Comparison of Feature Selection Methods for Robust Dexterous Decoding of Finger Movements from the Primary Motor Cortex of a Non-human Primate Using Support Vector Machine

by

Subash Padmanaban

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2015 by the
Graduate Supervisory Committee:

Bradley Greger, Co-Chair
Marco Santello, Co-Chair
Stephen Helms Tillery

ARIZONA STATE UNIVERSITY

August 2015

ABSTRACT

Robust and stable decoding of neural signals is imperative for implementing a useful neuroprosthesis capable of carrying out dexterous tasks. A nonhuman primate (NHP) was trained to perform combined flexions of the thumb, index and middle fingers in addition to individual flexions and extensions of the same digits. An array of microelectrodes was implanted in the hand area of the motor cortex of the NHP and used to record action potentials during finger movements. A Support Vector Machine (SVM) was used to classify which finger movement the NHP was making based upon action potential firing rates. The effect of four feature selection techniques, Wilcoxon signed-rank test, Relative Importance, Principal Component Analysis, and Mutual Information Maximization was compared based on SVM classification performance. SVM classification was used to examine the functional parameters of (i) efficacy (ii) endurance to simulated failure and (iii) longevity of classification. The effect of using isolated-neuron and multi-unit firing rates was compared as the feature vector supplied to the SVM. The best classification performance was on post-implantation day 36, when using multi-unit firing rates the worst classification accuracy resulted from features selected with Wilcoxon signed-rank test (51.12 ± 0.65%) and the best classification accuracy resulted from Mutual Information Maximization (93.74 ± 0.32%). On this day when using single-unit firing rates, the classification accuracy from the Wilcoxon signed-rank test was 88.85 ± 0.61 % and Mutual Information Maximization was 95.60 ± 0.52% (degrees of freedom =10, level of chance =10%)

ACKNOWLEDGMENTS

I would like to thank my parents, Padma and Padmanaban, for being supportive and encouraging me throughout my Master’s thesis. I am truly indebted to you both for giving me the freedom to pursue my dreams as early on as I can remember.

I consider myself fortunate to have carried out my Master’s research in the Neural Engineering Laboratory under Dr. Bradley Greger. Thank you Dr. Greger for showing us how to be a good researcher. In addition to promoting crazy ideas, I’m thankful to you for giving me a sense of bigger picture and showing me the right direction during critical moments of my thesis. I would also like to thank Dr. Marco Santello and Dr. Stephen Helms Tillery for their constant feedback which shaped up this thesis.

I would like to thank my brother, friends and family for their support.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

Microelectrode array brain machine interfaces (BMI) have shown the potential to alleviate various neurological disorders. BMIs utilizing advances in robotics and machine learning can restore limited lower and upper extremity motor function. Several research studies have investigated the viability of a cortical brain machine interface in humans and NHPs [1-3].

BMIs can be broadly classified based on the type of bio-signal used to control the prosthesis. Electroencephalogram (EEG), Local field potential (LFP) and Action potential (AP) constitute the majority of source signals used in brain machine interfaces. APs are discrete spiking events of an individual neuron. In statistics terms, APs or neural “spiking” can be thought of as a non-stationary point process in which neural information is largely encoded by changes in the AP firing rate coding (frequency of action potentials/spiking). In this paper, we utilize neural recordings of APs from individual neurons to classify various movements of the fingers.

Brain machine interfaces for controlling a robotic limb or moving a cursor have been successfully demonstrated in humans and non-human primates [10-12]. These systems provided real time control of a neuroprosthetic system by decoding neural signals moment by moment with an objective to provide certain functionality to replace the native arm. Communication prostheses focus on achieving discrete goals like moving cursor to specific targets [13-15]. These systems are based on decoding the endpoint goal of reach and map the neural signals to spatially distributed targets.

Motivating Problem

One of the important characteristics of the human upper extremity functioning is the ability to perform coordinated and dexterous finger movements. Typing, eating with a spoon, writing with a pen and opening a lock with a key are some of the examples in our daily life that require such dexterous manipulations. Incorporating dexterity as a feature in a neuroprosthesis would help amputees and paralyzed persons to carry out a wider range of tasks. To achieve such dexterous control requires a neural decoding algorithm that can map high-dimensional neural signals onto a high-dimensional hand prosthesis. Optimizing algorithms for decoding neural signals will be critical for providing useful control of upper extremity neuroprostheses. Feature selection is an important step in designing a machine learning system. Choosing a *w*-dimensional subset from a *p*-dimensional feature space consisting of '*p'* predictors using an objective metric is the aim of feature selection. Feature selection also reduces the dimensionality of feature space, inundating it with more "informative" features thus, removing lesser contributing ones that might occlude the feature space.

Potential Solution

Brain machine interfaces pose significant surgical risks and other health hazards which place them in the lower end of the therapeutic spectrum. Even in cases such as amputation and neurological diseases such as ALS where brain machine interfaces prove to be the only solution for recovering limited motor functions, the risk to benefit ratio of the current constructs make it unsuitable for pragmatic purposes. In order to make it a viable, long-term solution, the performance of the brain machine interface must be valuable to the user in terms of efficacy and durability.

Neural decoding is the process of converting raw neural signals acquired from the user to generate useful actuation signals for the neuroprosthesis to help accomplish a task. Neural decodes play a critical role in realizing the high levels of performance in a brain machine interface. Kalman filter based algorithms have proven to be efficient in decoding continuous parameters such as position and velocity [1, 3, 11, 16-17]. Kalman filter also known as liner quadratic estimation, is a set of equations describing the relationship of the system and its output by assuming a Gaussian noise error in each equation. The simple Kalman filter which has been used in the aforementioned papers, assumes a linear relationship between the input (neural data, in this case) and the output (movement trajectory). For decoding discrete targets, a variety of machine learning algorithms have been employed in motor neuroprosthetic application. Milekovic et al examined the applicability of regularized linear discriminant analysis for decoding bi-directional cursor movements on screen [18]. Linear discriminant analysis is a method of searching the optimal linear combination of features that best help separate the ‘k’ classes. It is closely related to principal component analysis and logistic regression in creating a linear decision boundary. Kim et al analyzed various linear and nonlinear filters such as Wiener filter, LMS adaptive filter, Gamma filter and subspace Wiener filter for a food-reaching and target hitting task from the motor cortex of a non-human primate [19]. These filters are loosely related to the simple Kalman filter and proved to work significantly better than the Kalman filter in both the food-reaching and target hitting tasks. The authors attributed the decreased performance of the simple Kalman filter to the inefficiency in estimating the Kalman Gain matrix due to errors in estimating the large covariance matrices.

An integral part of designing machine learning systems for neuroprosthetic applications is feature selection. Removing redundant information and inundating the feature space with relatively more “informative” features is the objective of feature selection. We investigate the performance of four feature selection algorithms namely, Wilcoxon signed-rank test, Relative Importance, Principal Component Analysis and Mutual Information Maximization in classifying dexterous finger movements from neural signals. The performance of these feature selection techniques will be assessed based on (i) efficacy (ii) endurance to simulated failure and (iii) longevity of neural decode. We also analyze the impact of using AP firing rates from individual neurons (single-unit recordings) and from multiple neurons (multi-unit recordings) as the input feature vector to the multiclass SVM. We believe that the metrics chosen here for comparing the performance of neural decodes encapsulate the crux of the issues that need to be addressed while designing a machine learning system for neuroprosthetic applications.

CHAPTER 2

METHODS

The recording setup, behavioral task, data collection and preliminary data processing approaches are explained elsewhere [4]. A 96 channel microelectrode array (MEA, Blackrock Microsystems) was implanted in the hand area of primary motor cortex of a male macaca mulatta. The non-human primate (NHP) was trained to perform cued combined flexions of the thumb, index and middle finger and individual flexions and extensions of the same digits using a manipulandum. Visual cues were provided using a computer screen placed in front of the monkey. In order to start a trial, the monkey had to relax all its fingers moving all of the finger switches in the manipulandum to the open state. After a randomized wait time of 1000-3000ms, a visual cue indicating which finger(s) to flex/extend appeared on the computer screen. The monkey then had 2000ms to react to the visual cue and depress the associated switch. Once the correct switch was pressed, the monkey had to hold the switch for 500ms. The trial was deemed successful if the monkey pressed the correct switch and adhered to the time constraints. The behavioral task was implemented using a real-time operations systems in a custom LabVIEW (National Instruments) program.

Neural Decoding System Architecture

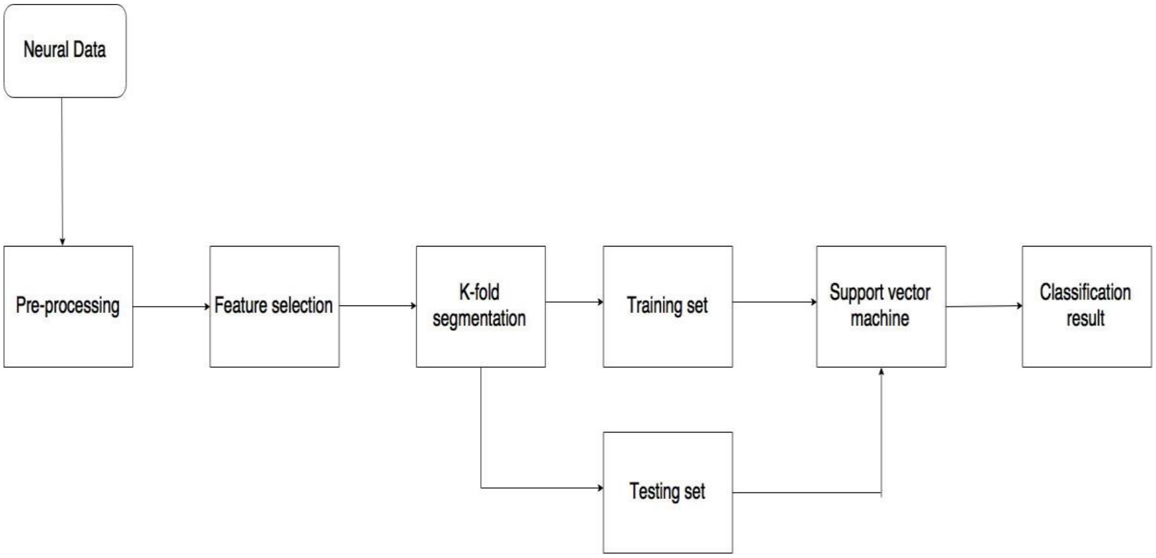

Figure 1. Neural decoding system architecture

Figure 1 shows the architecture for the machine learning decoding system. Neural data recorded from the NHP was spike sorted using an offline sorter (Plexon, Inc.). The timestamp of spike events was obtained from the offline sorter. Pre-processing also included binning/moving average windowing of the point process using a boxcar window. After applying the moving average technique, neural "firing rate" for each single or multi-unit was obtained. Neural firing rate was used as the feature vector (input) to the SVM. Trial snippets corresponding to each successful finger movement trial was extracted and concatenated. The entire dataset was randomly divided into 10 folds. Each fold served as the testing set once while data from the remaining folds was used for training. Model parameters such as box constraint(C) and sigma (of the RBF kernel) were estimated using an exhaustive grid search algorithm with exponentially increasing values from 1e-5 to 1e5. Classification accuracy was calculated after predictions were made on the unseen test set. This process was repeated 20 times to reduce generalization error of the SVM.

Pre-processing

The MEA is a 10x10 grid of 1 mm tall electrodes that are capable of recording single and multi-unit activity in addition to local field potentials [5]. The MEA data were sampled at 30 kHz. Neural data collected using the MEA were sorted offline using an expectation-maximization based competitive mixture of *t*-distributions decomposition algorithm [9]. Data were then imported to Matlab (Mathworks) for further analysis. The time stamps of action potentials recorded at 30 kHz were downsampled to 600 Hz. A boxcar moving average window of 300ms width and 33.3 ms step size was used to obtain a moving average “firing rate”. The moving average of the point process was downsampled in order to reduce data size. A 4th order low pass Butterworth filter with a cut-off frequency of 10 Hz was used prior to downsampling the neural firing rate to 20 Hz and the neural firing rate was obtained as a time varying vector. This process was repeated for all 96 electrodes to obtain multi-unit neural firing rate, i.e. the cumulative firing rate of all neurons recorded on a particular electrode. An average of 142.2 ± 36.3 neural units were recording from 96 electrodes during each session.

Data from individual trials was aligned in time on switch closure times of successful trials. A movement period was defined as the duration corresponding to 450ms prior and 1000ms after the switch closure. A baseline period (resting state) for a trial was defined as the duration corresponding to 2500ms to 1000ms prior to switch closure. Baseline and movement period data was obtained for all available degrees of freedom and all successful trials for each day experiments were conducted and represented a vector of time-series data.

Feature Selection

Using machine learning algorithms for multivariate, high-dimensional data is often computationally expensive. Due to the complexity of feature space and rigorous numerical computations involved in designing the hyperplane in this high-dimensional feature space, the performance of the machine learning algorithm is deterred. Feature selection is the process of selecting an O-dimensional subset feature space from a P – dimensional original feature space where ‘p’ is the number of predictors. In case of the neural data, there were 96 predictors for multi-unit based firing rate feature vector and an average of ~144 predictors for single-unit based firing rate feature vector.

Feature selection is usually applied to reduce information redundancy and trim the input space to better predict the responses. Some of the advantages of feature selection are:

- Facilitate data visualization and data understanding
- Reduce data measurement and storage requirements
- Reduce training and utilization times
- Simplify the learning model and aid in better understanding and interpretation by researchers
- Enhance generalization by reducing overfitting
- Defy the curse of dimensionality to improve predictor performance [23].

Identifying the best subset of features is a sub-optimal problem to solve. The only method to do this is through exhaustive grid search, i.e. exhaustively searching through every permutation of predictors available. Mathematically, there exists 2^p permutations of features that can be selected from ‘p’ features. In case of our neural data, this results in iterating through a minimum of 2^{96} (96 features for multi-unit firing rate and >96 features for single-unit firing rate based feature vector) permutations of features to identify the “best” subset.

When dealing with multivariate, time-series signals like neural signals, it is imperative to judge where the learning algorithm must focus its attention. *Filter* or *Criterion* based feature selection and *Wrapper* based feature selection are two broad categories of feature selection that are commonly applied in machine learning. Application of statistical, empirical or other “criteria” based methods such as mean, variance, student’s t-test and correlation are some examples of criterion based feature selection. Applying criterion based feature selection requires some domain expertise in order to determine what qualifies as a useful criteria. Wrapper based feature selection iteratively uses various combinations of features as input to a machine learning algorithm and evaluates the importance of each feature based on some evaluation criteria from the prediction such as coefficient of determination (r^2). Ideally, it is advisable to use the same machine learning algorithm as a classifier and a wrapper for feature selection. Oftentimes, it is also valuable to use a simpler, computationally efficient machine learning algorithm as a substitute wrapper. For example, SVMs are an efficient yet computationally intensive solution to solve the problem of face recognition by computing key points (that act as features) on the face. Using SVM as a wrapper in this case would demand access to a lot of resources (in terms of clusters) and still be time consuming. An alternative to using SVM in this case would be using a simpler algorithm such as Logistic regression. Care should be taken to ensure both the algorithms have similar assumptions about the data (such as nonlinearity, heteroscedasticity of noise). In this study, we have limited our comparisons to criteria based feature selection methods.

Redundancy, Correlation and Complementarity of Features

Guyon and Elisseeff examined the properties of multivariate features and their impact on feature subset selection. The motivation to use feature subset selection is to reduce the redundancy of information in the feature space. Oftentimes, redundant features are irrelevant and thus, do not contribute to increasing the classification accuracy. It should be noted that, there are certain cases where inclusion of few redundant variables can be beneficial in noise reduction. Consider two independently and identically distributed (i.i.d) variables that follow a Gaussian distribution with zero covariance as a feature vector to solve a two class problem. By averaging the two i.i.d. variables and using it as a new feature improves class separability by a factor of √2. In general, it can be mathematically proven that by averaging ‘n’ i.i.d. variables, we will get a reduction in standard deviation of √n. Noise reduction and subsequently better class separation can be obtained by adding presumably redundant variables [23-25].

Feature correlation impacts the amount of redundancy in the feature space. Let Ɛ represent the correlation of two i.i.d variables. It was found that, there was maximum improvement in class separability when the two distributions were perpendicular to each other (Ɛ is small, but not zero). In case of perfectly correlated variables (Ɛ = 0), the sum of the two variables results in an increase in intra-class covariance by a factor of α and does not necessarily improve class separability. Perfectly correlated variables are truly redundant in the sense that there is no additional information gain obtained by adding them.

One of the concerns of multivariate features is their property of overfitting. Let us consider the famous XOR problem (also known as the two-bit parity problem). The distribution for a two feature, two class problem is given below. The truth table of an XOR function is as follows:

Table 1. XOR truth table

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

X_1 and X_2 are two input features taking binary values. Y is the output of this problem that can also take binary values. X_1 and X_2 are useless by themselves as plotting X_1 vs. Y and X_2 vs. Y reveals that the two univariate problems are non-separable. But by combining the two features X_1 and X_2, we can get separability in two dimensions through a nonlinear decision boundary (using a sigmoid function). This is a classic example that illustrates the property of feature complementarity in machine learning. Two features that are useless by themselves, can be useful together.

Wilcoxon Signed-rank Test

Wilcoxon signed-rank test is a non-parametric alternative to the student's t-test. This non-parametric test can be used to identify if samples from two independent yet related distributions are significantly different. In the context of selecting single or multi-unit data as input to the SVM, the difference between baseline and movement related firing rate was computed. The null hypothesis was that the data came from a continuous, symmetric distribution with a median equal to zero (i.e. no electrode recorded increased firing rates in the movement period as compared to the baseline period). Electrodes for which the null hypothesis was rejected ($p<0.001$) with a positive median difference from baseline were kept. These electrodes were then sorted in order of increasing median difference. For the purpose of feature selection, the median difference was computed as a scalar to select features (single unit/multi-unit).

Relative Importance

Relative importance was a feature selection technique initially developed for selecting neurons in the primary motor cortex for decoding [6]. First the movement only firing rate (difference of movement and baseline firing rate) was computed. The trial averaged firing rate for each neuron for all the successful trials was calculated. Then, the inter-movement variance was computed as the difference of trial averaged firing rate and the average firing of a neuron for a degree of freedom. The neural recordings were then ranked in descending order of inter movement variance. For the purpose of feature selection, the inter movement variance was computed as a scalar to rank features (single/multi-unit).

Principal Component Analysis

Principal component analysis (PCA) can be used as a feature transformation technique, where a transform function is applied to the data to represent it in a higher dimensional transform space. For an '*n'* dimensional possibly correlated data, PCA represents the data in a (n-1) dimensional space in linearly uncorrelated principal component coordinates. The transformation is carried out in such a way that the first principal component contains the maximum possible variance of the data. The succeeding principal components are ordered in descending order of variance. This transformation of data according to the variance at each time point can be used to eliminate noise, but does not necessarily extract discriminative features. Neural firing rates corresponding to each degree of freedom was provided as an input to PCA.

The operation of PCA can be thought of as revealing the internal structure of the data based on its variance. For a multivariate dataset that can be represented in a high-dimensional space, PCA provides a better representation in low-dimensional space from an "informative" viewpoint. This is done by considering only the first few principal components and thus, PCA serves as a dimensionality reduction method.

Mutual Information Maximization

Mutual information is the mutual dependence of two random variables. Unlike correlation, mutual information is not limited to real-valued random variables and estimates how similar the joint distribution P(X|Y) is to the products of the factored marginal distribution P(X) and P(Y). Entropy of a random variable C can be defined as

$$H(C) = -\sum_{c} P(c)\log(P(c))$$

The conditional entropy of two random variables C and Y can be defined as

$$H(C|Y) = -()(\sum_{c} P(c|y)\log\big(P(c|y)\big)\,dy$$

Then, the mutual information of random variables C and Y can be defined as the I(C;Y) = H(C) – H(C|Y) and can be represented as

$$I(C|Y) = \sum_{c}\sum_{y} P(c|y)\log\frac{P(c|y)}{P(c).P(y)}$$

Mutual Information maximization was implemented using the FEAST Toolbox available for MATLAB [7]. For a class label X, the mutual information score of feature C_k is defined as:

$$J(C_k) = I(C_k;X)$$

This score $J(C_k)$ is referred to as mutual information maximization and we rank the features in descending order of the mutual information score. Neural firing rates corresponding to movement period for each degree of freedom was used as the input to Mutual Information Maximization algorithm.

Support Vector Machine

Support vector machine is a class of non-probabilistic, binary, linear classifier. Support vector machines represent the data in higher dimensional space and find the best separating hyperplane in this space. The objective of the SVM is to find a hyperplane that has the maximum distance from a point belonging to any class. Such a classifier is also called a maximum margin classifier whose generalization error is low. During training, each point in the training set is assigned a weight α. Those points with training weights $\alpha \neq 0$ are called the support vectors since, they help forming the hyperplane. In case of linearly non-separable cases, a soft margin classifier is implemented which allows for misclassified instances. Non-linear problems can be solved by using the “kernel trick” in the SVM. Kernel functions map data into a higher dimensional space where, the hyperplane is now formed. Gaussian (radial basis function) kernel was employed in our classification problem to account for non-linearity in the input-output relationship.

Gaussian kernel K(x,x') for two samples x and x' defined as a feature vector in some predictor space is defined by,

$$K(x, x') = \ exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

where σ is a free parameter that defines the smoothness of Gaussian kernel.

SVMs are inherently binary classifiers i.e. they can distinguish between only two classes. Their functionality can be expanded to solve multiclass problems by decomposing it into multiple binary sub-problems. We used a one-vs-one multiclass implementation of the SVM to differentiate between the many available degrees of freedom. For a problem of classifying *'k'* classes, we require $\frac{k(k-1)}{2}$ binary SVM classifiers for each pair of the 'k' classes. The class of a test instance is predicted by taking the mode of predictions of all the one-vs-one SVM pairs. In a one-vs-all implementation, there are 'k' pairs of SVM to classify *'k'* classes where each SVM differentiates between class k_i and the rest.

In addition to extracting snippets of neural activity corresponding to valid trials for all available degrees of freedom for a particular session, we included 30 random baseline periods as a "rest" phase (11th degree of freedom). The target/output instances used for training were created manually depending on the degree of freedom which was later used for estimating the classification accuracy. Target instances were assigned to each sample based on the trial type or degree of freedom it came from and were assigned discrete values like 1,2,3 and so on.

Performance Metrics

Efficacy of Neural Decode

The first step in assessing the performance of feature selection methods was to find the optimal number of features for each feature selection algorithm that best classified the different finger movements and the resting state. For this purpose, all available successful trials in a session were split into a 70% for training (and validation) and the remaining 30% for testing. A 10 fold cross validation routine was performed to reduce variability in performance estimates during validation. For a given input data (multi-unit or single-unit firing rate), the features were ranked based on the results of the feature selection algorithms. We iteratively incremented one feature at a time and used it as an input to the classifier to identify the optimal number of features. We also included random multi-unit and isolated unit firing rate selection to compare with the other methods.

Endurance to Simulated Failure

The performance of the brain machine interface (BMI) can be influenced by the quantity of neural information available for decode. Previous research has shown that there is a significant decrease in the signal to noise ratio of the neural signals and a steady decrease in impedance of the recording electrodes over time [8]. There can be a steady decrease in the number of electrodes that record action potentials, which can have a deleterious effect on BMI performance. Feature selection algorithms should be robust enough to handle the sudden losses in neural information over time. In order to test the endurance of the feature selection algorithms, we randomly dropped 10's of percent of the available features and tested its performance. The random removal procedure was repeated 20 times to reduce generalization bias.

Longevity of Neural Decodes

Brain machine interfaces are devices which will be used over an extended period of time. In order to be useful the neuroprosthetic device must be capable of accurate performance over this extended period of time. We present here the chronic decoding results of 32 sessions collected over 79 days. For a given session the optimal number of features was computed. Decoding accuracy for a feature selection algorithm on a particular day was then calculated using the cross validated optimal features.

Cross Validation

In machine learning, the performance of an algorithm is evaluated by splitting the entire dataset into a mutually exclusive training and testing sets. Supervised learning methods such as Support Vector Machine are induction algorithms that predict responses for unseen test data based on a model learned from the training data. Quantifying the unbiased performance of a classifier is imperative for model selection and predicting real-time values. For a set of training pair (x_i , y_i) where x_i refers to training data (input) and y_i ($y \in \{0,1\}$) its corresponding class label (output), the SVM computes a hyperplane or decision boundary which separates the binary classes 0 and 1 by minimizing a cost function $J(\Theta)$ [26]. The cost function of the SVM was inspired from the logistic regression and is as follows:

$$\frac{min}{\theta}\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\,cost_1\left(\theta^T x^{(i)}\right)+\left(1-y^{(i)}\right)cost_0\left(\theta^T x^{(i)}\right)]+\frac{\lambda}{2m}\sum_{i=1}^{m}\theta_j^2$$

Where $cost_1 = -(\log h_\theta(x^{(i)})$ (cost function when y=1 is misclassified)

$cost_0 = -\log(1-\ h_\theta x^{(i)})$ (cost function when y=0 is misclassified)

$h_\theta(x) = \frac{1}{1+\,e^{-\theta^T x}}$ (hypothesis relating input 'x' and 'y')

λ – regularization parameter, where $\lambda \in [10^{-5}, 10^{5}]$.

m – number of training instances

The cost function penalizes the learning algorithm by some quantity Ɛ, when it incorrectly learns a training instance. Thus, this correction leads to learning the correct mapping for the given dataset.

In order to select the optimal parameters for the SVM model, a cross validation routine was performed. Estimating this performance also helps predict the classification accuracy of future data. For assessing the final accuracy of the classifier, we would like a method that has low bias and low variance. Bias and variance are two types of error that are commonly encountered in supervised learning algorithms. Cross validation can be broadly divided into two categories:

1. Exhaustive cross validation
2. Non-exhaustive cross validation

Exhaustive cross validation measures learn and test the performance of a classifier in all possible ways. There are two popular methods of performing exhaustive cross-validation:

(i) Leave – p –out cross validation:

Let ‘n’ be the total number of instances in the dataset. In this method, ‘p’ instances are used as the validation set when ‘n-p’ instances are used for training. The process is repeated until all possible combinations of validation is exhausted. The classifier learns and validates the dataset a total of C_p^n times.

(ii) Leave-one-out cross validation:

This can be considered as a special case of leave-p-out cross validation where p=1. Each instance in the dataset serves as a validation set once while the remaining dataset is used for training. A total of $C_1^n = n$ iterations of unique training and validation is performed.

Non-exhaustive validation does not evaluate the performance of the classifier exhaustively. The popular methods of non-exhaustive cross validation are:

(i) K-fold cross validation:

In this method, the dataset is divided into ‘k’ folds. In a 10-fold cross validation scheme, data from 9 folds are used for training the model and the remaining one fold serves as a testing set exactly once. The final estimate is obtained by averaging the performance across all folds. When k=n, k-fold cross validation becomes identical to leave-one-out cross validation. Repeating the k-fold procedure multiple times produces a Monte Carlo type estimate which is a better generalization of the unbiased estimate of a classifier. K-fold cross validation suffers from high bias as the random splitting of datasets into may have unequal data points from each class.

(ii) Stratified k-fold cross validation:

Suppose our data has 1000 instances of class A and 10 instances of class B. Chances are that the data from Class B may not be equally represented in a 10-fold cross validation procedure. It may also be completely absent in certain folds. Stratified k-fold cross validation takes this into account and populates each fold with data from various classes of almost equal proportions. Similar to k-fold cross validation, the final classification accuracy is obtained by averaging the performance across all folds

(iii) Holdout cross validation:

This is the simplest cross validation procedure. Data is divided into two mutually exclusive sets of training and testing sets. This can be considered as a 2-fold cross validation. Since the training and testing sets are large, we can be sure that each data point would have served as a training and testing set at least once [27].

For our analysis, we used a repeated 10-fold cross validation procedure. The entire data was divided into a 70-30 mutually exclusive training and testing set. The training set was 10-fold cross validated for model selection and the parameters with the least error rate across all folds was used on the test set. K-fold segmentation was performed 20 times and the performance was averaged to get the final classification accuracy.

CHAPTER 3

RESULTS

Efficacy of neural decode

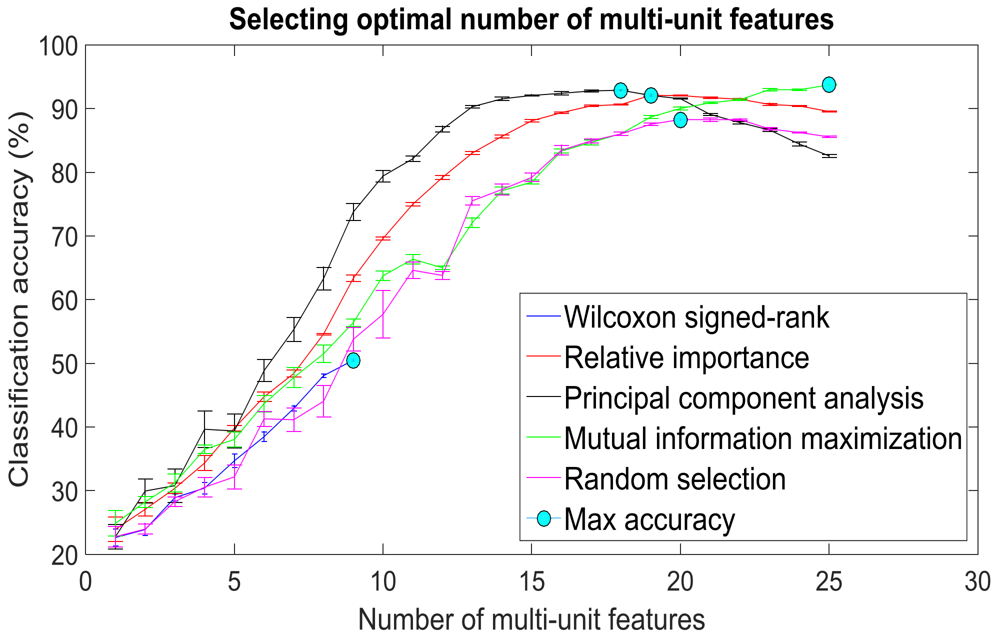

Figure 2(a) Selecting optimal number of channels. The plots above shows the cross validated accuracy of feature selection algorithms for increasing number of multi-unit features. The solid circle (cyan) in each graph shows the maximum accuracy for a feature selection algorithm.

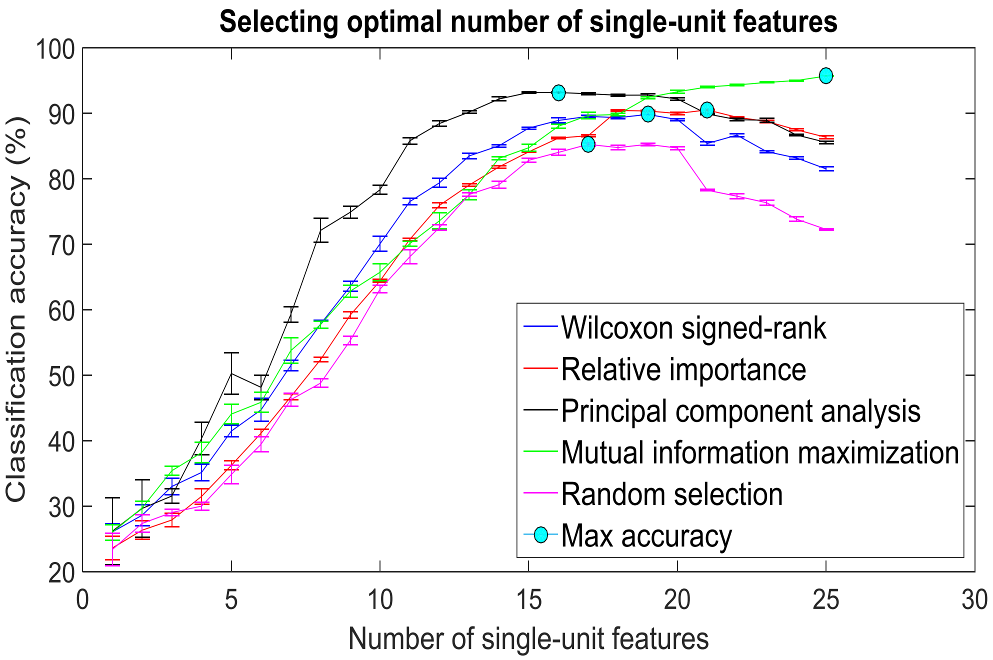

Figure 2 (b) Selecting optimal number of units. The plots above shows the cross validated accuracy of feature selection algorithms for increasing number of single-unit features (Fig. 2 b). The solid circle (cyan) in each graph shows the maximum accuracy for a feature selection algorithm. The number of channels/units corresponding to this accuracy was chosen as the optimal number of features.

From figure 2 (a-b), it can be seen that different feature selection methods have different optimal number of features. With an exception of Wilcoxon signed-rank test, the other feature selection algorithms did not show significant changes from using multi-unit and single-unit firing rate both in terms of number of optimal features and classification accuracy (less than ± 3% difference in classification accuracy and ± 1 feature). In case of Wilcoxon signed-rank test, the number of optimal features increased from 9 features for multi-unit firing rate to 19 feature for single-unit firing rate. The classification accuracy improved from 51.12 ± 0.65 % for multi-unit firing rate to 88.12 ± 0.61 % for single-unit firing rate (Figure 2).

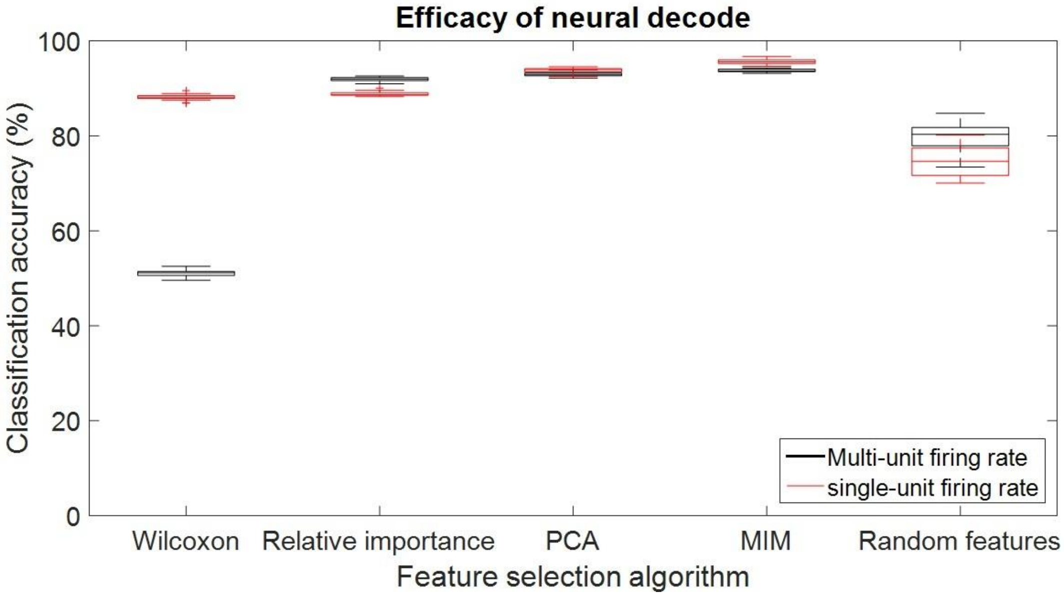

Figure 3. Efficacy of neural decode on post implantation day 35. Classification accuracy of feature selection algorithms on the test set using cross validated optimal number of features. The plots in red and black correspond to classification accuracy obtained using multi-unit firing rate and single-unit firing rate respectively. Level of chance was 10% (10 degrees of freedom).

Endurance to Simulated Failure

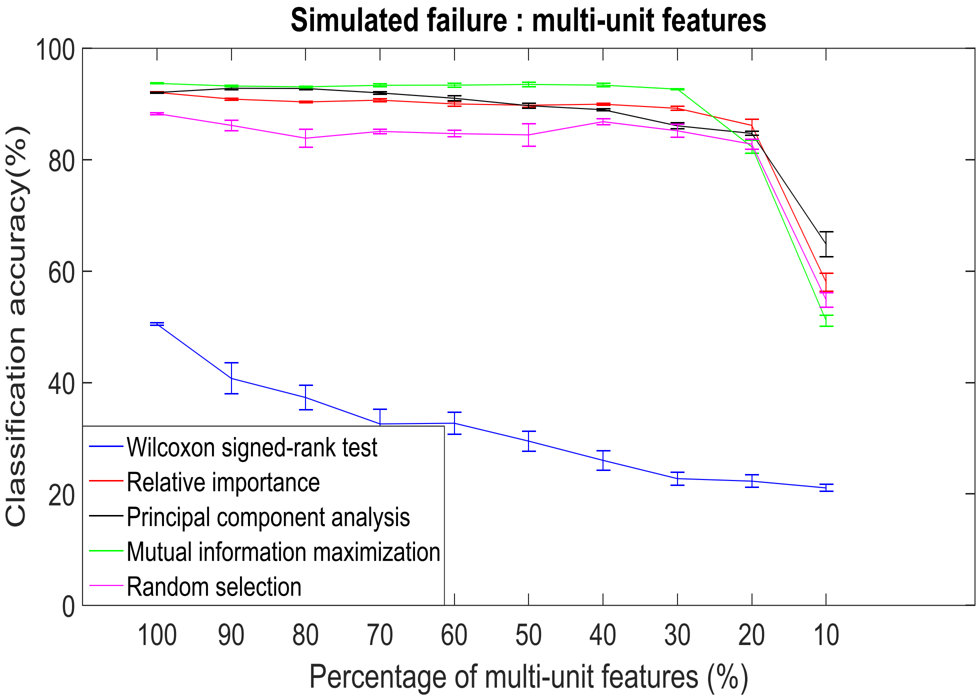

Figure 4a Endurance to simulated failure of multi-unit firing rate.

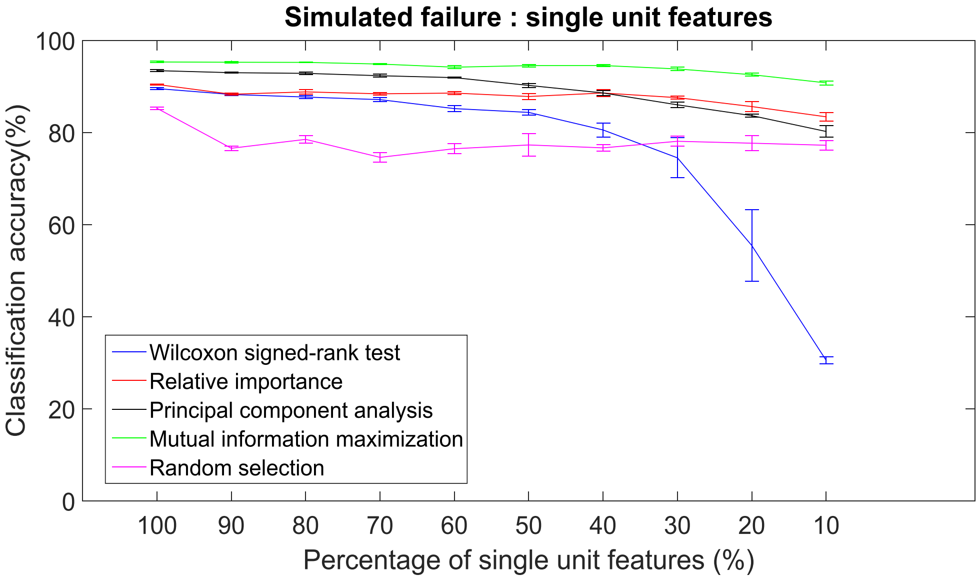

Figure 4b Endurance to simulated failure of singl-unit. Mutual information maximization

based feature selection had a classification accuracy of 90.79% with just 10% of the neural units as feature vector.

There is a general trend of decrease in the performance of feature selection algorithms when we decrease the number of features from 100% to 10%. While using multi-unit firing rate, the performance of Principal component analysis was best at 64.82 ± 2.27 % for 10% of channels, whereas the performance of Wilcoxon signed-rank test was 21.08 ± 0.63 %. When we used single-unit firing rate as the feature vector, the endurance to simulated failure was higher for all feature selection algorithms when compared to their respective multi-unit firing rate. In case of Wilcoxon signed-rank test there was a ~10% increase in classification accuracy while there was a ~40% increase in classification accuracy for Mutual information maximization based feature selection. The performance of mutual information maximization feature selection for single-unit firing rate stayed above 90% classification accuracy even while using only 10% of the available units. There is a clear advantage to using single-unit firing rate at times when the quantity of neural features (single/multi-units) decreases.

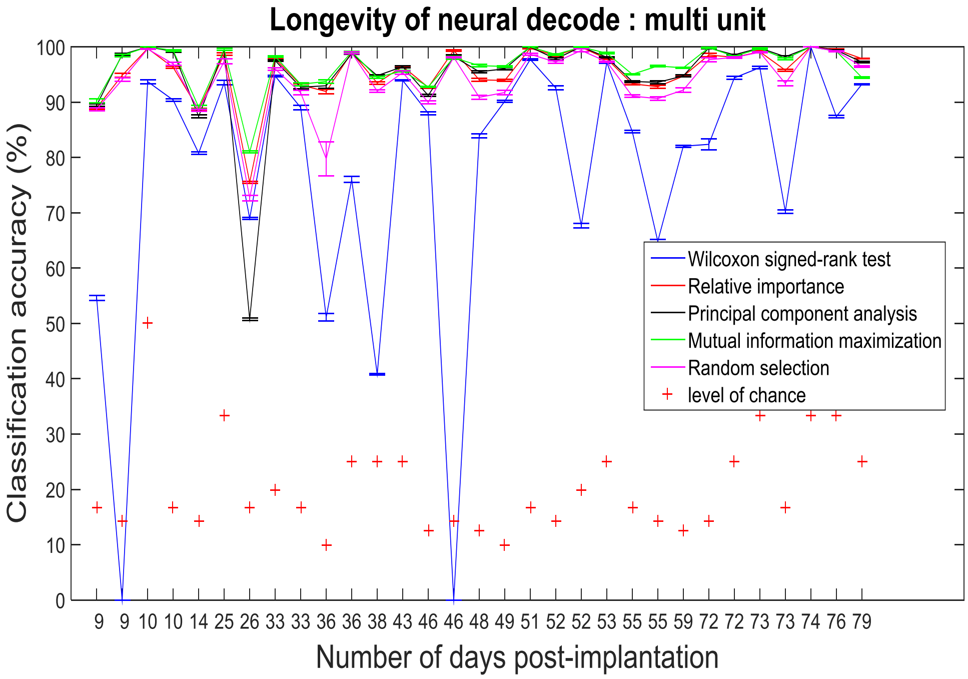

Figure 5a. Longevity of neural decodes using multi-unit firing rate. For a given session, 70% of all available successful trials were used as a training set and the remaining 30% were used for testing.

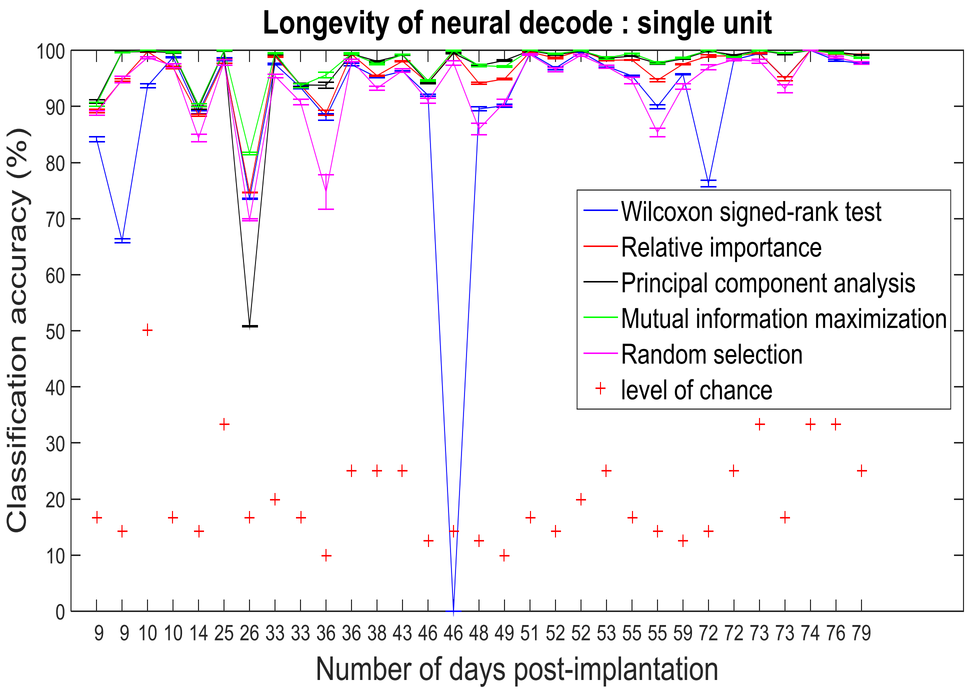

Figure 5b. Longevity of neural decodes using single-unit firing rate. The optimal number of features for each feature selection technique was identified using an iterative cross validation scheme. For a given session, 70% of all available successful trials were used as a training set and the remaining 30% were used for testing.

Classification accuracy of all feature selection algorithms except Wilcoxon signed rank-test using channel and single-unit firing rate had a difference of <~4% on average. The standard deviation of prediction for mutual information maximization was 7.09 while it was 18.51 for Wilcoxon signed-rank test (for single-unit firing rate). As it can be seen from Figures 5 a-b, the improvement in classification accuracy from channel to single-unit firing rate requires an action potential isolating pre-processing procedure. Relative importance, principal component analysis and mutual information maximization had comparable accuracies across 32 sessions and performed better than random selection of features.

CHAPTER 4

DISCUSSION

Comparing the efficacy of neural decodes based on input data, features from single-unit firing rate yield slightly better accuracy than features from multi-unit firing rate except for Wilcoxon signed-rank test. Principal component analysis and mutual information maximization are two commonly used feature selection techniques for time-series data. Comparing the raw classification accuracies based on feature selection techniques, Mutual Information Maximization performed better than the other feature selection techniques. It has better accuracy while using either multi-unit or single-unit firing rate. With the exception of Wilcoxon signed-rank test, the efficacy of neural decodes of the other feature selection techniques did not improve significantly for single-unit and multi-unit firing rates.

The performance of single-unit firing rate based features was more consistent and robust to simulated failure than multi-unit firing rate. For single-unit feature sets ~80-90% accuracy was obtained with only 10% of single-unit features for all feature selection methods. Mutual Information Maximization produced highest accuracies for simulated failure for both single-unit and multi-unit firing rates. Wilcoxon signed-rank test had the maximum improvement in endurance to simulated failure moving from multi-unit to single-unit firing rate.

Assessing the chronic decoding capability of various feature selection methods, Mutual Information Maximization produced the best results for both single-unit and multi-unit based firing rate. In general, single-unit firing rate feature vector yielded slightly better (~3-4% on average) performance compared to multi-unit firing rate feature vector for all feature selection methods except Wilcoxon signed-rank test. The chronic decoding results also validate the viability of using a neuroprosthetic device with high classification accuracies (> 90% on average and several folds better than level of chance).

Isolating the action potentials from individual neurons is routinely performed on neural recordings from microelectrodes. We have shown that by applying feature selection techniques to single-unit and multi-unit firing rates, we can get comparable performance on a chronic level. However, utilizing single-unit firing rates demonstrated better performance than multi-unit firing rates when the number of active electrodes decreased. Thus, the choice of input data (multi-unit or single-unit firing rate) and feature selection algorithm is more mandated by the primary need and available resources than convenience. We speculate that Mutual Information Maximization performs better across all three performance metrics as it maximizes the relevance of time-series features in the predictor space.

Future analysis will investigate the stability of neural decodes. Stability of neural decodes refers to the performance of a trained model over time without updating the model. The stability of neural decoding models will impact how often a user will need to retrain the classifier model.

REFERENCES

1. SP Kim, JD Simeral, LR Hochberg, JP Donoghue, MJ Black, " Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia", Journal of Neural Engineering, IOP Publishing, pp 455, 2008/12/05

2.JL Collinger, B Wodlinger, JE Downey, W Wang, EC Tyler-Kabara, DJ Weber, AJC McMorland, M Velliste, ML Boninger, AD Schwartz, “High-performance neuroprosthetic control by an individual with tetraplegia”, Lancet, 6736 : 61816-61819, 2012

3. Gilja V, Nuyujukian P, Chestek CA, Cunningham JP, Yu BM, Fan JM, Churchland MM, Kaufman MT, Kao JC, Ryu SI, Shenoy KV (2012) A high-performance neural prosthesis enabled by control algorithm design. *Nature Neuroscience*. 15:1752-1757

4. J Baker, W Bishop, S Kellis, T Levy, P House and B Greger, "Multi-scale recordings for neuroprosthetic control of finger movements", 31st International conference of the IEEE EMBS, Minnesota, USA, IEEE, pp 4573-4577, 2009/09/02

5. P. A. House, J. D. MacDonald, P. A. Tresco, and R. A. Normann, “Acute microelectrode array implantation into human neocortex: preliminary technique and histological considerations.” Neurosurg Focus, vol. 20, no. 5, p. E4, 2006.

6. HN Kim, YH Kim, HC Shin, V Aggarwal, MH Schieber, NV Thakor, "Neuron selection by relative importance for neural decoding of dexterous finger prosthesis control application", Biomed Signal Process Control, Elsevier, pp 632-639, 2012/11/01

7. G Brown, A Pocock, MJ Zhao, M Lujan, "Conditional likelihood maximization: a unifying framework for information theoretic feature selection", Journal of machine learning research, pp 27-66, 2012/01

8. RJ Vetter, JC Williams, JF Hetke, EA Nunamaker, DR Kipke,"Chronic neural recording using silicon substrate microelectrode arrays implanted in cerebral cortex",IEEE transactions on biomedical engineering, Vol. 51, No. 6, pp 896-904

9. S Shoham, MR Fellows, RA Normann, “ Robust, automatic spike sorting using mixtures of multivariate t-distributions”, Journal of Neurscince Methods, Vol. 127, pp 111-122

10. Carmena, J., Lebedev, M., Crist, R., O’Doherty, J., Santucci, D., Dimitrov, D., Patil, P., Henriquez, C., and Nicolelis, M. (2003). “Learning to control a brain-machine interface for reaching and grasping by primates”. PLoS Biology, pp 1:193–208.

11. Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., Branner, A., Chen, D., Penn, R. D., and Donoghue, J. P. (2006). “Neuronal ensemble control of prosthetic devices by a human with tetraplegia”. Nature, 442:164–171

12. Ganguly, K. and Carmena, J. M. (2009). “Emergence of a stable cortical map for neuroprosthetic control”. PLoS Biology, 7(7):e1000153+

13. Shenoy, K. V., Meeker, D., Cao, S., Kureshi, S. A., Pesaran, B., Mitra, P., Buneo, C. A., Batista, A. P., Burdick, J. W., and Andersen, R. A. (2003). “Neural prosthetic control signals from plan activity”. NeuroReport, 14:591–596

14. Musallam, S., Corneil, B., Greger, B., Scherberger, H., and Andersen, R. (2004). “Cognitive control signals for neural prosthetics”. Science, 305:258–262

15. Santhanam, G., Ryu, S. I., Yu, B. M., Afshar, A., and Shenoy, K. V. (2006). A high-performance brain-computer interface. Nature, 442:195–198

16. Wu W, Gao Y, Beinenstock E, Donoghue JP, Black MJ. (2006). “Bayesian population decoding of motor cortical activity using a Kalman filter”, Neural Computation, 18-1 pp 80-118

17. Wu W, Black MJ, Mumford D, Gao Y, Beinenstock E, Donoghue JP. (2004). “Modeling and decoding motor cortical activity using a switching Kalman filter”, IEEE Transactions on Biomedical Engineering, 51-6 pp 933-942

18. Milekovic T, Fischer J, Pistohl T, Ruescher J, Schulz-Bonhage A, Aertsen A, Rickert J, Ball T, Mhring C. (2012). “An online brain machine interface using decoding of movement direction from the human electrocorticogram”. Journal of Neural Engineering. pp 0464003

19. Kim SP, Sanchez JC, Rao YN, Erdogmus D, Carmena JM, Lebedev MA, Nicolelis MAL, Principe JC. (2006). "A comparison of optimal MIMO linear and non-linear models for brain machine interfaces". Journal of Neural Engineering. pp 145-161

20. Chestek CA, Gilja V, Blabe CH, Foster BL, Shenoy KV, Parvizi J, Henderson JM. (2013). "Hand posture classification using electrocorticography signals in the gamma band over human sensorimotor brain areas", Journal of Neural Engineering. 026002.

21. Hao Y, Zhang Q, Controzzi M, Cipriani C, Li Y, Li J, Zhang S, Wang Y, Chen W, Carrozza MC, Zheng X. (2014). "Distinct neural patterns enable grasp types decoding in monkey dorsal premotor cortex". Journal of Neural Engineering. 066011.

22. Wissel T, Pfeiffer T, Frysch R, Knight RT, Chang EF, Hinrichs H, Riecher JW, Rose G. (2013). "Hidden markov model and support vector machine based decoding of finger movements using Electrocorticography". Journal of Neural Engineering. 056020.

23. Guyon I, Elisseeff A. (2003). "An introduction to variable and feature selection". Journal of Machine Learning Research. Vol 3. pp 1157-1182

24. Liu H, Yu L. (2005). "Toward integrating feature selection algorithms for classification and clustering". IEEE Transactions on Knowledge and Data Engineering. 17-4. pp 1041-4347

25. Kohavi R, John GH. (1997). "Wrappers for feature subset selection". Artificial Intelligence. 0004-3702. pp 273-324

26. Platt J (1998). "Sequential Minimal Optimization: a fast algorithm for training support vector machines", Microsoft Research, MSR-TR-98-14

27. Kohavi R (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection", International Joint Conference on Artificial Intelligence 1995

APPENDIX -A

ALGORITHM AND CODE

This section of the appendix contains a brief overview of the algorithms used in different sections of the neural decoding architecture. It also contains the core pieces of the code used for analysis and generating the results.

Pre-processing

Computing multi-unit and single-unit firing rate from the NEV file. The NEV file contains time stamp of Action Potentials based on channels.

```
boxwin = 0.3; %boxcar window in sec
col = NEV.MetaTags.DataDuration;
Hd = design(fdesign.lowpass('N,F3dB',4,10,600),'butter'); %spikes initially downsampled
to 600S/sec for convolution, then down to 15
WF = [];
switch param
    case 'channel'
%           RateDS.Data = zeros(96,ceil(col/2000)); %total length for 15 S/sec
        RateDS.Data = zeros(96,ceil(col/1500)); %total length for 20 S/sec
        for k=1:96
            clc, disp(k)
            TS = double(NEV.Data.Spikes.TimeStamp(NEV.Data.Spikes.Electrode==k &
NEV.Data.Spikes.Unit~=0 & NEV.Data.Spikes.Unit~=255));
            %     WF =
double(NEV.Data.Spikes.Waveform(:,NEV.Data.Spikes.Electrode==k &
NEV.Data.Spikes.Unit~=0 & NEV.Data.Spikes.Unit~=255));
            %     RateDS.WfTS(k) = {TS./2000}; %timestamps downsampled to 15 S/sec
            %     RateDS.Wfs(k) = {WF};
            TS(TS <1) =1;
            RateB = zeros(1,ceil(col/50)); %total length for 600 S/sec
            if ~isempty(TS)
                RateB(ceil(TS./50)) = 1; %timestamps at 600 S/sec
                RateB = conv2(RateB,ones(1,boxwin*600)./boxwin);
                RateB = RateB(1:ceil(col/50));
                % RateB = conv2(RateB,gausswin(boxwin*300)'./(boxwin/2),'same'); %std =
N/5 (i.e. std = 90/5 = 18samples = 60ms)
                RateB = filter(Hd,RateB); %filter at 10Hz
%                   RateDS.Data(k,:) = RateB(1:40:end); %downsample from 600 S/sec to 15
S/sec
                RateDS.Data(k,:) = RateB(1:30:end); %downsample from 600 S/sec to 20
S/sec
            end
        end
        VarRate = RateDS.Data;
    case 'units'
        RateDS=struct([]);
        for k=1:96
            clc, disp(k)
            uniqueunits =
setdiff(unique(double(NEV.Data.Spikes.Unit(NEV.Data.Spikes.Electrode
==k))),[0,255]);
            % Compute the number of active units found in each electrode. Later
            % compute the firing rate for that particular units alone
            Data=zeros(numel(uniqueunits),ceil((col./1500)));
            for j=1:numel(uniqueunits)
                TS = double(NEV.Data.Spikes.TimeStamp(NEV.Data.Spikes.Electrode==k &
NEV.Data.Spikes.Unit==j));
                %     WF =
double(NEV.Data.Spikes.Waveform(:,NEV.Data.Spikes.Electrode==k &
NEV.Data.Spikes.Unit~=0 & NEV.Data.Spikes.Unit~=255));
                %     RateDS.WfTS(k) = {TS./2000}; %timestamps downsampled to 15 S/sec
                %     RateDS.Wfs(k) = {WF};
                TS(TS <1) =1;
                RateB = zeros(1,ceil(col/50)); %total length for 600 S/sec
                if ~isempty(TS)
                    RateB(ceil(TS./50)) = 1; %timestamps at 600 S/sec
                    RateB = conv2(RateB,ones(1,boxwin*600)./boxwin);
                    RateB = RateB(1:ceil(col/50));
                    % RateB = conv2(RateB,gausswin(boxwin*300)'./(boxwin/2),'same'); %std
= N/5 (i.e. std = 90/5 = 18samples = 60ms)
%                     RateB = filter(Hd,RateB); %filter at 7.5Hz
%                     Data(j,:) = RateB(1:40:end); %downsample from 600 S/sec to 15 S/sec
                    Data(j,:) = RateB(1:30:end); %downsample from 600 S/sec to 20 S/sec
                end
            end
            RateDS(k).Units=Data;
        end
        VarRate = vertcat(RateDS(:).Units);
End
```

Feature selection

```
for jj=1:5
            switch jj
                case 1
```

```
        [~,features] = findDrivenElects(ChanFeatures,ChanBaselines,'cns',baserate);
%Wilcoxon
            case 2
        [~,features] = relativeimp(ChanFeatures,ChanBaselines);
      % Relative Importance of features
      case 3
        [wcoeff,pcamat] = oldpca(Raw_Input','NumComponents',25);
        Raw_Input = pcamat';
            % Principal Component Analysis
      case 4
        features = randperm(numvariables,25);
            % Random features
      case 5
        features = feast('mim',25,Raw_Input',Output');
            % Mutual information maximization
    end
end
```

Support vector machine

```
[trainData, testData, trainLabel, testLabel] =
splitData(RawNewInput,Output,0.7,'random');
        t =
templateSVM('Standardize',1,'KernelFunction','gaussian','BoxConstraint',svmparam(jj),'K
ernelScale',svmparam(jj));
Mdl = fitcecoc(trainData',trainLabel','Learners',t,'Coding','onevsone');
predicted = predict(Mdl,testData');
```