measure. However, the difference in the performance becomes smaller as the Critic accuracy increases suggesting as before that the Critic is no longer the limitation, but the nature of the input data itself [112].

At lower Critic accuracies (80% and below) the difference in performance is approximately 13% and at 90% Critic accuracy the difference in performance is approximately 7%. 90% Critic accuracy means that 9 out of 10 feedback given by the Critic is correct. When the Critic feedback was always correct, the two systems converged to approximately the same performance value. Here we observe that the Critic is no longer a limiting factor for overall performance. The overall performance is now bound by the Actor/ MI neural data.

We conclude that our hypothesis is true: updating the Actor weights, only when the Critic is confident of the feedback, improves performance. Updating at low confidence values will introduce wrong feedback into the reinforcement learning trajectory of the Actor.

3.6 **Data Generation for the Critic**

The next step is to incorporate the Critic component; we tested this using synthetic NAcc data. Approximately 95% of the NAcc is comprised of medium spiny projection (MSP) neurons [114]. We needed a biologically realistic model to capture all the neurocomputational properties of MSP neurons but was reduced in computational complexity to be more efficient for simulations. Humpries et al. reduced neuron model bases the Izhikevich model and we used this with varying spike rate and modified spike-event generator to capture rewarding and non-rewarding trials [115, 116]. The ability of the MSP cells to switch between different states and its physiological properties are modulated by dopamine (DA) [114]. DA is directly related to rewarding behaviors; higher DA levels with higher

firing rates for higher rewards and lower DA levels with lower firing rates for lower rewards/ no rewards [107, 115].

We used 10 neural ensembles with 4 neurons each. 1st ensemble was tuned to the reward (higher firing rate during reward, i.e. DA high and random for no reward), 2nd ensemble tuned for non-reward (lower firing rate during non-reward and random for rewarding) and, the remaining ensembles were background activity not tuned to either action. The ensembles were chosen such that it represented reward modulation in the NAcc which has approximately 10-20% neurons modulating for reward [117]. 1 sec of data was used to simulate the trial.

The values used to set the Spike Train Parameters were altered for each type of trial. For Rewarding trials, the values were drawn from a uniform distribution [0.75 1]*10^-3. For non-rewarding trials, these values were selected from a uniform distribution [0.65 0.85]*10^-3. For both types of trials, the values for the ensemble not tuned, were drawn from a uniform distribution [0.5 0.85]*10^-3.

3.7 Critic Data Classification by different methods

3.7.1 Clusters in the data

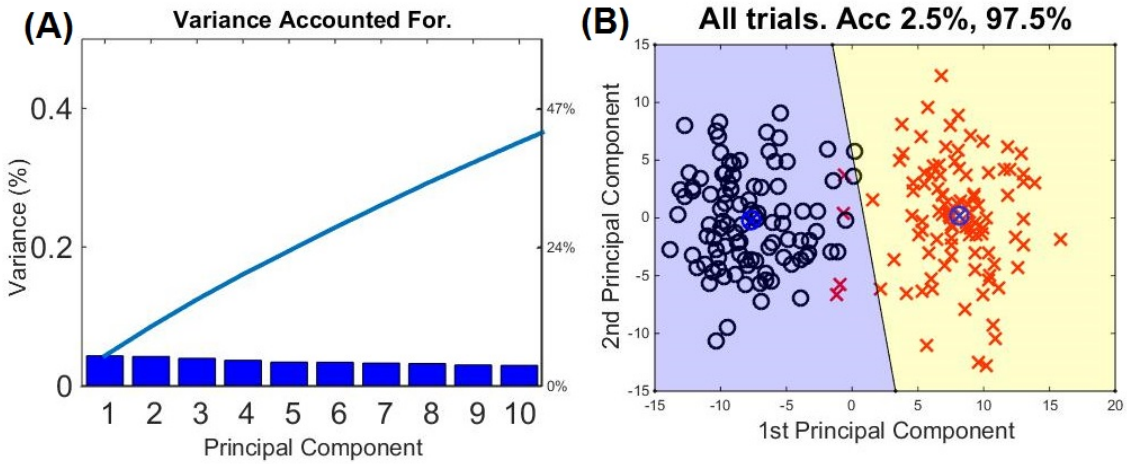

Figure 3.12: (A) Variance accounted for in the first 10 PCs. (B) data in PC space with the clusters from k-means.

We performed PCA on the data and 9 PCs accounted for 92% of the variance. For ease of visualization, we used the first two PCs (variance accounted for: 43%) and clustered the data using k-means with the criterion for minimization as the squared Euclidean distance. We labeled the clusters as rewarding or non-rewarding manually to maximize the classification accuracy. Next we cross referenced with the actual data classes and found that k-means clustering achieved 97.5% accuracy both false positives and false negatives. These false positives and false negatives can affect on the overall performance when using as the Critic [112]. It may be possible to reduce the misclassifications with more sophisticated classifiers. This is looked at in the next section.

3.7.2 Misclassification Rates

The goal of classification is to be able to reduce misclassifications (i.e. false positives and false negatives). However, closer to the decision boundary,

class overlapping occurs and this region has a higher probability of being misclassified by any classifier. Several methods have been developed to address this issue, but none of these methods assure zero misclassifications [118].

We used several different methods and all of them had at least one misclassification. Higher the number of training data used, the better the classification, but fewer trials for testing since we wanted to keep the total number of trials limited. Pruning the input space by PCA and using only the first two PCs for classification rendered better results than using all of the neurons for LDA.

Figure 3.13 shows how the different quantities of training data affected the classification. Higher the number of training data used, the better the classification, but fewer trials for testing since we wanted to keep the total number of trials limited. Therefore increasing trial number is not a feasible option.

Table 3-1: Confusion Matrix For Different Unsupervised Clustering Methods (*K-Means Function – Squared Euclidean Distance, **GMM – Highest Accuracy Of 10 Iterations).

			Predicted Class	
Clustering Algorithm			+1	-1
k-means*	Actual Class	+1	96	4
		-1	0	100
GMM**		+1	74	26
		-1	3	97

When the training data was 5%, the boundary was not as appropriate as when the training data was 10% or more. Increasing the training data set in this case, only removes the ambiguous trials from the testing data, but doesn't do

anything to the boundary. The confusion matrix for each set of classifications is given in Table 3-1 and Table 3-2. When the training data was limited to 10%, the LDA with PCA and the SVM both performed similar, but when the training data set was increased, LDA with PCA outperformed the SVM. However, even with 30% of training data, the classifiers still had at least 2 misclassifications out of 170. Since we are using this data as the feedback into a BMI, we want to reduce these misclassifications further. In the next section, we suggest a possible method of overcoming limitations of classification due to the data itself.

Table 3-2: Confusion Matrix For Different Supervised Classifiers And Different % Of Training Data.
Each Row Gives A Different Method Of Classification. Each Column Gives Different Percentage Of Data For Training, Eg. 10% Column Means That 10% Data Used For Training And 90% For Testing.

Training Data			5%		10%		15%		20%		25%		30%	
			Predicted Class											
Classifier			+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1
Bayes	Actual Class	+1	92	3	88	3	83	4	75	3	71	3	67	2
		-1	13	82	7	82	0	83	2	80	1	75	2	69
kNN		+1	88	7	80	11	75	12	73	5	68	6	64	5
		-1	2	93	1	88	1	82	2	80	2	74	1	70
SVM		+1	71	24	84	7	80	7	71	7	69	5	67	2
		-1	4	91	4	85	4	79	2	80	1	75	3	68
Tree Bagging		+1	90	7	86	9	90	2	86	4	86	1	83	2
		-1	47	51	11	84	25	68	6	84	2	86	0	85
LDA		+1	n/a		n/a		n/a		n/a		62	12	62	7
		-1									5	71	8	63
PCA & LDA		+1	91	4	87	4	83	4	74	4	70	4	66	3
		-1	1	94	2	87	1	82	2	80	0	76	0	71

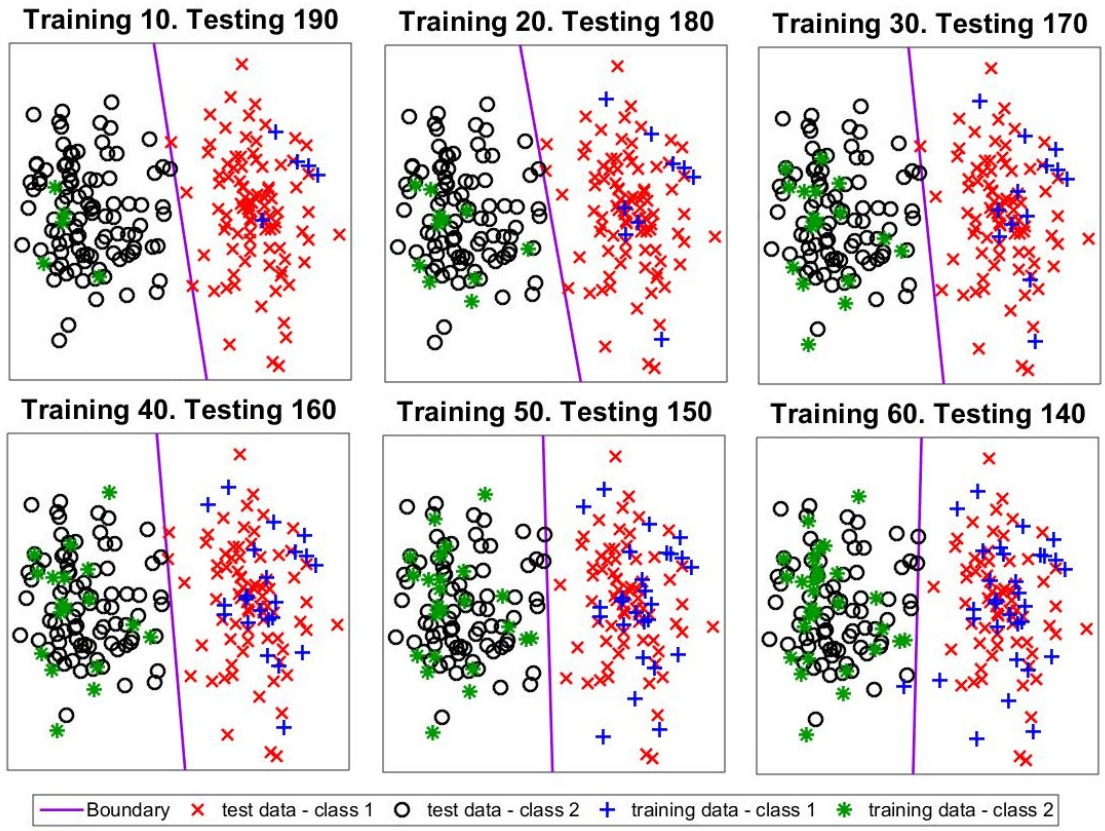

Figure 3.13: How the different training and testing data quatities effect the mislabeled trials. Data in PC space with LDA classification

3.8 **Implementing Offline HRL Decoder with Critic Feedback**

MI data and NAcc data were generated and used as input to our classifier which was a RL based BMI to predict the action to be taken. Figure 3.1 shows the architecture of the Actor-Critic RL. We used Izhikevich model to generate MI data and this was used as input to Actor. The previous MSP model was used to generate NAcc data and this was used as input to Critic. Actor was a fully connected feed forward neural network. The number of nodes in the input layer was the number of neuronal inputs of the MI. The hidden layer had 5 nodes and the output layer number of nodes was the number of actions to be selected from. Each node calculated the output, x_j based on the input, x_i and the weight, w_{ij}.

$$x_j = sgn[P_j] = sgn[f(\sum_i w_{ij} x_i)] \quad \text{Eq. 3.14}$$

Where the transfer function $f(\cdot)$, was a hyperbolic tangent function. The RL operated from a greedy policy, where the node with the highest value was selected as the action. The weight update rule for HRL is given by:

$$\Delta\omega_{ij} = \mu^+ \rho r (x_j - P_j) x_i + \mu^- (1 - \rho r)(1 - x_j - P_j) x_i \quad \text{Eq. 3.15}$$

where ρ is the confidence in the feedback, r ($0 \leq \rho \leq 1$ and $r = \pm 1$). μ^+ and μ^- represent the learning rates for the reward and penalty components, respectively. In our simulations we used $\mu^+ = \mu^-$ but with different values for hidden and output layers. For the Critic output driven by the NAcc, we assumed that the wrong feedback was due to data points being close to the boundary and not due to mislabeling of trials (i.e. wrong feedback with high confidence was not considered).

RL by nature is slower in adapting due to exploration. Once the RL agent has enough knowledge about the environment, it proceeds to exploit the situation. However, in a BMI setting, we do not have the luxury of long exploration; therefore previously, all the trials were replayed (epoching) 10 times in the background with keeping all the past data in the memory. However, in real time, epoching is not possible since each epoch needs a corresponding Critic output. To have a corresponding Critic output, there needs to be a robot movement for the NAcc to generate a rewarding/non-rewarding neural state. Therefore as a compromise, we used the first 10 trials for exploration. In the first 10 trials, we use a known Critic feedback and had memory (all the past trials)

replaying (epoching) 10 times. From the 11th trial onwards, the memory kept was only the most recent trial and there was no replaying/ epoching. The pseudo code for the HRL BMI is given in Table 3-3.

Table 3-3: Algorithm For The HRL BMI

(1)	Initialize weights w_{ij} and learning rates, μ^+ and μ^-
(2)	Generate MI data – simulating animal making a reach (Eq. 3.11,3.12,3.13)
(3)	Calculate the output based on Eq 3.14.
(4)	Execute Action
(5)	If within first 10 trials, set Epoch=10, go to step (6), else go to step (7)
(6)	Calculate reward (r) based on trial type. Set confidence =1 $(\rho = 1)$. Use all the past trials Update Weights based on Eq 3.15. Epoch++ Calculate the output based on Eq 3.14. If Epoch<10, repeat (6), else go to step (8)
(7)	Generate NAcc data – simulating animal's perception of reward Decode the reward (r) and confidence (ρ) from NAcc data Update Weights based on Eq 3.15. Calculate the output based on Eq 3.14.
(8)	Return to step (2)

3.9 **Deciding the Threshold**

One of the measures of the threshold was the normalized distance to the boundary. We varied the threshold from zero (no threshold) to 0.5 (half of the distance) in 0.05 intervals. Since with PCA, it is easier to visualize the data, LDA and PCA combination was used for the results shown in Figure 3.14 and Figure 3.15. As seen in Figure 3.14, the higher the threshold, the higher the size of the ambiguous class. Blue and green dots represent rewarding and non-rewarding classes respectively. The red and purple indicate which trials were in the ambiguous region.

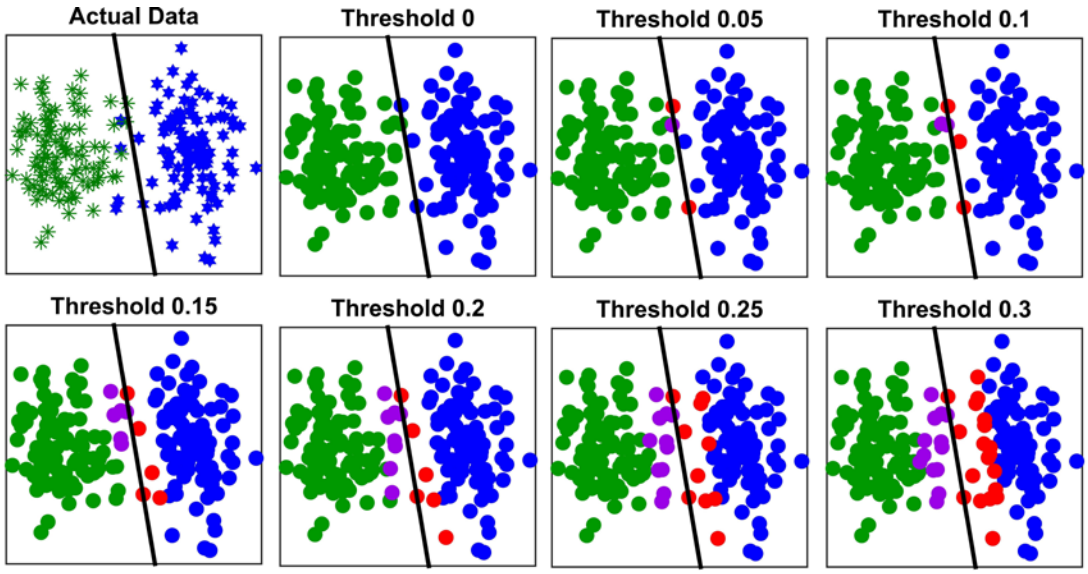

Figure 3.14: Data in PC space with LDA classification. 10% for training and 90% for testing. Blue and green dots represent rewarding and non-rewarding classes respectively. The red and purple indicate which trials were in the ambiguous region.

In deciding where to put the threshold, the different metrics of accuracy is important. Figure 3.15 gives the different metrics for accuracy. However, at higher threshold levels, more data points have lesser confidence and therefore, a lesser Critic output. The blue trace for each plot shows the results if there was no threshold used and the red traces show how the threshold affects the different accuracy levels. Subplot (A) shows how the overall accuracy drops with the increase in threshold while subplot (B) shows the new accuracy (i.e. not considering the ambiguous class) which increases as expected when the threshold increases. Subplot (C) shows the precision drops with the increase in threshold. (D) is the size of the ambiguous ("I don't know") class. As seen, the number of ambiguous data points increases as the threshold is increased. (E) through (H) show the different measures of performance. At 0.05 threshold, the false positives reduce to zero, but the true positive number also starts to drop from 100%. At 0.25 threshold, the false negatives drop to zero (subplot (H)).

Even though the false positives dropped as the threshold increased, the true positives and true negatives also dropped. Therefore, the threshold cannot be increased indefinitely. The purpose of the threshold was to reduce the erroneous feedback to the system, but it is unwise to ignore the correct feedback within the ambiguous region. Therefore, the threshold should be low enough to capture as many of the correct feedback, but also high enough so that wrong feedback does not have 100% confidence values. While a hard and fast rule cannot be given for the setting of the threshold, we should select it high enough that the FP and FN are close to zero and low enough that the accuracy and precision are as high as possible. For this particular data set, we selected threshold levels from 0 to 0.5 to evaluate how the thresholding affect the performance. This data was used as the Critic input in our Actor-Critic RL model.

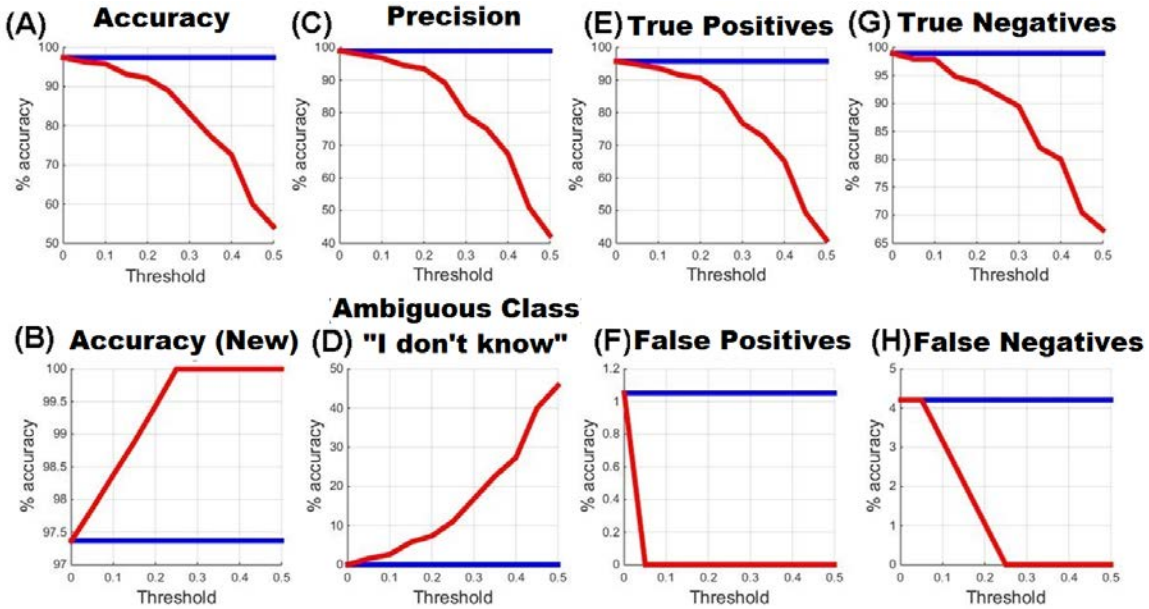

Figure 3.15: Data LDA & PCA (10% training). The blue trace for each plot shows the results if there was no threshold used and the red traces show how the threshold affects the different accuracy levels.
(A) Accuracy calculated from all the data points. (B) Accuracy as calculated from the data points outside the threshold (i.e. “I don’t know” class discounted). (C) Precision considering all the data points. (D) size of the ambigious/ “I don’t know” class. (E)-(H) the TP,TN,FP and FN rates.

3.10 **HRL BMI Simulations**

Using the data in the previous section and the algorithm in Table 3-3, we used 200 trials, 4 targets and performed multiple simulations with the same data. The first 10 trials as explained prior had a perfect Critic, used all past trials in memory and replayed (epoch) 10 times per trial. From 11^{th} trial, only the present trial was in the memory and no epoching was performed. We used a realistic Critic decoded from PCA+LDA with confidence and reward. Figure 3.16 shows how the system performed for each trial. Blue shows the type of target (1-4) and black shows the Critic confidence output (high or low confidence). Red shows the system performance: positive for correct and negative for wrong. (A), (B) and (C) subplots show no threshold, 0.12 threshold and 0.24 threshold respectively. For comparison, all three systems start with the same initial weights and the order of trials are the same. However, each have different Critic feedback. The overall accuracy of all three systems are the same and the first 10 trials have very similar outcomes. System (A) is susceptible to wrong Critic outcomes as shown by the trials following wrong Critic (red bars negative after black * zero). System (B) has a few low confidence Critic outputs in comparison to system (C). The reason for (C) having many lower confidence Critic outputs was that as the threshold increased, the ambiguous region also increased, causing more trials to have lesser update than when the Critic confidence was high.

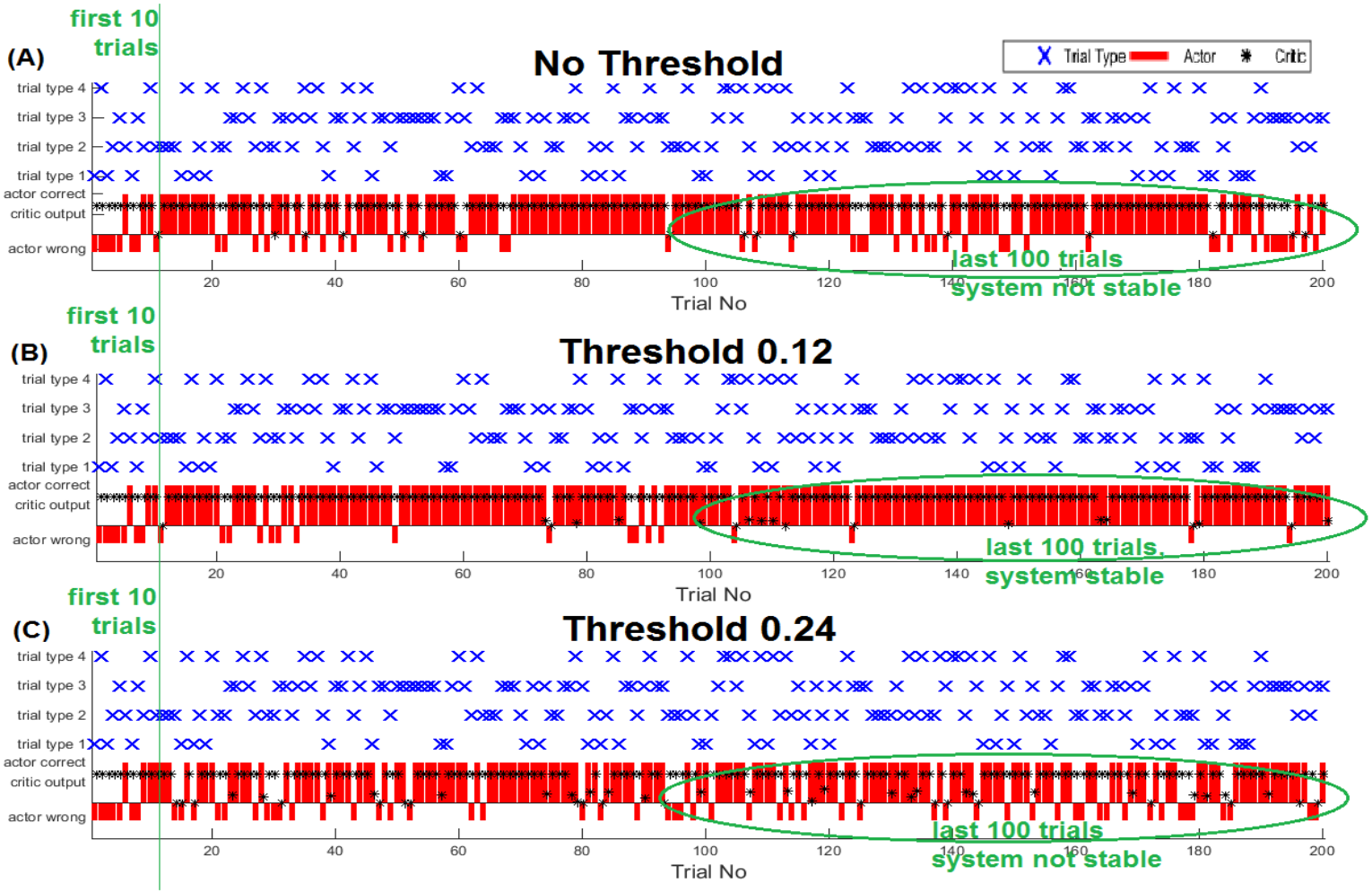

Figure 3.16: Performance of the system in one simulation and how threshold affects the performance. Blue – type of target. Red – system performance +ve-correct, -ve-wrong). Black – absolute of the critic output with confidence – y axis here shows the critic output (for black traces only). (A) No threshold (B) Threshold=0.12. (C) Threshold=0.24

Figure 3.17 shows how the accuracy per 20 trials change with time. The system without a threshold (blue) and the system with 0.12 threshold (red) start off with similar accuracies. The blue trace however, changes rapidly throughout the session ending with a low block accuracy. On contrast, the red system has less fluctuations and towards the end maintains a higher accuracy. The system with a 0.24 threshold (green) has fewer fluctuations than the system with no threshold, but is not able to achieve high accuracy levels even by the end of the session.

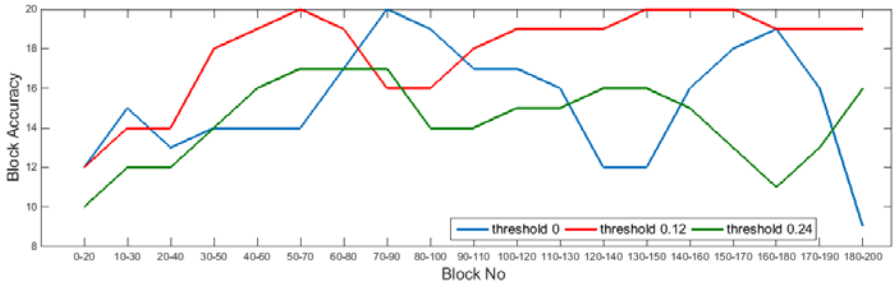

Figure 3.17: Block Accuracy for the 3 example simulations

Figure 3.18 shows how weights, w_{ij} changed during the experiment. The weights up to iteration number 550 are for the memory and epoching of the first 10 trials. Beyond 550, each weight update corresponds to one trial. Subplots (A) and (B) show system with no threshold, while (C) and (D) show a threshold of 0.12 for Critic and (E) and (F) show a threshold of 0.24. Hidden Weights for all three systems (A/C/E) are similar with the second system (C) showing less variance towards the end of the session. Here we see how the unstable system (A/B) can still have higher accuracies, but with introduction of the confidence measure, the instability is reduced, and performance increases (C/D) and when increasing the threshold even further, the weights are not as smooth (E/F). In the

last half of the session (100 trials) the hidden weights with no threshold have a change of 57% compared to the whole session, whereas the output weights in the last 100 trials have a change of 31% compared to the whole session. However, with a 0.12 threshold, for the same section (last 100 trials) the hidden weights change by 28% and output weights by 24% in comparison to the whole session. When the threshold is increased to 0.24, these values are 42% for hidden weights and 56% for output weights.

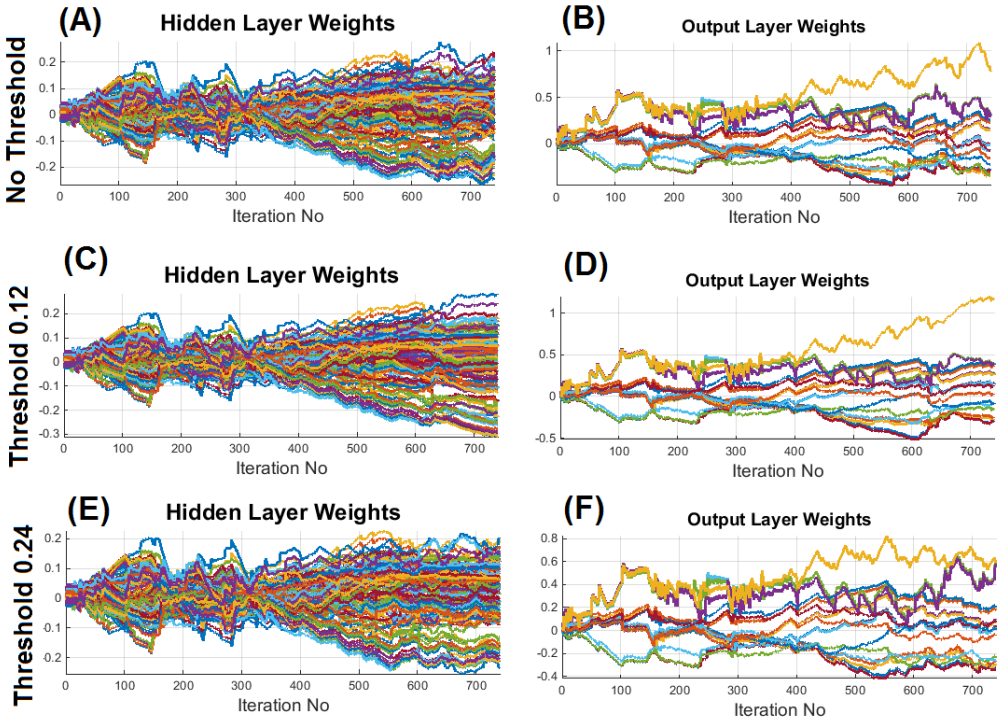

Figure 3.18: Weight traces for each of the simulations in the previous figure. The weights up to iteration number 550 is for the memory and epoching of the first 10 trials. (A/B) No Threshold. (C/D) Threshold=0.12. (E/F) Threshold=0.24. (A/C/E) Hidden Weights (B/D/F) Output Weights.

Figure 3.19 shows a summary of 1000 simulations done for each threshold level. Figure 3.19A shows how with the increase in the threshold level, the accuracy increases and drops after 0.1. This is expected since, when the threshold increases, the Critic feedback is lower within the ambiguous region

causing some of the correct Critic feedback also to not be considered. Figure 3.19B shows how the convergence varies with the threshold level. This explains the time varying nature of performance of RL; the latter trials have better performance than the beginning. Therefore we define the average accuracy of the latter part as convergence. Here, convergence is defined as the number of simulations that had more than 80% accuracy in the last x% of trials. Each colored bar shows a different x%: green, 10% and orange 40%. As the x% increases, the number of simulations converged reduces since at the beginning of the session, weights have not converged. The convergence plot findings are similar to that of the accuracy plot, where it is seen that the number of simulations converged increases with increase of threshold and then drops as the threshold is increased further.

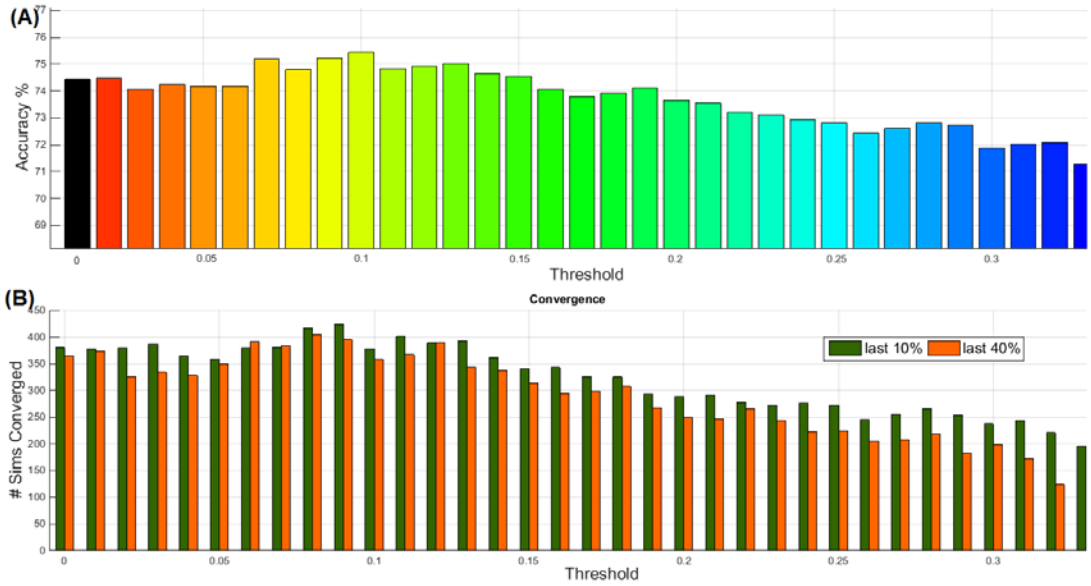

Figure 3.19: (A) How the Actor accuracy changes with the threshold. (B) How the Convergance (Accuracy of the last x% of trials) changes with the threshold level.
How many of the simulations showed convergance in the last 10% (green) and 40% (orange) are shown here. Weights were decided to be converged if the number of accurate trials in the last x% was beyond 80%.

It was also seen that the average is a skewed representation of the results. Hence Figure 3.20 shows the distribution of the different simulations. The red traces show lower thresholds, green/ blue with medium thresholds and purple with higher thresholds. It can be observed in the figure that as the threshold increases (from red to green), the peak rises and is around 76%. As the threshold is increased further (green to blue), the peak drops and the curve shifts to the left (lower accuracy). When the threshold is increased further (blue to purple), the trend of low performance continues. With this plot, the best threshold level can be set in the orange/green region.

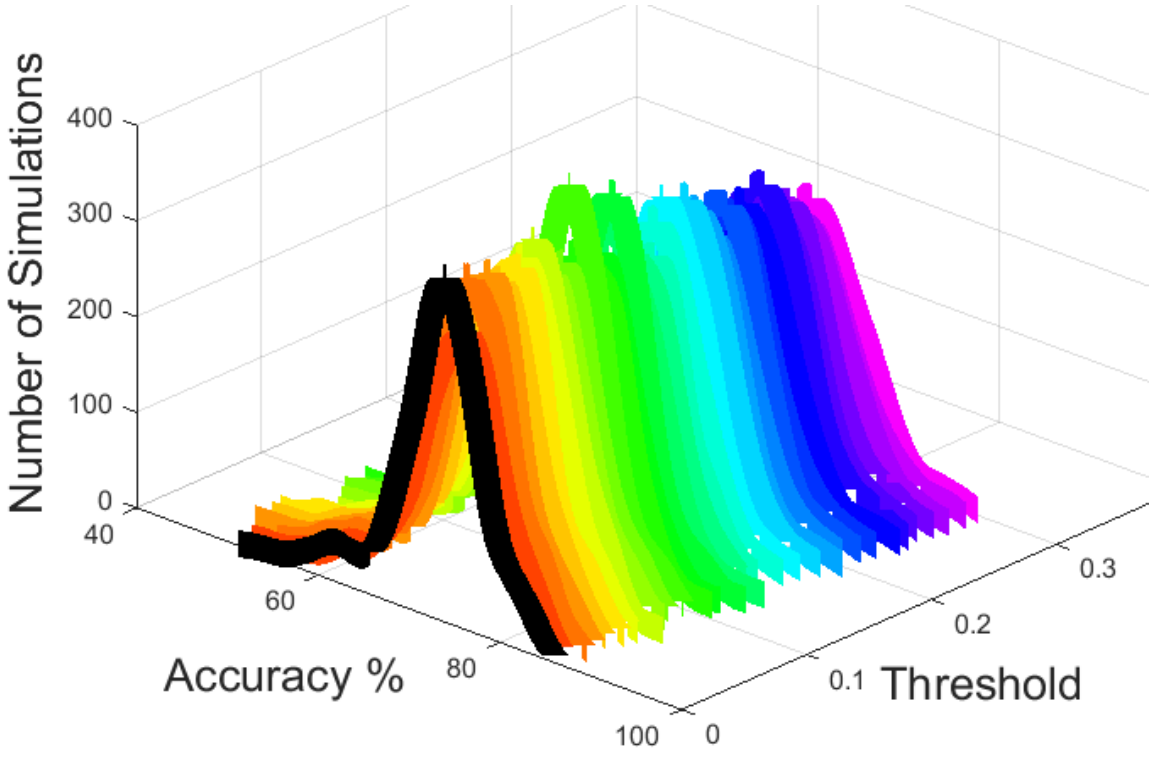

Figure 3.20: Distribution of the simulations for each threshold level. X-axis: accuracy percentage. Y-axis: threshold level. The red traces show lower thresholds, green/ blue with medium thresholds and purple with higher thresholds. Z-axis: how many simulations showed this accuracy.

3.11 **Summary and Conclusions**

In this chapter, Actor architecture was developed and tested with the aid of synthetic data. The hypothesis in this chapter was: using the feedback only when correct, improves the performance of the system. We tested this first by using a Critic which had complete knowledge of the accuracy of its own output and concluded that if such a Critic can be developed, the Actor performance can be improved. Actor data was generated by the Izhikevich model and tested on the system. The system was also tested on NHP data, with similar results to the synthetic data with noise. Next, the method of how the Critic can have knowledge in its own output, was proposed as a confidence measure. A graphical solution was provided to show proof of concept of how this can be implemented and was implemented on synthetic NAcc data generated by Humpries modified Izhikevich model. Finally, this NAcc synthetic data was provided to a Critic to which gave the rewarding feedback a well the confidence it had in its own output. This was used to update the Actor in an offline simulation of the BMI experiment. In these offline simulations, the confidence measure (via the threshold) was varied and the system performance was studied as the threshold varied. It was concluded that a very large threshold increased the ambiguous region, thus not providing enough feedback for the Actor on most trials. At the same time, having a zero threshold caused erroneous feedback to influence the Actor. A threshold of 0.10 to 0.15 was recommended for the data analyzed in this chapter. It was also seen that the average and standard deviation alone were lacking the insights a distribution provided.

Chapter 4 Closed-Loop Experiments

Neural signals from the NAcc and MI from two marmoset monkeys were shown in Chapter 2. In Chapter 3 a CL architecture to receive signals from the MI as Actor inputs and NAcc as Critic inputs was developed. In this chapter, these two concepts are combined: MI from a marmoset monkey (Don) is used to drive the Actor and the NAcc from the same monkey is used to drive the Critic in CL control. Figure 4.1 shows the architecture for biological Actor-Critic RL which was implemented.

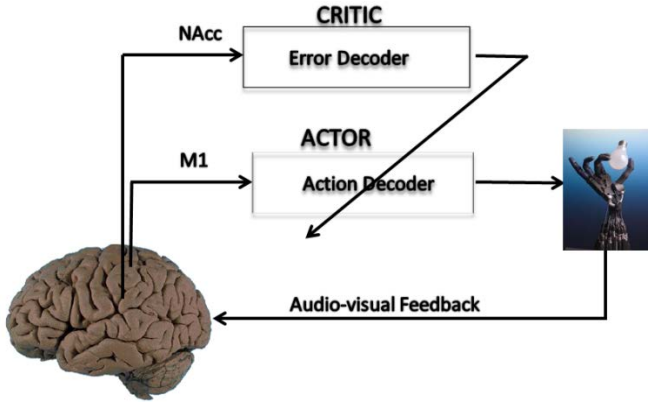

Figure 4.1: Architecture for Biological Actor-Critic Reinforcement Learning.
The Critic is controlled by the NAcc neural inputs and the Actor is controlled by the MI neural inputs. The Critic provides an evaluative feedback to the Actor.

4.1 Designing of the Closed-Loop Paradigm

The experimental setup was the same two-target reach paradigm as was presented in section 2.2.2. Instead the robot being controlled by the sensors pressed by the animal, it was controlled by the animal's neural signal via the HRL BMI. 1000 msec of data was binned 500 msec after the Go Tone from the MI array for the Actor input, as identified previously. Similarly, 1000 msec of data was binned at the start of the robot movement from NAcc array for the Critic input

and a Random Forests classifier was used as the Critic. The experiments were carried out according to the algorithm in Table 4-1. First, the results are confirmed by analyzing the data for Actor and Critic offline for the choice of window.

Table 4-1: Algorithm For The Closed-Loop HRL BMI

(1)	Use the past day's database of NAcc data to build the Critic Classifier
(2)	Initialize weights w_{ij} and learning rates, μ^+ and μ^-
(3)	Collect 1000 msec of MI data following 500 msec after Go Tone
(4)	Calculate the output based on Eq 3.14.
(5)	Execute Action
(6)	If within first 10 trials, set Epoch =0, go to step (7), else go to step (8)
(7)	Calculate reward (r) based on trial type. Set confidence =1 $(\rho = 1)$. Use all the past trials Update Weights based on Eq 3.15. Epoch++ Calculate the output based on Eq 3.14. If Epoch<10, repeat (7), else go to step (8)
(8)	Collect 1000 msec of NAcc data following 500 msec after the start of robot movement Decode the reward (r) and confidence (ρ) based on classifier built in step (1) Update Weights based on Eq 3.15. Calculate the output based on Eq 3.14.
(9)	Return to step (3)

4.1.1 Actor Neural Data

We used the data from the MI array recorded from the same day as that for which the Critic decoder was trained. There were 11 MI neuronal units and the number of units which showed significant difference for A trials (left hand movement) vs C trials (right hand movement) are shown in Table 4-2 for each window size analyzed (ANOVA with alpha 0.1). There were a total of 88 trials with 44 trials for each direction.

Table 4-2: Number Of Significant Units For MI Neurons (ANOVA, Alpha = 0.1)

	500 msec window					1000 msec window				
Window starting at	0	250	500	750	1000	0	250	500	750	1000
# of significant units	1	1	2	2	3	0	1	2	3	1

Figure 4.2 shows the average classification accuracy for 4 types of classifiers for the data shown above. From Table 4-2, we see that the number of units which show significant difference for the two classes classified is 2 out of 11 for the windows starting at 500 msec. Window 750-1750 msec have 3 out of 11 units significantly different for the two classes, however the accuracy is lower in this window. For this particular monkey, the best accuracy in classifying A trials (left arm movement) from C trials (right arm movement) is in the window starting at 500 msec after the Go Tone.

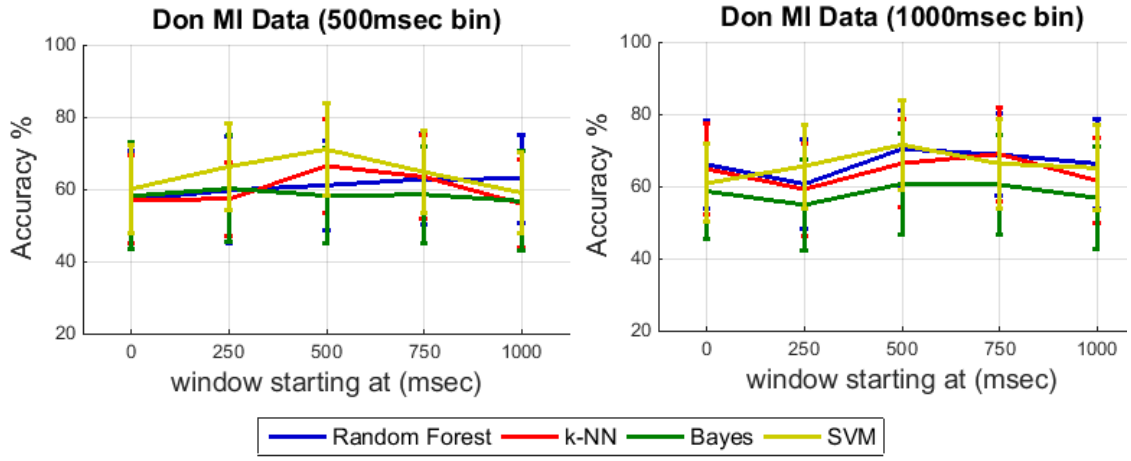

Figure 4.2: Average classification accuracy (100 iterations) for classifying A trials (left arm movement) and C trials (right arm movement) from MI neurons for different windows (A) 500 msec bin (B) 1000 msec bin. Each colored trace shows a different classification method.

4.1.2 Critic Classifier

The Critic was built using data from one day from the NAcc neural signals from the animal that the CL experiment was implemented on. There were 28 NAcc neuronal units and the number of units, which showed significant difference

for rewarding trials vs non-rewarding trials are shown in Table 4-3 for each window size analyzed (ANOVA with alpha 0.1).

Table 4-3: Number Of Significant Units For Nacc Neurons (ANOVA, Alpha = 0.1)

	500 msec window					1000 msec window				
Window starting at	0	250	500	750	1000	0	250	500	750	1000
# of significant units	0	0	5	5	1	1	3	0	1	1

The average classification accuracy for classifying rewarding trials and non-rewarding trials with 4 different classifiers are shown in Figure 4.3. The best accuracy here is shown at the window starting at 500 msec after the start of the robot movement. Random Forests classifier gave better accuracies on average and this was the classifier selected for the Critic. There were a total of 88 trials with 66 rewarding and 22 non-rewarding trials. The classes were balanced prior to training the classifier and 100 trees were used.

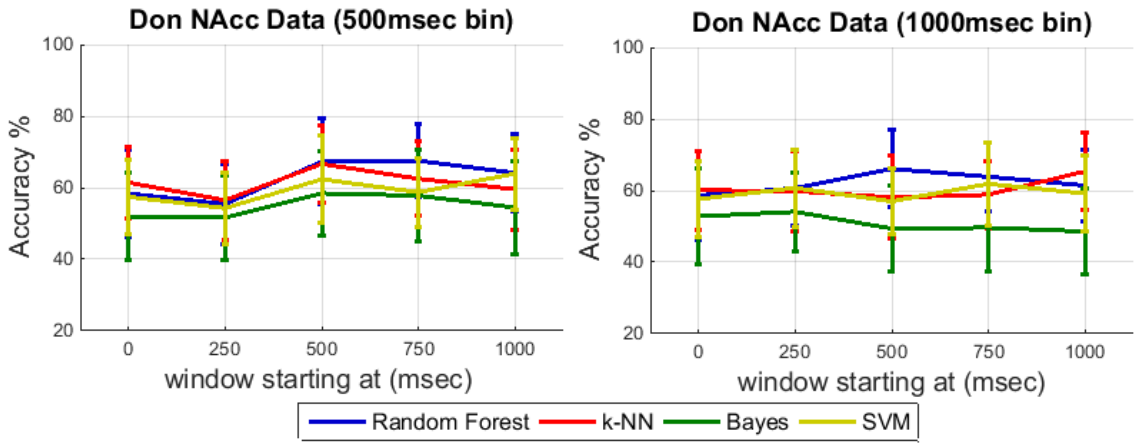

Figure 4.3: Average classification accuracy (100 iterations) for classifying rewarding trials and non-rewarding trials from NAcc neurons for different windows
(A) 500 msec bin (B) 1000 msec bin. Each colored trace shows a different classification method.

The test data for the classifier was given online during the experiment. The output from the Random Forests classifier (Critic) was the probability of the

trial being either a rewarding trial (robot correct action) or a non-rewarding trial (robot wrong action).

4.2 **Closed-Loop Experiment**

Three types of experiments were conducted changing the Critic for each of them. The first was a 100% accurate Critic given in a supervised manner. The purpose was to assess, given a perfect Critic, how well the performance is for this particular subject. Second and third experiments were using the NAcc as input to the Critic. In the second experiment the Actor used the Critic feedback (rewarding/non-rewarding) directly for the weight update, whereas for the third experiment, the Critic also gave a confidence value and the Actor took this into consideration when updating the weights. For the purpose of controllability, the second and third experiments were run in pairs; and each were conducted on the same day with the same initial conditions. The results of the experiments are shown here.

4.2.1 CL with a 100% accurate (artificial) Critic

Shown here are the results from the online CL experiment with 100% accurate (artificial) Critic feedback, starting with initial conditions from a previous week (Figure 4.4) and initial conditions from the previous session (Figure 4.5). We used this as a measure of evaluating if the temporal nature of the inputs required the weights to be initialized. The overall accuracy of the first system (weights from previous week) is 46% and the second system (weights from previous session) is 56%.

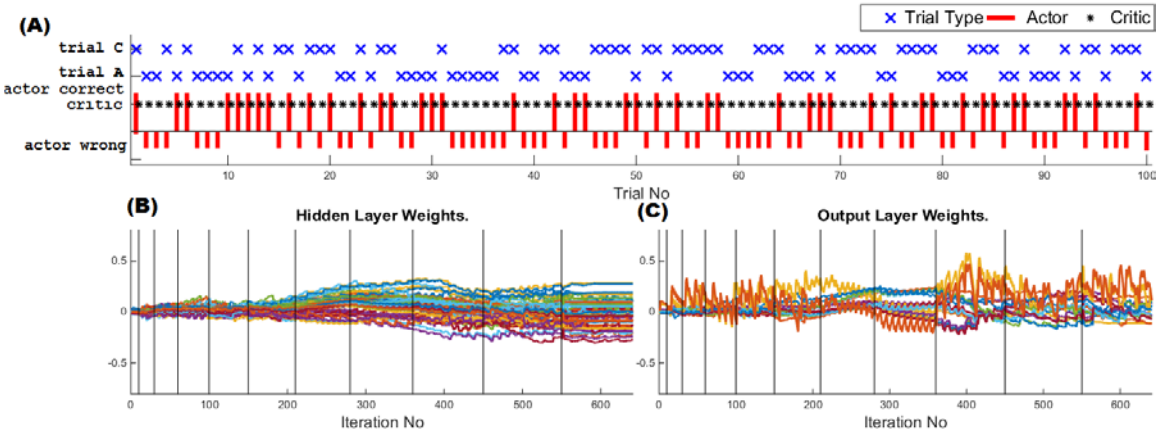

Figure 4.4: 100% accurate Critic CL experiment, with previous week's initial conditions. (A) Performance of each trial. (B) Hidden Layer Weights. (C) Output Layer Weights.

In addition, the weights are more stable in the second system. These results show the accuracy that can be achieved if the Critic is 100% accurate. These are two examples which were implemented in CL, but further analysis was done with different initial conditions similar to those done with the synthetic data.

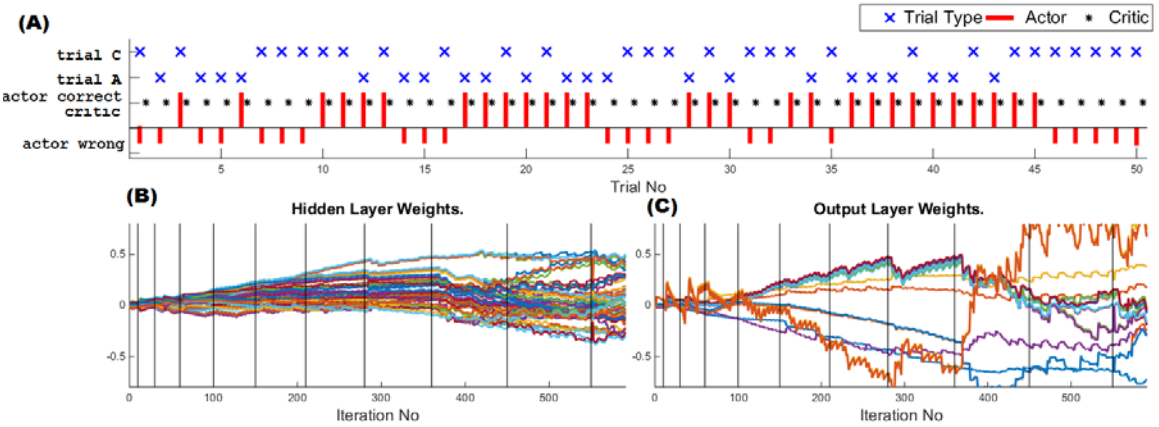

Figure 4.5: 100% accurate Critic CL experiment, with previous week's initial conditions. (A) Performance of each trial. (B) Hidden Layer Weights. (C) Output Layer Weights.

In Figure 4.6 the moving average per block of 20 trials is plotted (50% overlap). The system which started from more recent initial conditions (red) has higher accuracy overall. This is due to the unstationary nature of the neural data.

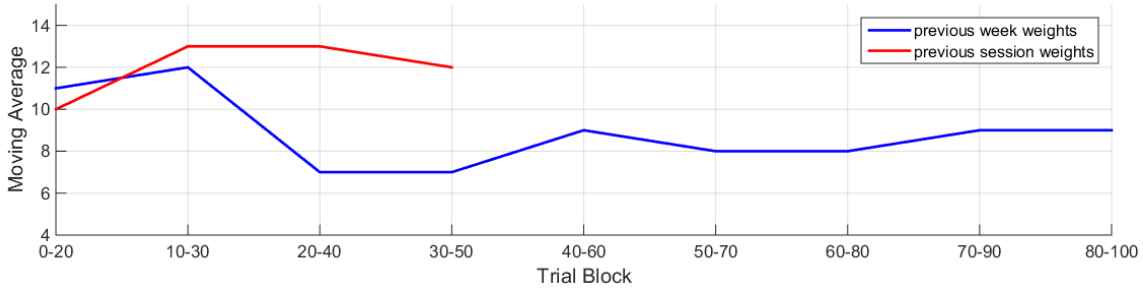

Figure 4.6: Accuracy (moving average) for the Perfect Critic CL experiment. Blue – with previous week's weights. Red – with previous session's weights. Red only up to 50 trials.

Next, the same data was run offline through an Actor-critc RL decoder with the same Critic feedback to arrive at a distribution of the performance. The results are shown in Figure 4.7. For the system with previous week's initial conditions overall performance on average was 55.2% (±3.02%) and for the system with the previous session initial conditions, the average performance was 57.9% (±5.75%). As discussed in the section introducing RL, it is seen that the system is susceptible to initial conditions and not all initial conditions have solutions that converge.

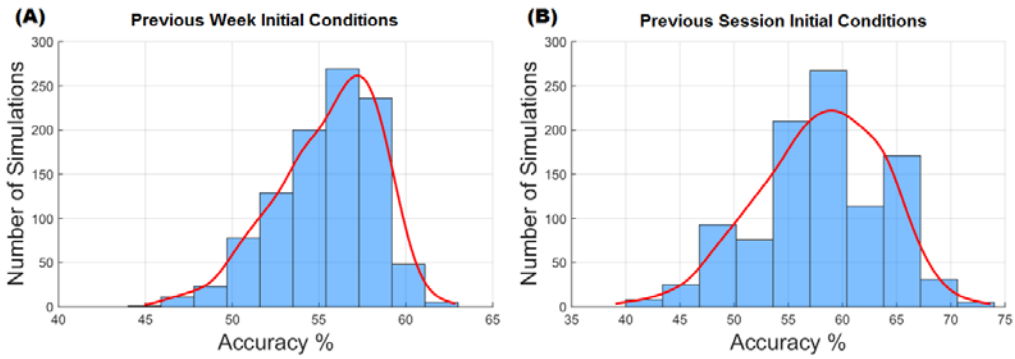

Figure 4.7: 100% accurate Critic CL experiment – offline simulations – distribution of overall accuracy. (A) with previpous week's initial conditions. (B) with previous session initial conditions.

4.2.2 CL with a NAcc Critic – Effect of threshold on Performance

Shown here are the results from the online CL experiment with NAcc as Critic input. Two cases are compared; Figure 4.8 (session 1) shows a system

with no threshold and Figure 4.9 (session 2) with a 0.1 threshold. The two sessions were started with the same initial conditions (initial conditions were decided from previous day's session) and the same Critic decoder for purposes of comparison. The Critic in session 1 was performing at 56% accuracy, while with a threshold, it increased to 60% in session 2. The 4% increase in Critic accuracy coupled with using this information intelligently, gave rise to an overall performance increase of 12% from 36% in session 1 to 48% in session 2 (See Figure 4.10 for detailed explanation). Subplot (A) shows the performance in each trial; red for Actor (positive – correct, negative – wrong), black for Critic (high – correct, low – wrong) and blue for type of trial (high – C trials, low – A trials). The trial types were balanced so as to not yield a skewed result. At the beginning (first 10 trials), both sessions perform at 50% accuracy, but as the session proceeds, session 1, is not able to maintain the performance. In contrast, session 2, picks up the performance as the session proceeds, especially after trial 35, where more trials are correct in a sequence (from both A and C).

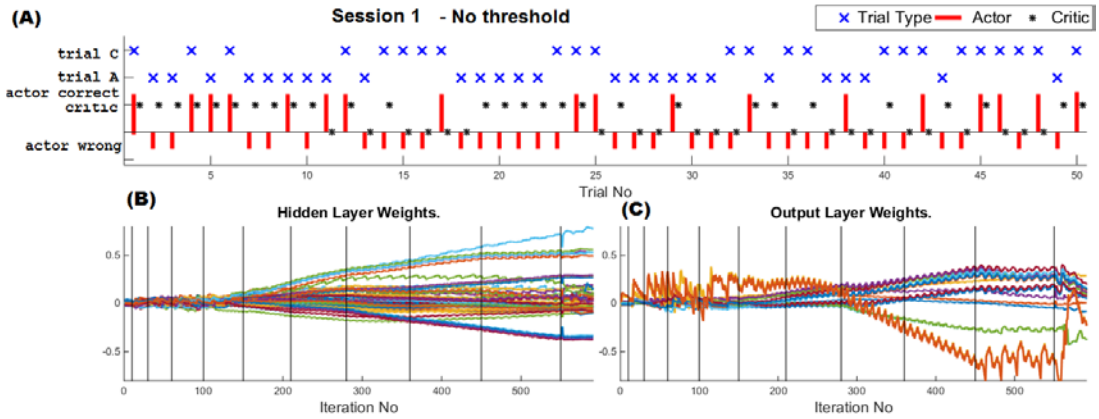

Figure 4.8: NAcc Critic CL experiment, with no threshold. (A) Performance of each trial. (B) Hidden Layer Weights. (C) Output Layer Weights.

Subplots (B) and (C) give the hidden layer weights and output layer weights for the respective sessions. Each vertical line corresponds to the weights at the end of the epoching, (i.e. weights corresponding to the particular robot movement). The weights of session 1, vary beyond ±0.5 whereas in session 2, the hidden layer weights vary between ±0.1 while the output layer weights vary between ±0.5. This suggests that session 2 is smoother than session 1. However, both systems have not converged to a solution in 50 trials. In session 1, the wrong Critic feedback is causing the overall performance to be very poor, whereas in session 2, the wrong Critic feedback has low confidence and therefore updates at a lower rate during these trials, causing the system adaptation being much slower and therefore the system will need longer to converge.

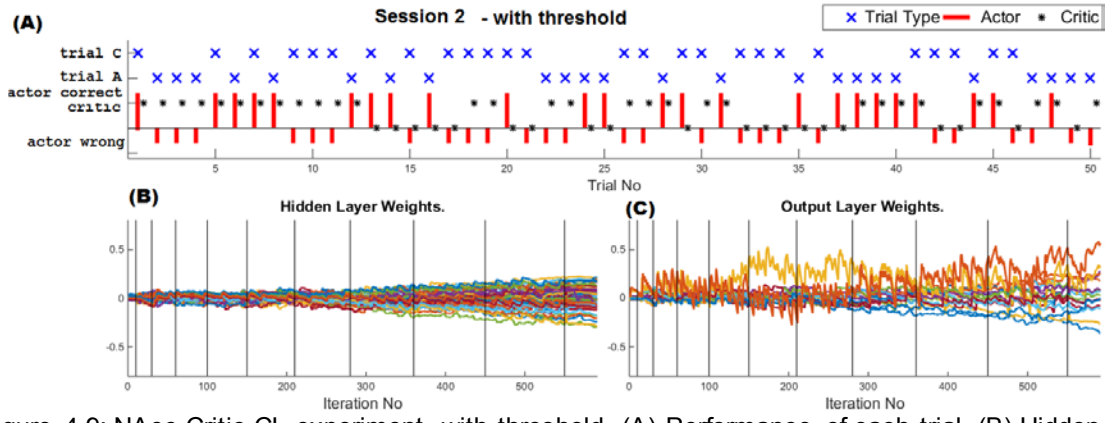

Figure 4.9: NAcc Critic CL experiment, with threshold. (A) Performance of each trial. (B) Hidden Layer Weights. (C) Output Layer Weights.

In Figure 4.10 the moving average is plotted. Here, the accuracy is considered in blocks of 20 with a sliding window of 10 (50% overlap). The system which had a threshold (session 2 – red) has higher accuracy overall.

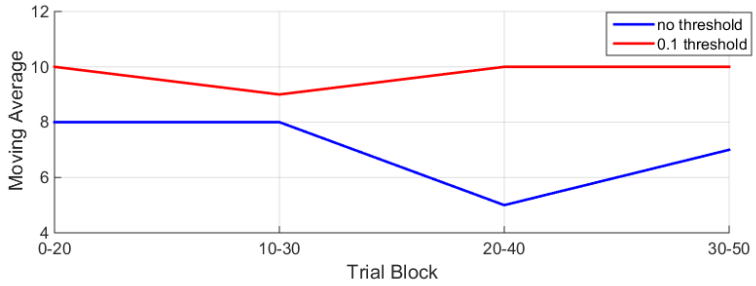

Figure 4.10: Accuracy (moving average) for the NAcc Critic CL experiment. Blue – with no threshold. Red – with threshold.

Figure 4.11 shows the data from session 1 (A) and session 2 (B) in offline simulations, if started with random initial weights, the distribution of the overall accuracy, given the Critic was perfect. Both systems range from 40s to 80s (x axis), peaking at 75% for session 1 and at 70% for session 2. This suggests that given, a perfect Critic, session 1 is able to perform approximately at 75% and session 2 at 70%. This does not however, take into account the erroneous nature of the Critic which lowers the performance.

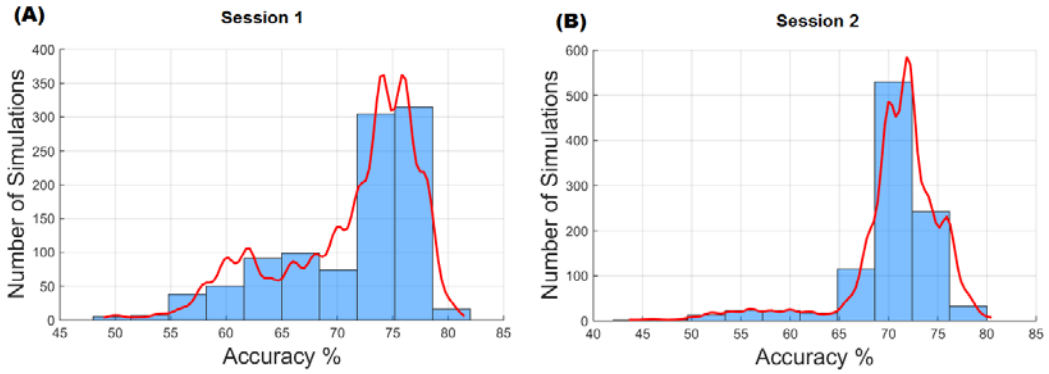

Figure 4.11: 100% accurate Critic CL experiment – offline simulations – distribution of overall accuracy. (A) for data in Figure 4.8. (B) for data in Figure 4.9.

Figure 4.12 shows the data from session 1 (A) and session 2 (B) in offline simulations, given the Critic performance at the same performance as the online CL experiment and starting with the same initial conditions as the online CL experiment. In session 1, it is seen that the distribution peaks at 50%, whereas

session 2 peaks at 55%-60%. Session 2 distribution has a lower variance (40.92) than session 1 (77.40).

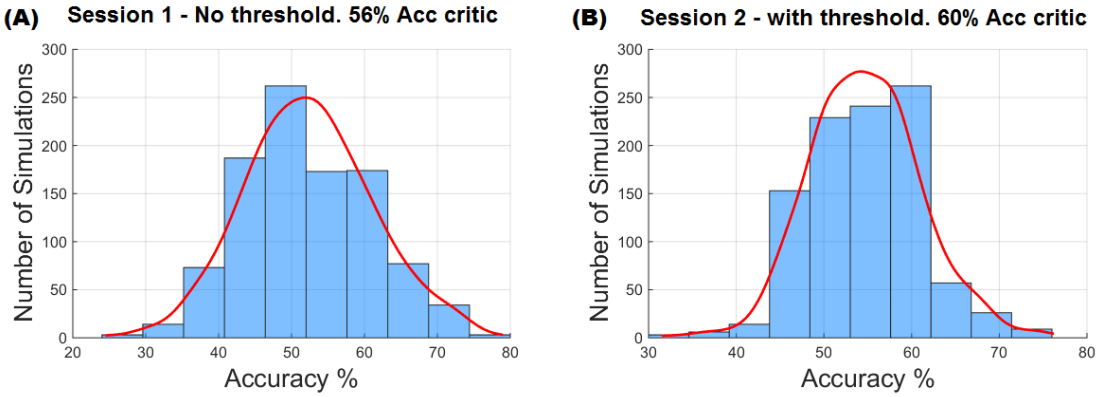

Figure 4.12: Inaccurate Critic CL experiment – offline simulations – distribution of overall accuracy. (A) for data in Figure 4.8. (B) for data in Figure 4.9.

4.3 **Summary and Conclusion**

In this chapter, it was shown how the confidence measure was extracted from the NAcc data and the accuracies of the different classifiers. CL experiments were designed such that a biological feedback can be used as the Critic signal. The Critic provided the confidence it had in the class label in addition to the class label. This was tested online with a NHP on a two choice decision making task. The first system was a Critic with a perfect feedback, which was done to get an idea of sensitivity to initial conditions and the performance levels that can be achieved with a perfect Critic. This system if initialized with appropriate initial conditions was able to perform 10% better than with inappropriate initial conditions. The final set of experiments was with a NAcc decoded Critic with no threshold and with 0.1 threshold. The system with a threshold performed 12% better than the system without a threshold. In OL simulations, the second system had lower variance than the first, and we

conclude that by using an appropriate threshold level, with a real Critic, the performance can be increased. Further, it was observed that the accuracy changes with the length of the session. Having a longer session, it was observed that the latter trials are decoded with higher accuracy than the earlier trials in the session. Inherently, the RL system performance increases with exploration (i.e. addition of experience/ trials).

Chapter 5 Summary and Future Work

5.1 Summary of Work

BMIs have great potential to alleviate lives of paralyzed individuals by electronically by passing the damaged areas and linking the neural signal to an actuator or one's own limbs. Most of the current BMIs are trained in a supervised manner with a training signal. A class of decoders, that learn from the environment (known as RL decoders) have been adapted to BMI field. Since it learns by exploring its environment, it does not need a supervised training signal. It has been hypothesized that the NAcc can provide a reward signal which is similar to the TD error in RL decoders. Proof of concept had been shown in previous work of using this signal as a feedback in RLBMI. An Actor-Critic RL paradigm had been proposed and here it was implemented in CL with a monkey using a real Critic feedback.

In this work, the nature of the NAcc signal was studied and a detailed analysis of the NAcc neural signals was performed. Different machine learning techniques were applied to the NAcc data for classification of rewarding and non-rewarding trials. The purpose was to build confidence of using this signal as an input to the Critic in a Actor-Critic RL paradigm. However, since the classification accuracy was less than perfect, and since the Critic feedback limited the overall performance, a new paradigm was needed to deal with this.

One of the hypotheses of this work was that if the Critic could provide the accuracy of its own feedback and it was used intelligently, the overall performance can be improved. This was shown to be true by synthetic data simulations; using the feedback information intelligently and updating only when the feedback is accurate, improved the system performance. This was particularly true for lower Critic accuracies since at lower Critic accuracies the information provided by the Critic in the old system was wrong most of the time.

Next it was shown how the Critic can give the accuracy of its own feedback. For proof of concept, a graphical solution was utilized, where the distance to the boundary was used as a measure of the accuracy. This was converted to a confidence metric and given out of the Critic along with the rewar signal. The assumption here was that closer to the decision margin, there are more misclassifications. This was implemented in an offline experiment with synthetically generated MI and NAcc data.

Other methods can also be used to give a confidence measure. For example, most probability based classifiers decide the label of the data point based on the probability that it belongs to one class or the other. If the difference in probability is high, this suggests that the decision is made with higher confidence and vice versa. Random Forests Classification was used for the NAcc data and implemented in CL with a marmoset monkey. Three systems were compared in CL experiments: 100% accurate artificial Critic, real NAcc Critic with and without a confidence.

In conclusion, if a feedback signal is to be derived from the brain, due to non-stationarity and inherent noise issues, a method needs to be formulated of knowing how accurate this feedback is. If such a confidence metric can be attached to the feedback, then using this in an Actor-Critic RL can improve performance of the system.

5.2 Novel Contributions and Implications

The following contributions were made by this work.

5.2.1 A Confidence Metric can be Attached to the NAcc Neural Signal

Given that the NAcc neurons have a wealth of information it can be hard to extract the right kind of information. Even though only 10-20% of the neurons modulated with reward, it was still possible to extract the reward information with 66% accuracy on average with classifiers which were able to give a confidence metric in addition to the class label. For the ease of visualizing, the normalized distance to the boundary was used as the confidence measure. A LDA is the best example of a classifier where the distance to the boundary can be used. Other classifiers with a decision boundary can also utilize this measure. However, in probabilistic classifiers such as Bayes and Random Forests, the confidence measure is the arrived at from the probability the data belongs to the particular class. Both the distance to the boundary and the probabilistic measure were shown in this work as methods of extracting the confidence from the data.

5.2.2 Confidence Measure Improves the Overall Performance

It was confirmed in this work, that using a confidence measure in the feedback signal can improve the overall performance. The use of the confidence measure in feedback is not only limited to applications of RLBMI. A BMI that uses any form of less-than-perfect feedback can utilize this confidence metric. This can also be extended on to other machine learning techniques where feedback is less-than-perfect.

5.2.3 Developed a Paradigm for Real Time Implementation

Showing proof of concept of the confidence measure working is the first step. The next step of implementing this in CL, have a series of mini challenges in-between. It is important to tackle each of these mini challenges in order to have a working CL system. These challenges were addressed and the system was implemented in real time. One of them was the implementation of a supervised decoder at the beginning of the session to tackle the inherent slow learning of the RL system. In this work, a complete paradigm was developed where the proposed system was implemented in real time with supervised learning at the beginning and RL after the first 10 trials.

5.2.4 A Closed-Loop BMI with Motor Control and Biologic Feedback

A closed-loop BMI that uses motor control and biologic reward feedback was designed, prototyped and tested on a NHP. The experiment paradigm was designed in house with sensors and all components needed for the experiment including hardware and software codes. Both the motor control and biological reward feedback were tested.

5.3 **Improvements and Future Work**

SNR in neural signals is very low. There were different techniques that were used (grounding, CAR) to increase the SNR. However, despite, all these efforts, biological signals are noisy. As neural engineers, it is a challenge to design a system robust to noise, most of which are inherently is unpredictable. Due to this, extracting the necessary features from data becomes cumbersome, and the results can vary depending on the noise level. Another outcome of the noise is the misclassification of data, thus the basis for this work. It is because of such data that the confidence measure adds value. However, getting the confidence from the data itself has different challenges associated with it. In this work, two methods (distance to boundary and probability of class) were implemented. Other methods of extracting the confidence measure needs to be explored and this will differ from application to application.

Biological signals are inherently non-stationary. In the brain, the statistics of the neural signals constantly change over time. This is the biggest challenge that neural engineers face. This same challenge, applies to the data that was analyzed in this work. Given the nature of the animal model in this work, the sessions were of limited time, usually 20-40 min per session. For longer sessions it is possible that the statistics of the beginning of the session and the end of the session is different. However, one of the advantages of RL is its ability to adapt to changes in the environment, which includes the input neural data non-stationarity. Therefore, the Actor is able to adapt to changes in the neural input.

However, the Critic implemented was a static decoder. There were several experimental parameters which needed to be paid attention to in order to do this. First is that the number of neuronal units recorded for both from MI and NAcc needed to be the same across sessions if they were to be combined. In addition to the number of units being the same, it was important to assess if the electrodes were picking up the same unit across days. In most cases, it was difficult to find a two week period with the same units. Unless this was the case, we tried not to combine data sets.

Combining data sets became a necessity out of the animal model chosen. Unlike rhesus macaques, our marmosets, worked less time (<1 hour with marmosets as compared to 3 hours with rhesus macaques). Therefore the number of trials per day is also low. Particularly due to this, combining data sets from different days became a necessity but also needed careful consideration. For the CL experiments, we gathered data and implemented on the same week. For the data analysis in Chapter 2, we combined several days/ sessions of data. This is one of the reasons for the high variance seen in the data.

As seen in Chapter 4, the RL is sensitive to initial conditions. It is also known that certain initial conditions do not have a solution that converges, and therefore several Monte Carlo simulations need to be performed to find optimal initial conditions (See Figure 3 B in [112] and Figure 7 in [119]). Preliminary tests were done to test the initial condition comparison and how much the performance was affected due to this. Comparing the statistics of these initial weights showed no particular indication as to which weights to choose. Therefore, further studies

are needed to be able to decide upon initial conditions that are optimal to be used.

The animal experiments that were conducted assumed that the animal always performs the correct action. Although in a few trials per session, it is possible that the animal either performs the wrong action or is uninterested or not engaged with the task. These trials can be removed from offline analysis and if the animal performs the wrong choice too often, the entire session can be removed. However, in CL experiments, this cannot be done and the fundamental assumption is that the animal is correct. When performed CL experiments with low performance for a few days, the animal is confused and its performance drops. This was the case in a few of the CL experiments and these sessions have not been included in the results. The trial type given did not match the animal's performance. Even though the robot mimicked the animal's performance, the system considers this to be a wrong action when calculating performance. This indicates that the animal needs to be re-trained before conducting CL experiments again, prolonging experimental time. If a paradigm can be designed and controlled such that when the robot matched the aninmal's action, it will always be considered a correct action, even though the trial type did not match, this would reduce additional extensive training needed.

The contribution of electrodes to decoding was not focused in this work. However, with earlier animals, we had issues due to design flaws which we discussed with suppliers and were able to resolve. Earlier on during the animal's recordings, the signals change a lot. However, as the implant time increased, the

signals become more stable than at first, but due to encapsulation, the signal quality also reduces. This gives a window of optimal time after implantation which needs to be exploited.

One of the main concerns of this work is for better targeting of MI and NAcc implant sites. The animal that the CL experiments were conducted on, has not been sacrificed at the time of writing this work and histology reports have not been received. We are fairly certain that the electrodes are in the targeted locations as the procedure has been modified from previous work, but can be further improved with imaging techniques.

At present the Actor algorithm is a HRL decoder which has its limitations. The Actor-Critic RL paradigm is flexible to include any decoder in to the Actor and the feedback/ error/ reward can be provided by the Critic. This work was limited to testing the system with the HRL Actor, but other avenues can be explored for different results.

One of the challenges that was encountered was the inherent slow adaptation of the RL paradigm. This was overcome by adding memory and epoching to each trial. This was feasible in offline experiments. However, when implementing online, the epoching could not be done due to lack of a matching Critic output. Hence, we resolved to use the first 10 trials as a supervised training for the system to get stabilized. However, in some cases, the system was not converged at the end of 10 trials causing performance to drop. This supervised training trial quantity at the beginning can be changed and further analysis can

be done. The supervised training at the beginning can be used to build the Critic classifier instead of needing a past day's database of trials.

The HRL decoder has an inbuilt feature where the adaptation stops when the weights are stabilized. However, when the system performs poorly, it takes a very long time for the RL to get the weights back on track, given the added slow adaptation due to the confidence metric. When the performance of the system is dropped below a certain percentage, supervised learning with epoching can be introduced to boost the performance.

References

[1] "One degree of separation | Paralysis and spinal cord injury in the United States," Christopher & Dana Reeve Foundation2009.

[2] (2012-13, Aug 14). *Sci-info-pages. Quadriplegic, Paraplegic & Caregiver Resources*. Available: http://www.sci-info-pages.com/facts.html

[3] K. D. Anderson, "Targeting recovery: priorities of the spinal cord-injured population," *Journal of Neurotrauma,* vol. 21, pp. 1371-1383, 2004.

[4] C. T. Moritz, S. I. Perlmutter, and E. E. Fetz, "Direct control of paralysed muscles by cortical neurons," *Nature,* vol. 456, pp. 639-642, 2008.

[5] L. A. Farwell and E. Donchin, "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials," *Electroencephalography and Clinical Neurophysiology,* vol. 70, pp. 510-523, 1988.

[6] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, "Cortical control of a prosthetic arm for self-feeding," *Nature,* vol. 453, pp. 1098-1101, 2008.

[7] J. R. Wolpaw and D. J. McFarland, "Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 101, pp. 17849-17854, 2004.

[8] L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan*, et al.*, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," *Nature,* vol. 442, pp. 164-171, 2006.

[9] J. L. Collinger, B. Wodlinger, J. E. Downey, W. Wang, E. C. Tyler-Kabara, D. J. Weber*, et al.*, "High-performance neuroprosthetic control by an individual with tetraplegia," *The Lancet,* 2012.

[10] W. Wang, J. L. Collinger, A. D. Degenhart, E. C. Tyler-Kabara, A. B. Schwartz, D. W. Moran*, et al.*, "An electrocorticographic brain interface in an individual with tetraplegia," *PloS one,* vol. 8, p. e55344, 2013.

[11] J. C. Sanchez and J. C. Principe, *Brain-Machine Interaction Engineering* vol. 17: Morgan & Claypool Publishers, 2007.

[12] J. M. Carmena, M. A. Lebedev, C. S. Henriquez, and M. A. Nicolelis, "Stable ensemble performance with single-neuron variability during reaching movements in primates," *The Journal of Neuroscience,* vol. 25, pp. 10712-10716, 2005.

[13] R. D. Flint, Z. A. Wright, M. R. Scheid, and M. W. Slutzky, "Long term, stable brain machine interface performance using local field potentials and multiunit spikes," *Journal of Neural Engineering,* vol. 10, p. 056005, 2013.

[14] K. C. Kowalski, B. D. He, and L. Srinivasan, "Dynamic analysis of naive adaptive brain-machine interfaces," *Neural Computation,* vol. 25, pp. 2373-2420, 2013.

[15] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, and et al., "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature,* vol. 408, pp. 361-5, 2000.

[16] M. A. Lebedev, J. M. Carmena, J. E. O'Doherty, M. Zacksenhouse, C. S. Henriquez, J. C. Principe*, et al.*, "Cortical ensemble adaptation to represent velocity of an artificial actuator controlled by a brain-machine interface," *The Journal of Neuroscience,* vol. 25, pp. 4681-4693, 2005.

[17] J. M. Carmena, M. A. Lebedev, R. E. Crist, J. E. O'Doherty, D. M. Santucci, D. F. Dimitrov*, et al.*, "Learning to control a brain–machine interface for reaching and grasping by primates," *PLoS Biology,* vol. 1, p. e42, 2003.

[18] M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue, "Brain-machine interface: Instant neural control of a movement signal," *Nature,* vol. 416, pp. 141-142, 2002.

[19] J. P. Donoghue, "Connecting cortex to machines: recent advances in brain interfaces," *Nature Neuroscience,* vol. 5, pp. 1085-1088, 2002.

[20] M. J. Black, E. Bienenstock, J. P. Donoghue, M. Serruya, W. Wu, and Y. Gao, "Connecting brains with machines: the neural control of 2d cursor movement," in *Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on*, 2003, pp. 580-583.

[21] L. Paninski, M. R. Fellows, N. G. Hatsopoulos, and J. P. Donoghue, "Spatiotemporal tuning of motor cortical neurons for hand position and velocity," *Journal of Neurophysiology,* vol. 91, pp. 515-532, 2004.

[22] D. M. Taylor, S. I. H. Tillery, and A. B. Schwartz, "Direct cortical control of 3D neuroprosthetic devices," *Science,* vol. 296, pp. 1829-1832, 2002.

[23] A. B. Schwartz, X. T. Cui, D. J. Weber, and D. W. Moran, "Brain-controlled interfaces: movement restoration with neural prosthetics," *Neuron,* vol. 52, pp. 205-220, 2006.

[24] A. B. Schwartz, "Cortical neural prosthetics," *Annu. Rev. Neurosci.,* vol. 27, pp. 487-507, 2004.

[25] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner, "Neuronal population coding of movement direction," *Science,* vol. 233, pp. 1416-1419, 1986.

[26] A. P. Georgopoulos, R. E. Kettner, and A. B. Schwartz, "Primate motor cortex and free arm movements to visual targets in three-dimensional space. Ii. Coding of the direction of movement by a neuronal population," *The Journal of Neuroscience,* vol. 8, pp. 2928-2937, 1988.

[27] W. Wu, M. Black, Y. Gao, E. Bienenstock, M. Serruya, and J. Donoghue, "Inferring hand motion from multi-cell recordings in motor cortex using a Kalman filter," in *SAB'02-Workshop on Motor Control in Humans and Robots: On the Interplay of Real Brains and Artificial Devices*, 2002, pp. 66-73.

[28] W. Wu, M. J. Black, D. Mumford, Y. Gao, E. Bienenstock, and J. P. Donoghue, "Modeling and decoding motor cortical activity using a switching Kalman filter," *Biomedical Engineering, IEEE Transactions on,* vol. 51, pp. 933-942, 2004.

[29] W. Wu, Y. Gao, E. Bienenstock, J. P. Donoghue, and M. J. Black, "Bayesian population decoding of motor cortical activity using a Kalman filter," *Neural Computation,* vol. 18, pp. 80-118, 2006.

[30] M. Y. Byron, C. Kemere, G. Santhanam, A. Afshar, S. I. Ryu, T. H. Meng*, et al.*, "Mixture of trajectory models for neural decoding of goal-directed movements," *Journal of Neurophysiology,* vol. 97, pp. 3763-3780, 2007.

[31] J. P. Cunningham, P. Nuyujukian, V. Gilja, C. A. Chestek, S. I. Ryu, and K. V. Shenoy, "A closed-loop human simulator for investigating the role of feedback control in brain-machine interfaces," *Journal of Neurophysiology,* vol. 105, pp. 1932-1949, 2011.

[32] D. Sussillo, P. Nuyujukian, J. M. Fan, J. C. Kao, S. D. Stavisky, S. Ryu*, et al.*, "A recurrent neural network for closed-loop intracortical brain–machine interface decoders," *Journal of Neural Engineering,* vol. 9, p. 026027, 2012.

[33] S. Kim, J. Sanchez, Y. Rao, D. Erdogmus, J. Carmena, M. Lebedev*, et al.*, "A comparison of optimal MIMO linear and nonlinear models for brain–machine interfaces," *Journal of Neural Engineering,* vol. 3, p. 145, 2006.

[34] S.-P. Kim, J. D. Simeral, L. R. Hochberg, J. P. Donoghue, and M. J. Black, "Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia," *Journal of Neural Engineering,* vol. 5, p. 455, 2008.

[35] L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel*, et al.*, "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm," *Nature,* vol. 485, pp. 372-375, 2012.

[36] V. Gilja, P. Nuyujukian, C. A. Chestek, J. P. Cunningham, B. M. Yu, J. M. Fan*, et al.*, "A brain machine interface control algorithm designed from a feedback control perspective," in *34th Annual International Conference of the IEEE EMBS*, San Diego, California USA, 2012.

[37] J. C. Sanchez and J. C. Principe, "Prerequesites for symbiotic brain-machine interfaces," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, 2009, pp. 1736-1741.

[38] J. M. Fuster, "Upper processing stages of the perception–action cycle," *Trends in Cognitive Sciences,* vol. 8, pp. 143-145, 2004.

[39] J. R. Hollerman, L. Tremblay, and W. Schultz, "Involvement of basal ganglia and orbitofrontal cortex in goal-directed behavior," *Progress in Brain Research,* vol. 126, pp. 193-215, 2000.

[40] G. Montagne, M. Buekers, C. Camachon, A. De Rugy, and M. Laurent, "The learning of goal-directed locomotion: A perception-action perspective," *The Quarterly Journal of Experimental Psychology A: Human Experimental Psychology,* 2003.

[41] K. Doya, "What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?," *Neural Networks,* vol. 12, pp. 961-974, 1999.

[42] K. Doya, "Complementary roles of basal ganglia and cerebellum in learning and motor control," *Current Opinion in Neurobiology,* vol. 10, pp. 732-739, 2000.

[43] A. Parent and L. N. Hazrati, "Functional anatomy of the basal ganglia. I. The cortico-basal ganglia-thalamo-cortical loop," *Brain Research Reviews,* vol. 20, pp. 91-127, 1995.

[44] R. A. Wise and P.-P. Rompré, "Brain dopamine and reward," *Annual Review of Psychology,* vol. 40, pp. 191-225, 1989.

[45] T. E. Schlaepfer, M. X. Cohen, C. Frick, M. Kosel, D. Brodesser, N. Axmacher*, et al.*, "Deep brain stimulation to reward circuitry alleviates anhedonia in refractory major depression," *Neuropsychopharmacology,* vol. 33, pp. 368-377, 2007.

[46] H. Mayberg, "Modulating limbic-cortical circuits in depression: targets of antidepressant treatments," in *Seminars in Clinical Neuropsychiatry*, 2002, pp. 255-268.

[47] A. G. Phillips, "Brain reward circuitry: a case for separate systems," *Brain Research Bulletin,* vol. 12, pp. 195-201, 1984.

[48] P. Apicella, T. Ljungberg, E. Scarnati, and W. Schultz, "Responses to reward in monkey dorsal and ventral striatum," *Experimental Brain Research,* vol. 85, pp. 491-500, 1991.

[49] J. R. Hollerman, L. Tremblay, and W. Schultz, "Influence of reward expectation on behavior-related neuronal activity in primate striatum," *Journal of Neurophysiology,* vol. 80, pp. 947-963, 1998.

[50] S. M. Nicola, "The nucleus accumbens as part of a basal ganglia action selection circuit," *Psychopharmacology,* vol. 191, pp. 521-550, 2007.

[51] C. Pennartz, H. J. Groenewegen, and F. Da Silva, "The nucleus accumbens as a complex of functionally distinct neuronal ensembles: an integration of behavioural, electrophysiological and anatomical data," *Progress in Neurobiology,* vol. 42, pp. 719-761, 1994.

[52] E. Rolls, M. Burton, and F. Mora, "Neurophysiological analysis of brain-stimulation reward in the monkey," *Brain Research,* vol. 194, pp. 339-357, 1980.

[53] A. G. Barto, "Adaptive critics and the basal ganglia," *Models of Information Processing in the Basal Ganglia,* p. 215, 1995.

[54] B. Mahmoudi and J. C. Sanchez, "A symbiotic brain-machine interface through value-based decision making," *PloS One,* vol. 6, p. e14760, 2011.

[55] B. Mahmoudi, E. Pohlmeyer, A. Prasad, and J. C. Sanchez, "Brain-machine interfaces in activities of daily living: innovating a new roadmap for experimentation," *IEEE,* 2011.

[56] B. W. Balleine and A. Dickinson, "Goal-directed instrumental action: contingency and incentive learning and their cortical substrates," *Neuropharmacology,* vol. 37, pp. 407-419, 1998.

[57] Y. Niv, "Reinforcement learning in the brain," *Journal of Mathematical Psychology,* vol. 53, pp. 139-154, 2009.

[58] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction* vol. 1: Cambridge Univ Press, 1998.

[59] J. F. DiGiovanna, "Changing the brain-machine interface paradigm: co-adaptation based on reinforcement learning," Doctoral Dissertation, Dept. Biomedical Eng, University of Florida, Gainesville, 2008.

[60] B. Mahmoudi, "Integrating robotic action with biologic perception: a brain-machine symbiosis theory," Doctoral Dissertation, Dept. Biomedical Eng, University of Florida, Gainesville, 2010.

[61] J. DiGiovanna, B. Mahmoudi, J. Fortes, J. C. Principe, and J. C. Sanchez, "Coadaptive brain–machine interface via reinforcement learning," *Biomedical Engineering, IEEE Transactions on,* vol. 56, pp. 54-64, 2009.

[62] E. A. Pohlmeyer, B. Mahmoudi, G. Shijia, N. Prins, and J. C. Sanchez, "Brain-machine interface control of a robot arm using actor-critic reinforcement learning," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 2012, pp. 4108-4111.

[63] B. A. S. Hasan and J. Q. Gan, "Temporal modeling of EEG during self-paced hand movement and its application in onset detection," *Journal of Neural Engineering,* vol. 8, p. 056015, 2011.

[64] S. G. Mason and G. E. Birch, "A brain-controlled switch for asynchronous control applications," *Biomedical Engineering, IEEE Transactions on,* vol. 47, pp. 1297-1307, 2000.

[65] Z. C. Chao, Y. Nagasaka, and N. Fujii, "Long-term asynchronous decoding of arm motion using electrocorticographic signals in monkeys," *Frontiers in Neuroengineering,* vol. 3, 2010.

[66] Z. Wang, A. Gunduz, P. Brunner, A. L. Ritaccio, Q. Ji, and G. Schalk, "Decoding onset and direction of movements using Electrocorticographic (ECoG) signals in humans," *Front Neuroeng,* vol. 5, p. 15, 2012.

[67] S. H. Lee, K. Choi, S. Jeong, J. S. Kim, and C. K. Chung, "Classifying ECoG signals prior to voluntary movement onset," in *Brain-Computer Interface (BCI), 2013 International Winter Workshop on*, 2013, pp. 36-38.

[68] J. J. Williams, A. G. Rouse, S. Thongpang, J. C. Williams, and D. W. Moran, "Differentiating closed-loop cortical intention from rest: building an asynchronous electrocorticographic BCI," *Journal of neural engineering,* vol. 10, p. 046001, 2013.

[69] F. Galán, M. Nuttin, E. Lew, P. W. Ferrez, G. Vanacker, J. Philips*, et al.*, "A brain-actuated wheelchair: asynchronous and non-invasive brain–computer interfaces for continuous control of robots," *Clinical Neurophysiology,* vol. 119, pp. 2159-2169, 2008.

[70] J. F. Borisoff, S. G. Mason, A. Bashashati, and G. E. Birch, "Brain-computer interface design for asynchronous control applications: improvements to the LF-ASD asynchronous brain switch," *Biomedical Engineering, IEEE Transactions on,* vol. 51, pp. 985-992, 2004.

[71] N. Achtman, A. Afshar, G. Santhanam, M. Y. Byron, S. I. Ryu, and K. V. Shenoy, "Free-paced high-performance brain–computer interfaces," *Journal of Neural Engineering,* vol. 4, p. 336, 2007.

[72] A. M. Graybiel, T. Aosaki, A. W. Flaherty, and M. Kimura, "The basal ganglia and adaptive motor control," *Science,* vol. 265, pp. 1826-1831, 1994.

[73] E. V. Evarts and J. Tanji, "Gating of motor cortex reflexes by prior instruction," *Brain Research,* vol. 71, pp. 479-494, 1974.

[74] Y. Kobayashi and T. Isa, "Sensory-motor gating and cognitive control by the brainstem cholinergic system," *Neural Networks,* vol. 15, pp. 731-741, 2002.

[75] R. Kaji, "Basal ganglia as a sensory gating devise for motor control," *Journal of Medical Investigation,* vol. 48, pp. 142-146, 2001.

[76] G. Abbruzzese and A. Berardelli, "Sensorimotor integration in movement disorders," *Movement Disorders,* vol. 18, pp. 231-240, 2003.

[77] K. Mansfield, "Marmoset models commonly used in biomedical research," *Comparative Medicine,* vol. 53, pp. 383-392, 2003.

[78] N. Kishi, K. Sato, E. Sasaki, and H. Okano, "Common marmoset as a new model animal for neuroscience research and genome editing technology," *Development, Growth & Differentiation,* vol. 56, pp. 53-62, 2014.

[79] H. Okano, K. Hikishima, A. Iriki, and E. Sasaki, "The common marmoset as a novel animal model system for biomedical and neuroscience research applications," in *Seminars in Fetal and Neonatal Medicine*, 2012, pp. 336-340.

[80] C. D. Hardman and K. W. Ashwell, *Stereotaxic and Chemoarchitectural Atlas of the Brain of the Common Marmoset (Callithrix jacchus)*: CRC Press, 2012.

[81] J. D. Newman, W. M. Kenkel, E. C. Aronoff, N. A. Bock, M. R. Zametkin, and A. C. Silva, "A combined histological and MRI brain atlas of the common marmoset monkey, Callithrix jacchus," *Brain Research Reviews,* vol. 62, pp. 1-18, 2009.

[82] X. Palazzi and N. Bordier, *The Marmoset Brain in Stereotaxic Coordinates*: Springer, 2009.

[83] G. Paxinos, C. Watson, M. Petrides, M. Rosa, and H. Tokuno, *The Marmoset Brain in Stereotaxic Coordinates*: Academic Press, 2012.

[84] H. Stephan, G. Baron, and W. K. Schwerdtfeger, *The Brain of the Common Marmoset (Callithrix jacchus): A Stereotaxic Atlas*: Springer-Verlag Berlin:, 1980.

[85] H. Tokuno, I. Tanaka, Y. Umitsu, T. Akazawa, and Y. Nakamura, "Web-accessible digital brain atlas of the common marmoset (Callithrix jacchus)," *Neuroscience Research,* vol. 64, pp. 128-131, 2009.

[86] H. Tokuno, I. Tanaka, Y. Umitsu, and Y. Nakamura, "Stereo Navi 2.0: Software for stereotaxic surgery of the common marmoset (Callithrix jacchus)," *Neuroscience Research,* vol. 65, pp. 312-315, 2009.

[87] S. Yuasa, K. Nakamura, S. Kohsaka, and K. S. S. Sentā, *Stereotaxic Atlas of the Marmoset Brain: With Immunohistochemical Architecture and MR Images*: National Institute of Neuroscience, National Center of Neurology and Psychiatry, Japan, 2010.

[88] D. Cyranoski, "Marmosets are stars of Japan's ambitious brain project," *Nature,* vol. 514, pp. 151-152, 2014.

[89] K. A. Ludwig, J. D. Uram, J. Yang, D. C. Martin, and D. R. Kipke, "Chronic neural recordings using silicon microelectrode arrays electrochemically deposited with a poly (3, 4-ethylenedioxythiophene)(PEDOT) film," *Journal of Neural Engineering,* vol. 3, p. 59, 2006.

[90] K. A. Ludwig, R. M. Miriani, N. B. Langhals, M. D. Joseph, D. J. Anderson, and D. R. Kipke, "Using a common average reference to improve cortical neuron recordings from microelectrode arrays," *Journal of Neurophysiology,* vol. 101, pp. 1679-1689, 2009.

[91] A. Prasad and J. C. Sanchez, "Quantifying long-term microelectrode array functionality using chronic in vivo impedance testing," *Journal of Neural Engineering,* vol. 9, p. 026028, 2012.

[92] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network: Computation in Neural Systems,* vol. 9, pp. R53-R78, 1998.

[93] M. A. Nicolelis, D. Dimitrov, J. M. Carmena, R. Crist, G. Lehew, J. D. Kralik*, et al.*, "Chronic, multisite, multielectrode recordings in macaque monkeys," *Proceedings of the National Academy of Sciences,* vol. 100, pp. 11041-11046, 2003.

[94] F. Wood, M. J. Black, C. Vargas-Irwin, M. Fellows, and J. P. Donoghue, "On the variability of manual spike sorting," *Biomedical Engineering, IEEE Transactions on,* vol. 51, pp. 912-918, 2004.

[95] O. Bai, P. Lin, D. Huang, D.-Y. Fei, and M. K. Floeter, "Towards a user-friendly brain–computer interface: Initial tests in ALS and PLS patients," *Clinical Neurophysiology,* vol. 121, pp. 1293-1303, 2010.

[96] N. W. Prins, S. Geng, E. A. Pohlmeyer, A. Prasad, and J. C. Sanchez, "Representation of natural arm and robotic arm movement in the striatum of a marmoset engaged in a two choice task," in *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, 2013, pp. 399-402.

[97] D. W. Moran and A. B. Schwartz, "Motor cortical representation of speed and direction during reaching," *Journal of Neurophysiology,* vol. 82, pp. 2676-2692, 1999.

[98] D. A. Adamos, E. K. Kosmidis, and G. Theophilidis, "Performance evaluation of PCA-based spike sorting algorithms," *Computer Methods and Programs in Biomedicine,* vol. 91, pp. 232-244, 2008.

[99] J. K. Chapin and M. A. Nicolelis, "Principal component analysis of neuronal ensemble activity reveals multidimensional somatosensory representations," *Journal of Neuroscience Methods,* vol. 94, pp. 121-140, 1999.

[100] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: a k-means clustering algorithm," *Applied Statistics,* pp. 100-108, 1979.

[101] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," ed, 2003.

[102] L. Auria and R. A. Moro, "Support vector machines (SVM) as a technique for solvency analysis," 2008.

[103] P. Cunningham and S. J. Delany, "k-Nearest neighbour classifiers," *Mult Classif Syst,* pp. 1-17, 2007.

[104] B. L. Smith, B. M. Williams, and R. K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transportation Research Part C: Emerging Technologies,* vol. 10, pp. 303-321, 2002.

[105] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *ICML*, 2003, pp. 616-623.

[106] J. Cheng and R. Greiner, "Comparing Bayesian network classifiers," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999, pp. 101-108.

[107] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust prediction of fault-proneness by random forests," in *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on*, 2004, pp. 417-428.

[108] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning,* vol. 40, pp. 139-157, 2000.

[109] E. A. Pohlmeyer, B. Mahmoudi, S. J. Geng, N. W. Prins, and J. C. Sanchez, "Using reinforcement learning to provide stable brain-machine interface control despite neural input reorganization," *Plos One,* vol. 9, Jan 30 2014.

[110] E. M. Izhikevich, "Simple model of spiking neurons," *Neural Networks, IEEE Transactions on,* vol. 14, pp. 1569-1572, 2003.

[111] E. M. Izhikevich, "Which model to use for cortical spiking neurons?," *Neural Networks, IEEE Transactions on,* vol. 15, pp. 1063-1070, 2004.

[112] N. Prins, J. Sanchez, and A. Prasad, "A confidence metric for using neurobiological feedback in actor-critic reinforcement learning based brain-machine interfaces," *Frontiers in Neuroscience,* vol. 8, p. 111, 2014.

[113] E. A. Pohlmeyer, B. Mahmoudi, S. Geng, N. W. Prins, and J. C. Sanchez, "Using reinforcement learning to provide stable brain-machine interface control despite neural input reorganization," *PloS One,* vol. 9, p. e87253, 2014.

[114] J. A. Wolf, L. F. Schroeder, D. Contreras, and L. H. Finkel, "Afferent stream integration in a model of the nucleus accumbens," in *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, 2001, pp. 796-801.

[115] M. D. Humphries, N. Lepora, R. Wood, and K. Gurney, "Capturing dopaminergic modulation and bimodal membrane behaviour of striatal medium spiny neurons in accurate, reduced models," *Frontiers in Computational Neuroscience,* vol. 3, 2009.

[116] M. D. Humphries, R. Wood, and K. Gurney, "Dopamine-modulated dynamic cell assemblies generated by the GABAergic striatal microcircuit," *Neural Networks,* vol. 22, pp. 1174-1188, 2009.

[117] Y. Goto and A. A. Grace, "Dopaminergic modulation of limbic and cortical drive of nucleus accumbens in goal-directed behavior," *Nature Neuroscience,* vol. 8, pp. 805-812, 2005.

[118] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artificial Intelligence Review,* vol. 34, pp. 133-143, 2010.

[119] M. A. Nicolelis and M. A. Lebedev, "Principles of neural ensemble physiology underlying the operation of brain–machine interfaces," *Nature Reviews Neuroscience,* vol. 10, pp. 530-540, 2009.

[120] R. Newport, R. Pearce, and C. Preston, "Fake hands in action: embodiment and control of supernumerary limbs," *Experimental Brain Research,* vol. 204, pp. 385-395, 2010.

[121] A. Berti and F. Frassinetti, "When far becomes near: Remapping of space by tool use," *Journal of Cognitive Neuroscience,* vol. 12, pp. 415-420, 2000.

[122] A. Iriki, M. Tanaka, and Y. Iwamura, "Coding of modified body schema during tool use by macaque postcentral neurones," *Neuroreport,* vol. 7, p. 2325, 1996.

[123] C. Acosta-Calderon and H. Hu, "Robot imitation: body schema and body percept," *Applied Bionics and Biomechanics,* vol. 2, pp. 131-148, 2005.

[124] H. Head and G. Holmes, "Sensory disturbances from cerebral lesions," *Brain,* vol. 34, pp. 102-254, 1911.

[125] C. Nabeshima, Y. Kuniyoshi, and M. Lungarella, "Adaptive body schema for robotic tool-use," *Advanced Robotics,* vol. 20, pp. 1105-1126, 2006.

[126] A. Maravita and A. Iriki, "Tools for the body (schema)," *Trends in Cognitive Sciences,* vol. 8, pp. 79-86, 2004.

[127] M. A. Lebedev, J. M. Carmena, J. E. O'Doherty, M. Zacksenhouse, C. S. Henriquez, J. C. Principe*, et al.*, "Cortical ensemble adaptation to represent velocity of an artificial actuator controlled by a brain-machine interface," *The Journal of Neuroscience,* vol. 25, pp. 4681-4693, 2005.

[128] G. E. Alexander and M. D. Crutcher, "Functional architecture of basal ganglia circuits: neural substrates of parallel processing," *Trends Neurosci,* vol. 13, pp. 266-271, 1990.

[129] M. Corbetta and G. L. Shulman, "Control of goal-directed and stimulus-driven attention in the brain," *Nature Reviews Neuroscience,* vol. 3, pp. 201-215, 2002.

Appendix A Adapted from "Prins et al. Representation of Natural Arm and Robotic Arm Movement in the Striatum of a Marmoset Engaged in a Two Choice Task"

One of the findings from the analysis of 0 is that the striatum neural data clusters separated out for robot movement left and right very clearly. We explored this idea further and found preliminary results to be interesting in the subject of body schema and tool use. These findings are interesting for BMIs but is not directly relevant to the work described in this dissertation and therefore we include in the appendix. Some of these results have been adapted from published work in the IEEE Neural Engineering Conference [96].

Introduction to Body Schema and Tool Use

For organisms to access the external environment and make decisions in everyday activities, the intent of the brain is often expressed through a physical medium (the body by nature), which interacts with the environment. Numerous studies have shown that the concept of the brain's embodiment is dynamic [120]. The brain, body, and even tools can become extensions of each other during use through reorganization of neural representation evident in single unit activity during tool use [121-123]. The concept of 'body schema' was highlighted by Head and Homes to describe the mapping of proprioception and motor commands to body posture and movements as well as mapping of tactile sensation from the body surface [124, 125]. These studies showed that the body

schema changes with learning, which indicate that the plasticity of the brain is also involved [126].

The concept of schema has an impact on the field of Brain-Machine Interfaces (BMI). The external device control can be thought of as similar to the idea of body schema in using mechanical tools [127]. Often, when training animals to use BMIs, the animal is initially trained to control the system by moving its own limb. Later, even when the animal has stopped moving its limb or when the limb is paralyzed, there is activity in the motor cortex that can be used for decoding intended arm movement [4] and this can be used to guide a robotic device. Questions arise about how an animal's intent is remapped or projected onto the device that the animal is controlling. Similarly, we are interested in studying how neural representations change while an animal is interacting with a robot arm. It remains unknown in the BMI context how the body schema changes, as better control emerges over time. Since learning may be involved in the modification of behavior and the body schema, we have focused our work on neural circuits involved in goal-directed motor control: the ventral striatum (nucleus accumbens – NAcc) and motor cortex [128, 129].

In this section we explore the possibility of the actions of an external device being represented in the striatum, which has been known to play a key role in reward [51]. We analyzed in this study how the striatal neural signals modulate according to the robot movement direction when the monkey reached in the same direction as robot or opposite direction.

Task and Results

The task was the two-target reach task described in 2.2.2. Recordings of 4 sessions within one week were analyzed (S1, S2, S3, S4) with a total of 124 trials. For each session, the windows analyzed were 0-500ms, 100-600ms and 200-700ms. Figure A.1 (bottom) shows an example of the mean spike count from one neuronal signal from one session with a bin size of 25ms. There is a statistically significant difference (t test. $p<<0.001$) in the firing patterns during the time period (0-700ms). The firing rate when the robot moves towards the left is higher than when the robot moves to the right. For comparison against the natural arm movement Figure A.1 (top) shows the same neuronal unit during the monkey's physical arm movements (window 600-1400ms after go signal). The left arm movement shows higher firing than the right arm movement.

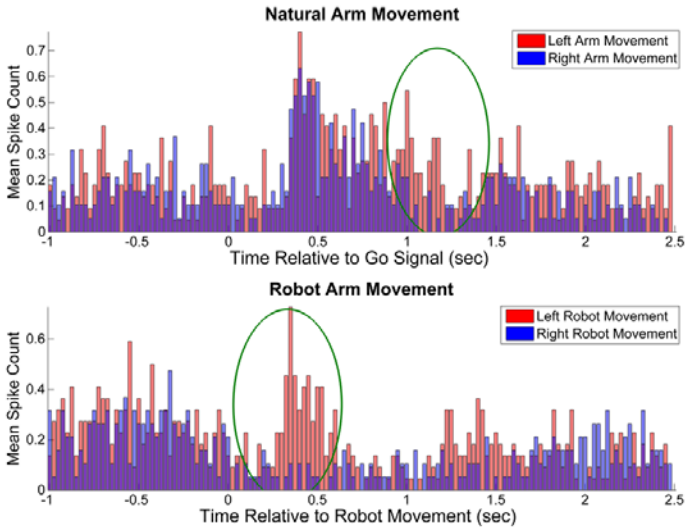

Figure A.1: Mean Spike count for one neuronal signal showing left (red) and right (blue) (Session 1). Bin size 25ms. Top: Natural Arm Movement. Time relative to Go Signal. Bottom: Robot Arm Movement. Time relative to the Robot Movement [96].

Next, the data was converted to PC space and the first two PCs were plotted in 2D. k-means was used to partition the PC space into two clusters as

shown in (green and yellow Voronoi diagrams). Next the trials were labeled manually as left (Figure A.2(A) red data points) and right (Figure A.2(A) blue data points) in the PC space. Finally, the labeled trials were compared against the k-means classes to calculate the resulting clustering accuracy. There is separability of the trials that can be observed in the PC space. For comparison with natural arm movements we plotted the neural data during the arm reach time in Figure A.2(B). The left and right arm movement can be seen in red and blue respectively. The time window in Figure A.2(B) is 800-1300ms after the go signal, which is during the time of the natural reach whereas Figure A.2(B) shows data during the robot movement, i.e. after the natural reach.

The clustering accuracy for the session shown in Figure A.2(A) is 92.7%. The reduced percentage is due to outliers in the PC space and inaccuracies in the PC space being separated. The clustering accuracy for each session and the different time windows are shown in Table A-1 with the highest accuracy of 92.7%. The first column in Table-I gives the clustering accuracy during the hold time as the baseline for comparison. When the plots of the PC space during the hold time are examined further, there is no separation between the clusters as is expected since there is no interaction with the robot during this time period (see Discussion). Clustering accuracy for the first time-window in sessions S3 and S4 was further increased to 88.9% and 77.8% with manual clustering suggesting that more sophisticated methods can improve the accuracy. However, there was no such pattern in the window during the hold time.

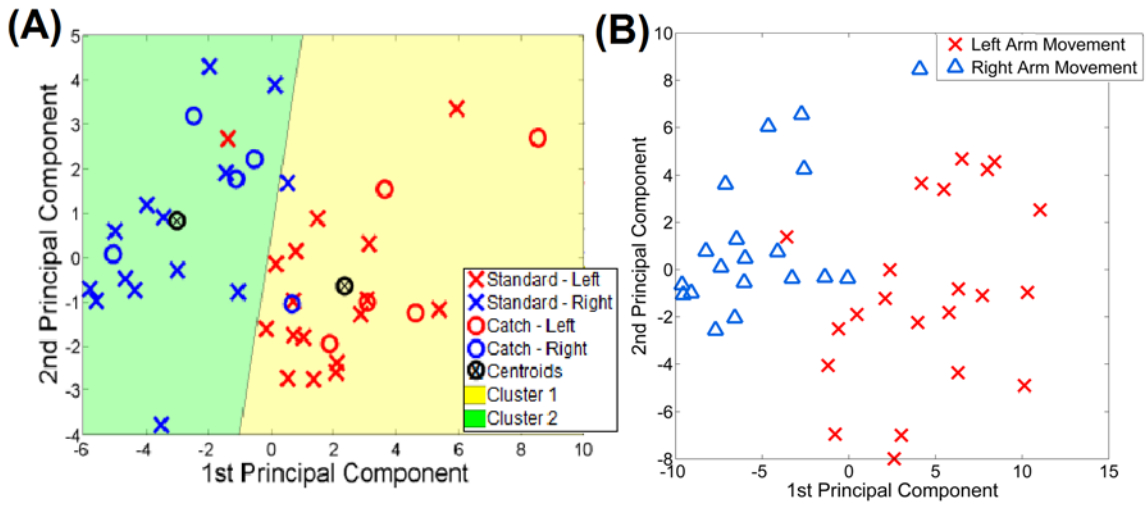

Figure A.2: (A) Data in the first and second principal component space during Robot Movement. Session 2 window 100-600ms. Accuracy 92.7%. Red – left robot movement (A standard trials in x and C catch trials in o). Blue – right robot movement (C standard trials in x and A catch trials in o). Green – cluster 1, Yellow – cluster 2. ⊗ – centroids of each clutser of points. (B) Data in the first and second principal component space during the natural arm movement. Red – left arm movement. Blue – right arm movement. (Similar to the robot movements in figure above, separation between the left and right physical arm movements is evident) [96].

Table A-1: Overall Accuracy Of Clustering Using K-Means For Different Window Sizes. The Accuracy Of Hold Time Is Given As Baseline (Chance Level) [96]

Session	Accuracy of Clustering			
	Hold Time (Baseline)	0- 500ms	100-600ms	200-700ms
S1	61.0%	90.2%	92.7%	85.4%
S2	51.3%	84.6%	89.7%	89.7%
S3	52.6%	60.5%	89.5%	89.5%
S4	60.9%	63.0%	69.6%	89.1%
Average	**56.4%**	**74.6%**	**85.4%**	**88.4%**

Figure A.1 shows a statistically significant difference in the firing patterns for the two directions within the time window analyzed (0-700ms). This was typical of the population of neuronal units recorded. 4, 8, 5 and 6 neuronal units out of 27 neuronal units showed a statistical significant difference (t test: $p<0.05$) for sessions S1, S2, S3 and S4 respectively. Even though the variance accounted for in the first two PCs is only 47%, we are still able to see clear

separability in the data and simple unsupervised clustering techniques show that it is possible to extract robot movement direction from the neural data.

The clustering accuracy during the hold time which was used as the baseline was 56.4% on average. Since the manual labeling of the clusters was always done to maximize the resulting classification accuracy, chance was by default greater than 50%. When the PC space was plotted, there was poor separability between the two clusters. This is expected as during this time there is no robot movement or arm movement present. It is interesting to note how the classification accuracy increases across the time window. The third time window gives the best accuracy results (88.4% on average). Higher accuracy obtained in later time windows could be explained due to the time the animal required to realize and appreciate to which target the robot was moving. The variance in the accuracy of clustering was lowest in the third window (0.04%) where the average performance is the highest. The variance is 2.26% in the first window and 1.13% in the second window. The variance of the results for the baseline (hold time) was 0.27%.

Mathematical Representation of the Different Clusters

We are interested in determining the mathematical representation of the different clusters. This section describes a probabilistic framework for the different clusters. We define the variables in the environment during the robot movement with the following notation.

G	tar**G**et LED
B	ro**B**ot LED
S	target **S**ensor LED
L	target **L**ocation
RM	**R**obot **M**ovement
E	**E**rror
F	**F**lashing

The probability of A standard trials is the probability of the target LED (G) being left, the robot LED (B) being left, the target sensor (S) being left, the robot movement (RM) being left, the error(E) being correct and the lights flashing (F).

$$P(A\_standard) = P(G=Left,\ B=Left,\ S=Left,\ L=Left,\ RM=Left,\ E=Correct,\ F=Flash) \quad \text{Eq. A.1}$$

Similarly, the probability of C standard trials is the probability of the target LED (G) being right, the robot LED (B) being right, the target sensor (S) being right, the robot movement (RM) being right, the error(E) being correct and the lights flashing (F).

$$P(C\_standard) = P(G=Right,\ B=Right,\ S=Right,\ L=Right,\ RM=Right,\ E=Correct,\ F=Flash) \quad \text{Eq. A.2}$$

The C catch trial probability is the probability of the target LED (G) being right, the robot LED (B) being right, the target sensor (S) being right, the robot movement (RM) being left, the error(E) being wrong and the lights not flashing (F).

$$P(C\_Catch) = P(G=Right,\ B=Right,\ S=Right,\ L=Right,\ RM=Left,\ E=Wrong,\ F=Off) \quad \text{Eq. A.3}$$

Similarly, the probability of A catch trials can be represented with the probability of the target LED (G) being left, the robot LED (B) being left, the target sensor (S) being left, the robot movement (RM) being right, the error(E) being wrong and the lights not flashing (F).

P(A_Catch) = P(G=Left, B=Left, S=Left L=Left, RM=Right, E=Wrong, F=Off) Eq. A.4

if we assume that A standard and C catch trials have similar firing properties in the NAcc, we can combine Eq. A.1 and Eq. A.3

P(A_standard) + P(C_Catch)
= P(G=Left, B=Left, S=Left, L=Left, RM=Left, E=Correct, F=Flash)
+ P(G=Right, B=Right, S=Right, L=Right, RM=Left, E=Wrong, F=Off)

= P(G= Left or Right, B= Left or Right, S= Left or Right, L=Left or Right, RM=Left, E=Correct or Wrong, F=Flash or Off) Eq. A.5

The possible choices for G are left and right. Therefore we can marginalize over G;

P(A_standard) + P(C_Catch)
$= \sum_G P(G, B = Left\ or\ Right, S = Left\ or\ Right, L = Left\ or\ Right, RM = Left, E = Correct\ or\ Wrong, F = Flash\ or\ Off)$
= P(B= Left or Right, S= Left or Right, L=Left or Right, RM=Left, E=Correct or Wrong, F=Flash or Off)
Eq. A.6

Similarly, since the possible choices for B, S and L are left and right, we can marginalize over all three variables.

P(A_standard) + P(C_Catch)
=
$\sum_{B,S,L} P(B, S, L, RM = Left, E = Correct\ or\ Wrong, F = Flash\ or\ Off)$
Eq. A.7

$= P(RM = Left, E = Correct\ or\ Wrong, F = Flash\ or\ Off)$

RM has two possible choices, left and right, but in the above equation only RM=left is included and therefore we cannot remove this variable.

The only possible choices for E are correct and wrong and since all the possible states are represented inside the probability above, we can marginalize over E and remove the variable E. Similarly, since all the possibilities of the variable F (flash and off) are included, we are able to marginalize out the variable F as well.

$$\text{P(A\_standard) + P(C\_Catch)} = \sum_{E,F} P(RM = Left, E, F) = \text{P(RM=Left)} \quad \text{Eq. A.8}$$

Similarly, if we assume that the firing properties for C standard and A catch are similar, we can combine Eq. A.2 and Eq. A.4. Variables G, B, S, L, E and F can be marginalized out.

P(C_standard) + P(A_Catch)
= P(G=Right, B=Right, S=Right, L=Right, RM=Right, E=Correct, F=Flash) + P(G=left, B=Left, S=Left, L=Left, RM=Right, E=Wrong, F=Off)

= P(G= Left or Right, B= Left or Right, S= Left or Right, L=Left or Right, RM=Right, E=Correct or Wrong, F=Flash or Off)

$= \sum_{G,B,S,L,E,F} P(G, B, S, L, RM = Right, E, F)$
= P(RM=Right) Eq. A.9

In conclusion if A standard and C catch are considered together, the results we see are a strong indication of robot movement towards left and

similarly when we consider C standard and A catch together the results are an indication of the robot moving to the right.

P(A_standard) + P(C_Catch) = P(RM=Left) Eq. A.10
P(C_standard) + P(A_Catch) = P(RM=Right) Eq. A.11

Next we consider the results where we see A standard and C standard have similar neuronal activity. Therefore we sum Eq. A.1 and Eq. A.2

P(A_standard) + P(C_standard)
= P(G=Left, B=Left, S=Left, L=Left, RM=Left, E=Correct, F=Flash)
+P(G=Right,B=Right, S=Right, L=Right, RM=Right, E=Correct, F=Flash)

= P(G= Left or Right, B= Left or Right, S= Left or Right, L=Left or Right, RM=Left or Right, E=Correct, F=Flash)

= $\sum_{G,B,S,L,RM} P(G,B,S,L,RM,E = Correct,F = Flash)$
= P(E=Correct, F=Flash) Eq. A.12

Similarly, when considering A catch and C catch trials together, we can sum Eq. A.3 and Eq. A.4

P(A_catch) + P(C_catch)
= P(G=Left, B=Left, S=Left, L=Left, RM=Right, E=Wrong, F=Off)
+ P(G=Right, B=Right, S=Right, L=Right, RM=Left, E=Wrong, F=off)

= P(G= Left or Right, B= Left or Right, S= Left or Right, L=Left or Right, RM=Left or Right, E=Wrong, F=Off)
= $\sum_{G,B,S,L,RM} P(G,B,S,L,RM,E = Wrong,F = Off)$
= P(E=Wrong, F=Off) Eq. A.13

In conclusion, when we consider the A and C standard trials together, the results indicate both the robot performing a correct action as well as lights flashing. When we consider A and C catch trials together, the results indicate the robot performing a wrong action and the lights bring off.

P(A_standard) + P(C_standard) = P(E=Correct, F=Flash) Eq. A.14
P(A_catch) + P(C_catch) = P(E=Wrong, F=Off) Eq. A.15

Final Remarks

Here we examined the neural representation of the movements of an external robotic arm in the striatum and compared its neuromodulation during natural and robotic arm reaches. We were able to visualize the data in two-dimensional space and cluster the data according to left and right robot movements. We observed the difference in neuronal firing in the striatum during both natural arm movement and robot arm movement. We also showed mathematical proof that the clustering given by k-means corresponded to left and right robot arm movement.

Summary

Preliminary results shown in this study provides evidence of representation of an external device in a subcortical structure. The results from the body schema concept if incorporated into the BMI design may be used in the development of future intelligent controllers which can incorporate interactions between motor and reward structures for better controller design. However, the target space also needs to be expanded in order to test the reliability of extraction of direction-specific information.