TRACKING SINGLE-UNITS IN CHRONIC NEURAL RECORDINGS FOR BRAIN MACHINE INTERFACE APPLICATIONS

By

Ahmed Ibrahim Eleryan

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering - Master of Science

2013

ABSTRACT

TRACKING SINGLE-UNITS IN CHRONIC NEURAL RECORDINGS FOR BRAIN MACHINE INTERFACE APPLICATIONS

By

Ahmed Ibrahim Eleryan

Ensemble recording of multiple single-unit activity has been used to study the mechanisms of neural population coding over prolonged periods of time, and to perform reliable neural decoding in neuroprosthetic motor control applications. However, there are still many challenges towards achieving reliable stable single-units recordings. One primary challenge is the variability in spike waveform features and firing characteristics of single units recorded using chronically implanted microelectrodes, making it challenging to ascertain the identity of the recorded neurons across days. In this study, I present a fast and efficient algorithm that tracks multiple single-units recorded in non-human primates performing brain control of a robotic limb, based on features extracted from units' average waveforms and interspike intervals histograms. The algorithm requires a relatively short recording duration to perform the analysis and can be applied at the start of each recording session without requiring the subject to be engaged in a behavioral task. The algorithm achieves a classification accuracy of up to 90% compared to manual tracking. I also explore using the algorithm to develop an automated technique for unit selection to perform reliable decoding of movement parameters from neural activity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background

Extracellular recording is the observation of the membrane potential of an ensemble of neurons using a voltage sensing device. Different types of signals that can be extracted from extracellular recording are described in [36]. Invasive methods have higher time and space resolutions than non-invasive methods. In this study, we are primarily interested in action potentials. An action potential (AP), or a spike, is the main communication method between neurons and it is a short event ($< 1ms$) where the neuron's membrane potential rises and falls due to the inflow and outflow of different types of ions. These signals can be recorded invasively using a single electrode or an array of electrodes.

Figure 1.1 shows the neural trace recorded on an electrode. The signal is band-filtered using a Butterworth filter with frequency range 300-5000 Hz in order to extract spikes [36]. Detection of spikes occurs when the filtered signal crosses a user-defined threshold (typically equals to three times the root mean square of the noise [52]) and snippets of the signal surrounding the spikes are extracted which we call *"spike waveforms"*. Spike waveforms are time series (since they are subsets of the recorded signal) and they are aligned using one of two sample points: the threshold-crossing point or the waveform minimum peak.

The signal recorded in extracellular recordings is the aggregate contribution of the mem-

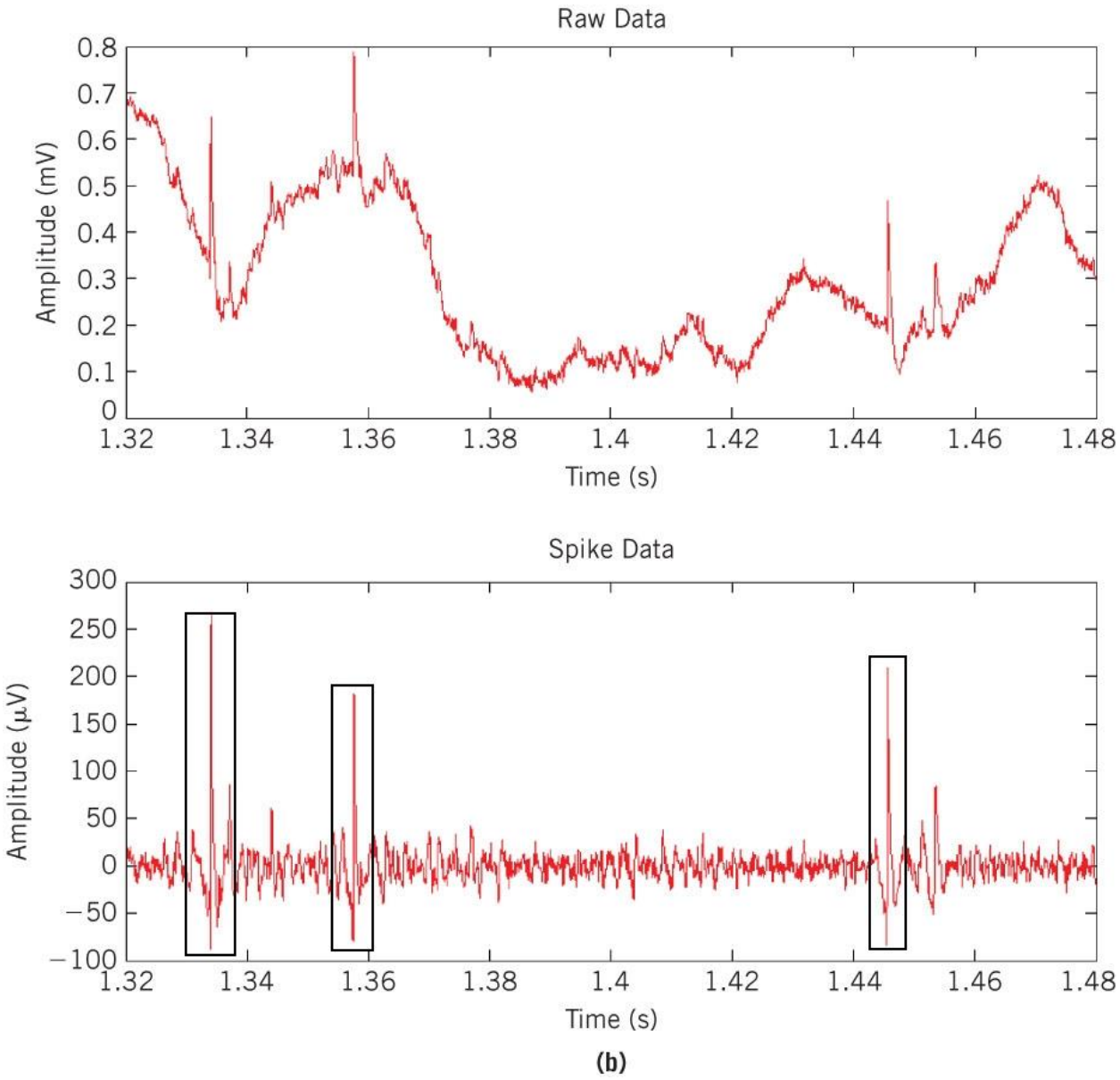

Figure 1.1: Figure a shows a sample neural trace recorded on an electrode. Figure b displays the APs detected in the trace (surrounded by boxes). The APs are obtained after filtering the raw data with Butterworth filter with frequency range 300-5000 Hz. The figure is taken from [36]. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis.

brane potential of all the neurons in the vicinity of the electrode (i.e. the spikes detected on one electrode might belong to more than one neuron). To be able to understand the brain encoding mechanisms and the response of individual neurons to different stimuli, we need to distinguish between the events of different neurons. Therefore, a spike sorting algorithm is needed to sort out the input spike waveforms into different clusters where each represents a single-unit (neuron) (Figure 1.2). There are many algorithms to perform spike sorting such as: using wavelets, principal components analysis, and k-means.

The spike waveforms are characteristic for each single-unit and their shape depends on the type of neuron and its location relative to the recording electrode [7]. The spike waveforms from a single-unit form a stochastic process, (denote it as W), consisting of a number of random variables equals to the number of samples in each waveform snippet.

$$\mathrm{W} = [\mathrm{W}(1) \dots \mathrm{W}(m)]^T \tag{1.1}$$

where m is the number of random variables in the waveforms stochastic process (= number of samples in the waveform snippets).

Each random variable reflects the variability at a time instance (i.e. sample) along the spike waveform and it can be modelled as [36]:

$$\mathrm{W}(t) = \mathrm{S}(t) + \mathrm{n}(t) \tag{1.2}$$

where $t \in [1, m]$ is the random variable index (or time instance), $\mathrm{W}(t) \in \mathbb{R}$ is the recorded signal, $\mathrm{S}(t) \in \mathbb{R}$ is the real membrane potential, and $\mathrm{n}(t)$ is the noise resulting from other neurons. $\mathrm{n}(t)$ can be assumed to be colored and Gaussian distributed with zero mean and

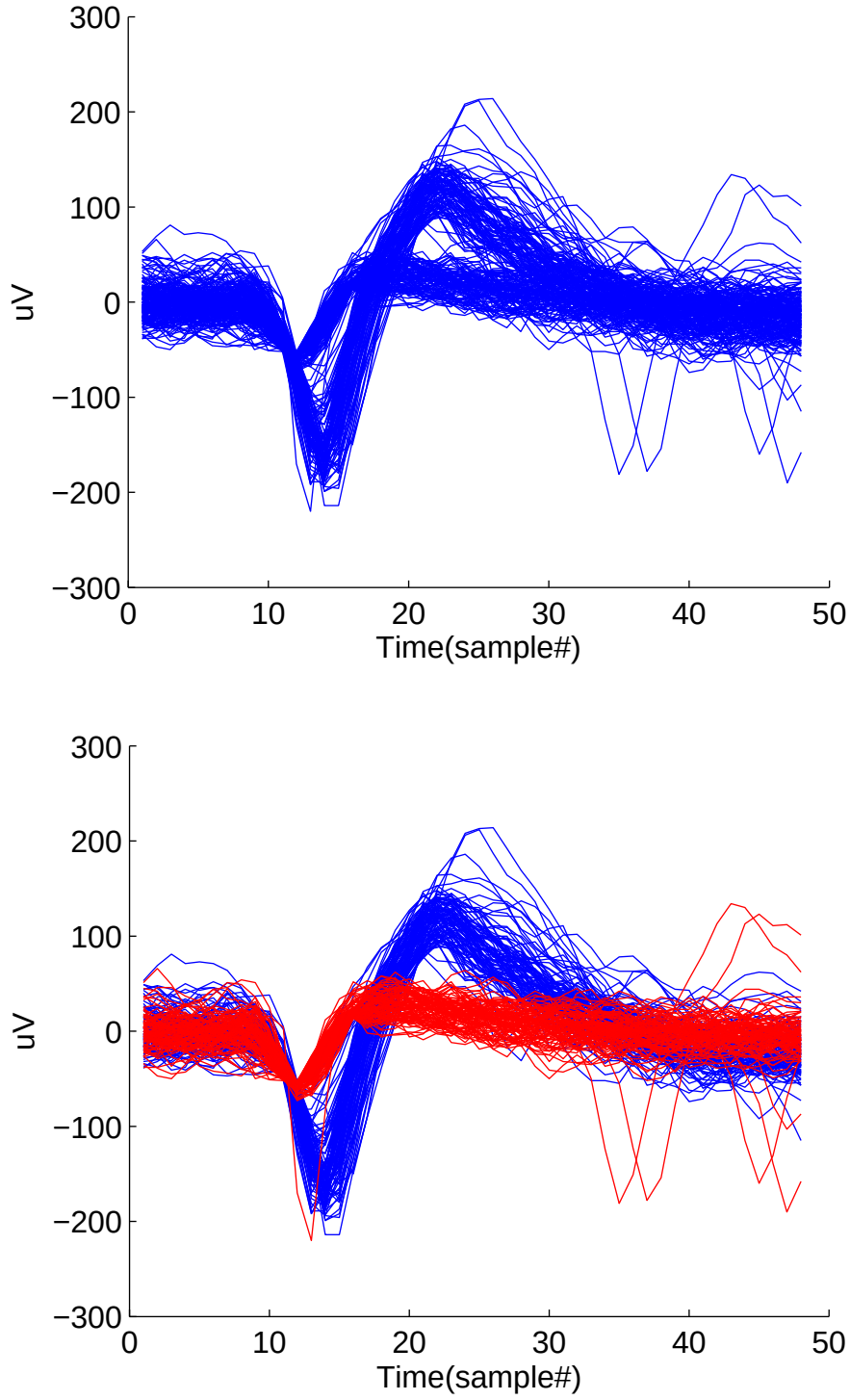

Figure 1.2: A spike-sorted channel with two single-units.

coveriance Σ (i.e. $n\ \mathcal{N}(0, \Sigma)$) [36]. Although the Gaussian assumption may not be realistic, it is reasonable for sufficiently large observation intervals. Each recorded waveform belonging to a single-unit is a realization of the stochastic process forming its waveforms:

$$\mathrm{W}_i = [\mathrm{W}^i(1) \dots \mathrm{W}^i(m)]^T \tag{1.3}$$

The average waveform for a single-unit with l recorded waveforms (i.e. realizations) is:

$$\bar{\mathrm{W}} = \left[\sum_{i=1}^{l} \mathrm{W}^i(1)/l \cdots \sum_{i=1}^{l} \mathrm{W}^i(m)/l \right]^T \tag{1.4}$$

Figure 1.3 displays an example of a single-unit waveforms stochastic process. The figure shows different realizations of the process and their average. The number of random variables in this process is 48. The waveforms were aligned based on the point of threshold crossing.

A spike train is a sequence of zeros and ones at millisecond precision. A "one" means the occurrence of a spike at that time instance, and a zero means the absence of a spike. Another single-unit characteristic that can be inferred from its spike train is the "*interspike interval histogram (ISIH)*", which is a popular feature to characterize the firing pattern of a neuron. An ISIH is a plot of the distribution of the times between a neuron's spiking events. The standard ISIH fits an exponential distribution reflecting the underlying homogenuous poisson process that governs the firing of the neuron [10]. Each neuron has a refractory period after each spike where it is hard (but not impossible) for it to fire another spike. In the presence of a stimulus in the receptive field of a neuron, the firing characteristics of the neuron become time-varying and they can be modelled by an inhomogenuous poisson process. During that period, the shape of the neuron's ISIH changes to reflect the changes

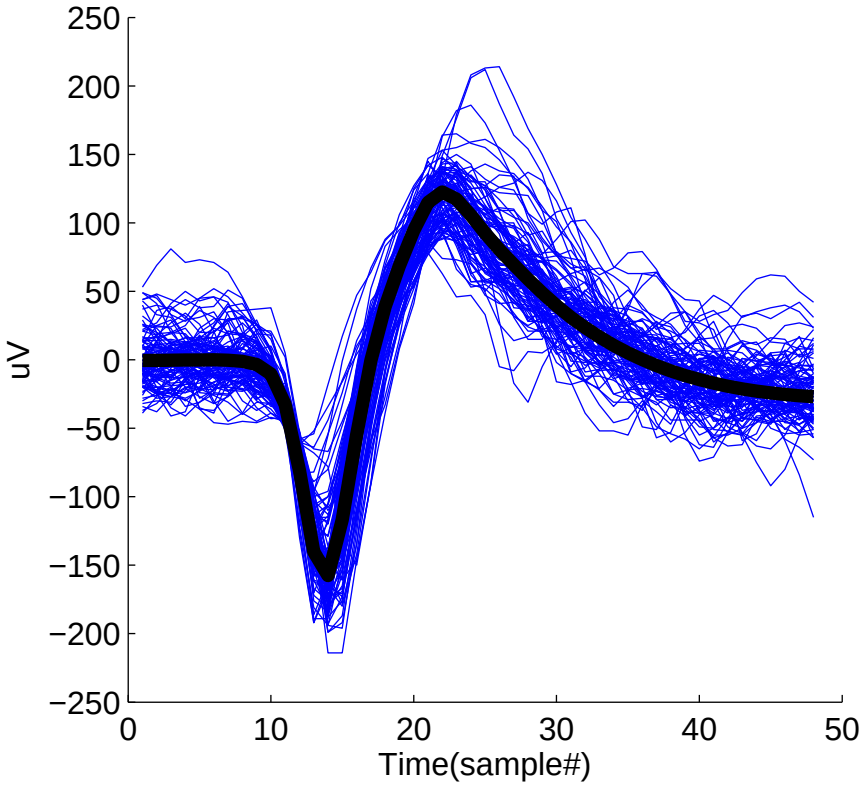

Figure 1.3: Recorded waveforms of a single-unit along with their average. The unit's waveforms form a stochastic process with a number of random variables equals to the number of sample in each waveform snippet (in this case each waveform is made up of 48 samples). Each waveform is a realization of this stochastic process.

in the surrounding environment [40]. The ISIH can also be used to show whether the neuron phase-locks to a periodic input.

Next, I show how a unit's ISIH is computed. Let n be the number of desired bin in the histogram, ST be the set of the unit's time stamps. Then the time interval between any two consecutive spikes is:

$$\Delta t_i = ST_{i+1} - ST_i \tag{1.5}$$

where $i \in [1, \|ST\| - 1]$ is the index of the spike.

Let B be the set of bin boundaries in the ISIH such that:

$$B_0 = 0 \tag{1.6}$$

$$B_i = B_{i-1} + step, \ \forall i \in [1, n] \tag{1.7}$$

where the bins step is defined as:

$$step = \frac{max(\Delta t) - min(\Delta t)}{n} \tag{1.8}$$

Finally, the value of the ith ISIH bin is

$$H[i] = \{\Delta t_i : \Delta t_i \geq B_i, \Delta t_i < B_{i+1}\}, \ \forall i \in [1, n-1] \tag{1.9}$$

with the only exception:

$$H[n] = \{\Delta t_n : \Delta t_n \geq B_n\} \tag{1.10}$$

Figure 1.4 shows the spike train of a single-unit and the ISIH inferred from it.

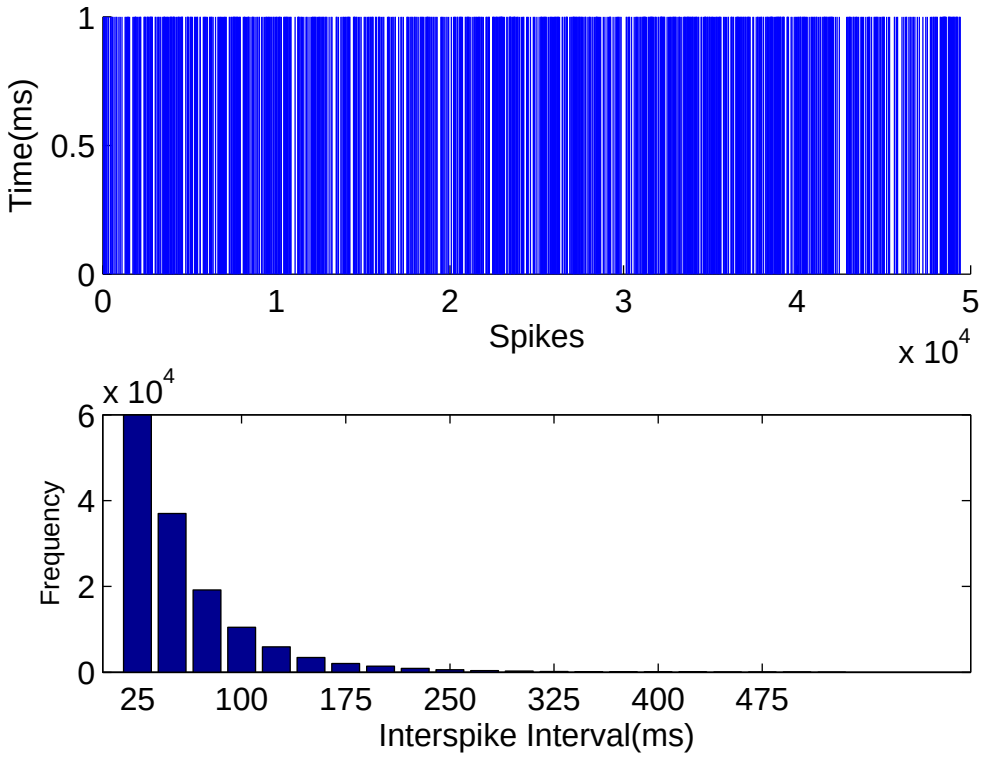

Figure 1.4: The spike train for a single-unit is shown in Figure a (top) and the ISIH inferred from it is shown in Figure b (bottom).

1.2 Challenges Facing Brain Machine Interface Systems

While chronically implanted multi-electrode silicon arrays have been very popular in recording neural signals, these recordings, however, frequently exhibit variability in spike waveform shapes and firing characteristics over days and months, making it hard to assess whether recordings on consecutive days are made from the same neurons [9, 31]. This variability potentially results from micro-movements of neurons and electrodes, and the brain tissue reaction to the abiotic interface (electrodes), or from functional plasticity [15, 16, 37]. This variability presents a major challenge to neural decoding - the process of translating the firing pattern of a fixed subset of the recorded neurons into control signals to maintain a fixed mapping between the neural input space and the kinematic task space. Most of the current decoding techniques necessitate the calibration of the decoder at the start of each session where signals are recorded while the subject observes the task being performed and a new decoder is trained using those signals. This process is a major limitation towards the practical use of Brain Machine Interface (BMI) systems because it requires an operator to perform this operation.

For subjects (non-human primates) to routinely use decoders without the need for frequent calibration, the mapping from the neural input space to the kinematic task space needs to be fixed, and that requires the accurate identification of neurons to use at the start of each recording session, preferably when the subject is not engaged in any behavioral task. Using the same set of neurons during each session would ensure the fixation of this mapping. Hence, there is a need for a fast and efficient single-unit tracking algorithm that analyzes neural data recorded over a short duration of time and that uses only features extracted

from the units' waveforms (as opposed to their stimulus-driven firing characteristics) for the decoder's routine calibration. This can pave the way for fully automated tracking and thereby eliminating the need for human supervision all-together.

1.3 Related Work

Previous work on assessing the stability of these signals has focused on qualitative measures to track units based primarily on visual inspection [6, 8, 15, 29, 34, 41]. Few studies have attempted to develop automated solutions to the unit tracking problem. Jackson et al. [23] tracked units using the peak of the normalized cross-correlation between average waveforms. They reported that 80% of the stable units had correlation values of > 0.95 across days.

In [45], Suner et al. approach the problem from a signal reliability perspective. They compare the signals recorded on the same channel across days by comparing the centers of clusters in the principal components space and the Kolmogorov-Smirnov statistic between interspike intervals histograms of the formed clusters.

Movable tetrodes were used in [46]. The authors determined a so-called "null distribution" of pairwise distances by comparing signals before and after adjusting the depths of the tetrodes. The "null distribution" can then be used to train a classifier to track the stability of single-units. Their method is unapplicable in the case of a fixed array.

Dickey et al. [11] trained a classifier on features extracted from both the unit's average waveform and interspike interval histogram (ISIH). The authors argued that using both the average waveforms and ISIHs gave a better accuracy. However, they also stated that subjects were overtrained on the behavioral task which minimized the variability of the units' ISIHs across days. Furthermore, the method the authors used to match ISIHs is

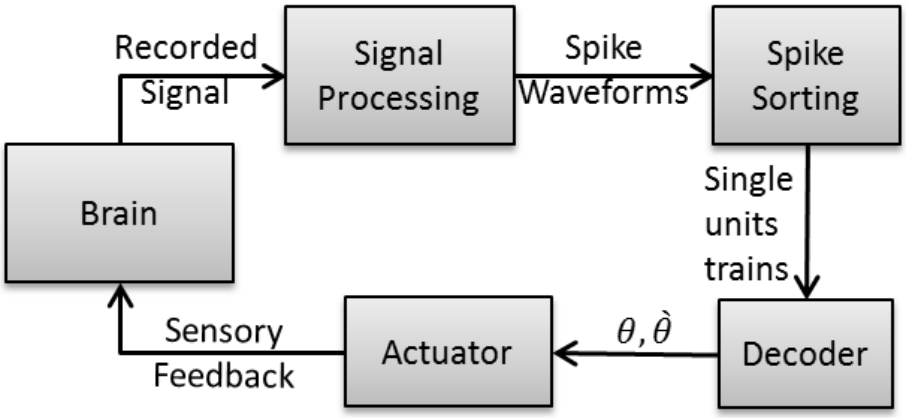

Figure 1.5: Components of a typical motion-restoration BMI system: a recording device, a signal processing module to extract information from raw signals, a spike sorter to identify single-units, a decoder to map the neuronal input signal to movement parameters, and an actuator to perform the action.

computationally intensive and time consuming, which makes it unsuitable for real time Brain Machine Interface applications. The authors reported that 57% of the units were stable through 7 days and 39% of the units were stable through 15 days. They concluded that if units are stable beyond the 7 days time marker, they are more likely to remain stable in subsequent recording sessions.

Fraser et al. [14] used features extracted from functional relations between units, such as pairwise cross correlograms, to track units across days. However, this method requires the subject to be engaged in a behavioral task and may be susceptible to plasticity-mediated changes in functional connectivity between units.

1.4 BMI Systems

A BMI system divides its operation among several sequential modules (Figure 1.5): a recording device, a signal processing module, a spike sorting module, a decoding module and an actuator [18, 35].

The neuronal signals recorded from the brain undergo some signal processing such as: band-pass filtering to extract spiking events, denoising of the extracted spikes (typically by thresholding), and artifact rejection. The processed signal is then spike sorted to produce spike trains of individual neurons. Spike trains are all-or-none representation of the spiking events of a neuron. Spike trains are then input to the decoding module. A decoder is what translates the neuronal activity into motion. It extracts information from the spike trains and outputs the equivalent movement parameters to perform the desired motion.

In this thesis, I present a fast and accurate single-unit tracking algorithm that uses features extracted from single-units' waveform shapes and firing characteristics. The algorithm achieves an accuracy of up to 90% when compared to manual tracking. I describe the details of the algorithm in Chapter 2. I also present one application of single-units tracking in Chapter 3: unit selection for BMI decoding.

Chapter 2

Single-Unit Tracking

In this chapter, I present an algorithm to track neurons recorded using a chronically implanted silicon array. Section 2.1 gives an overview of the algorithm. Sections 2.2 and 2.3 list some features that can be extracted from single-units. In Section 2.4, I describe four types of classifiers that can be used in the tracking algorithm, and in Section 2.5 I describe the algorithm in more details. Finally, I show the results I obtained by applying the tracking algorithm in Section 2.6 .

2.1 Algorithm Overview

Figure 2.1 illustrates the algorithm flow diagram. The algorithm builds profiles of single-units from instances of unit recordings across days. The inputs to the tracking module are the spike sorted units recorded on a given day, and the output is a set of updated profiles of single-units for that day. The tracking module makes the choice of either assigning a unit to an existing profile or creating a new profile for that unit on any given day. The distance vectors used in this decision are computed using different characteristics of a single-unit: the waveform shape, and firing patterns.

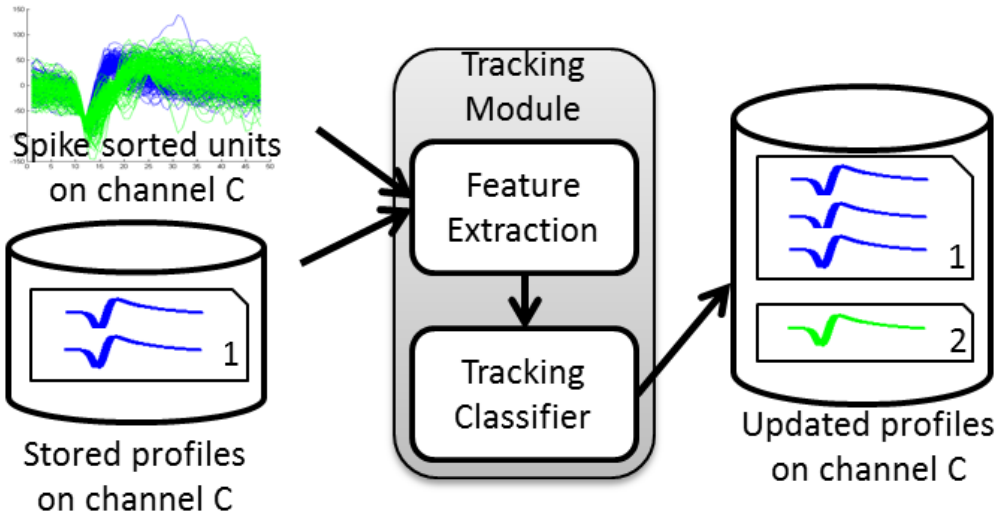

Figure 2.1: Tracking Algorithm: The tracking module takes in the spike sorted units and the profiles stored on the channel and calculates the distance between units and stored profiles. It then assigns each unit to either an existing profile or a new profile based on the distance computed. The output is the updated set of profiles for that channel.

2.2 Single-Unit Waveform Characteristics

A single-unit's average waveform (defined in Equation 1.4) can be used to distinguish it from the surrounding units. Waveforms features, such as amplitude, temporal dynamics, time between minimum and maximum peak and peaks shapes, can all be used to track single-units across days. However, due to small movements in the recording electrode's location relative to the single-unit, the shape of the recorded waveform differs across days. Therefore, we need measures to compute how similar two waveforms recorded on the same channel on two different days, and based on such measures, we could decide whether the two waveforms are recorded from the same single-unit.

In this study, I chose four distance measures to quantify the difference between two recorded average waveforms $\bar{\mathrm{W}}_x$ and $\bar{\mathrm{W}}_y$. Table 2.1 lists the chosen measures and the waveform features each of them uses to compute the distance.

Table 2.1: Waveform-derived distance measures

Distance Measure	Used Feature
Correlation	Waveform temporal dynamics
Normalized Peak-to-Peak Height Difference	Amplitude
Normalized Peak-to-Peak Time Difference	Transition time between minimum and maximum peaks
Peak Matching	Peaks shape

2.2.1 Correlation

The Correlation (PC) is used to compare the temporal dynamics of the waveforms during an action potential. The PC is calculated by:

$$PC(\bar{\mathrm{W}}_x, \bar{\mathrm{W}}_y) = \frac{\sum_{i=1}^{m} (\bar{\mathrm{W}}x(i) - \bar{\bar{W}}_x)(\bar{\mathrm{W}}y(i) - \bar{\bar{W}}_y)}{\sqrt{\sum_{i=1}^{m} (\bar{\mathrm{W}}x(i) - \bar{\bar{W}}_x)^2}\sqrt{\sum_{i=1}^{m} (\bar{\mathrm{W}}y(i) - \bar{\bar{W}}_y)^2}} \tag{2.1}$$

where m is the number of samples in each of the two average waveforms, $\bar{\bar{W}}$ is the mean of an average waveform.

2.2.2 Normalized Peak-to-Peak Height Difference

The Normalized Peak-to-Peak Height Difference (PH) is defined as:

$$PH(\bar{\mathrm{W}}_x, \bar{\mathrm{W}}_y) = \left| \frac{\big(max(\bar{\mathrm{W}}_y) - min(\bar{\mathrm{W}}_y)\big) - \big(max(\bar{\mathrm{W}}_x) - min(\bar{\mathrm{W}}_x)\big)}{\big(max(\bar{\mathrm{W}}_x) - min(\bar{\mathrm{W}}_x)\big)} \right| \tag{2.2}$$

where max and min are the maximum and minimum functions.

This metric quantifies the difference in the amplitude of the two waveforms and normalizes it to the amplitude of one of them. Therefore, a larger difference between two high amplitude

units would result in a smaller PH distance than if the difference was between two low amplitude units. This is desirable to reflect that a large difference between two low amplitude units indicate a higher variability than in the case of two high amplitude units.

2.2.3 Normalized Peak-to-Peak Time Difference

The Normalized Peak-to-Peak Time Difference (PT) is defined as:

$$PT(\bar{\mathrm{W}}_x, \bar{\mathrm{W}}_y) = \left| \frac{\big(t[max(\bar{\mathrm{W}}_y)] - t[min(\bar{\mathrm{W}}_y)]\big) - \big(t[max(\bar{\mathrm{W}}_x)] - t[min(\bar{\mathrm{W}}_x)]\big)}{\big(t[max(\bar{\mathrm{W}}_x)] - t[min(\bar{\mathrm{W}}_x)]\big)} \right| \tag{2.3}$$

where $t[.]$ is a function that retrieves the sample number (i.e. time instance) of the passed argument.

This metric quantifies the change in the transition time between the minimum and maximum peaks. Combined with the PH, it essentially compares the slopes of the lines joining the minimum and maximum peaks of the two waveforms.

2.2.4 Peak Matching

The Peak Matching (PM) is a heuristic asymmetric measure developed by Strelkov [44] that measures the difference in the shapes of peaks of two signals. The algorithm used to calculate the distance finds the peaks in both signals and assigns weights to each one of them. It then finds heuristic pairwise closeness factors between peaks and finally calculates the total distance as the sum of peak weights multiplied by the computed closeness factors. Refer to Appendix A for a full description of this metric.

2.3 Single-Unit Firing Characteristics

Different single-units have different firing patterns and these patterns can be used to distinguish units from each other [7, 30]. Several features can be extracted from a unit's firing patterns such as:

- Average firing rate (λ)
- Coefficient of variation (CoV)
- Interspike interval histogram (ISIH)

I chose to use the ISIH since it can provide more information about firing characteristics of a neuron and how it responds to external stimuli.

Distance measures that are used to compare two probability distributions can be used to compare two normalized ISIHs H_x and H_y. I chose four of such distance measures in this study, where each quantifies the differences between two probability distributions from a different prespective. Table 2.2 lists the chosen measures and the criterion each of them uses to compute the distance.

Table 2.2: ISIH-derived distance measures

Distance Measure	Distance Criterion
KL-Divergence	Total information divergence
Bhattacharyya Distance	Amount of overlap
KS-Statistic	Maximum divergence
Earth Mover's Distance	Cost of turning one dist. to the other

2.3.1 KullbackLeibler Divergence

The KullbackLeibler Divergence (KLD) is a non-symmetric measure that measures how divergent two statistical populations are [27].

The KLD is calculated as:

$$D(\mathrm{H}_x||\mathrm{H}_y) = \sum_{i=1}^{l} \ln\left(\frac{\mathrm{H}_x[i]}{\mathrm{H}_y[i]}\right)\mathrm{H}_x[i] \tag{2.4}$$

where l is the number of bins in each ISIH, $\mathrm{H}[i]$ is the value of the i^{th} bin in the histogram H.

To compute a symmetric measure out of the KLD, we simply compute the following:

$$D(\mathrm{H}_x, \mathrm{H}_y) = \frac{D(\mathrm{H}_x||\mathrm{H}_y) + D(\mathrm{H}_y||\mathrm{H}_x)}{2} \tag{2.5}$$

2.3.2 Bhattacharyya Distance

The Bhattacharyya Distance (BD) is closely related to the Bhattacharyya coefficient (BC) which is a measure of the amount of overlap between two statistical populations [3]. It is calculated using:

$$BD(\mathrm{H}_x, \mathrm{H}_y) = -\ln BC(\mathrm{H}_x, \mathrm{H}_y) \tag{2.6}$$

$$BC(\mathrm{H}_x, \mathrm{H}_y) = \left(\sum_{i=1}^{l} \sqrt{\mathrm{H}_x[i]\mathrm{H}_y[i]}\right) \tag{2.7}$$

where l is the number of bins in each ISIH, $\mathrm{H}[i]$ is the value of the i^{th} bin in the histogram H.

2.3.3 KolmogorovSmirnov Statistic

The KolmogorovSmirnov Statistic (KS) quantifies the distance between two empirical distributions [19] and it is calculated as:

$$KS(x, y) = \sup_{i=1}^{l} \left| \mathrm{F}_x[i] - \mathrm{F}_y[i] \right| \tag{2.8}$$

where F_x and F_y are the cumulative distribution function derived from H_x and H_y, l is the number of bins in each ISIH, and $\mathrm{F}[i]$ is the value of the cumulative distribution at the i^{th} bin.

2.3.4 Earth Mover's Distance

The Earth Mover's Distance (EMD) is a measure of the distance between two probability distributions. The EMD is the minimum cost of turning one distribution into the other by moving distribution mass around. For 1-D distributions, the EMD is computed by the following algorithm:

$$EMD_0 = 0 \tag{2.9}$$

$$EMD_{i+1} = (\mathrm{H}_x[i] + EMD_i) - \mathrm{H}_y[i] \tag{2.10}$$

$$EMD(\mathrm{H}_x, \mathrm{H}_y) = \sum_{i=1}^{l} |EMD_i| \tag{2.11}$$

where l is the number of bins in each ISIH, $\mathrm{H}[i]$ is the value of the i^{th} bin in the histogram.

Each of the previous distance measures computes the distance between two probability distributions from different perspectives. While the KLD calculates the total divergence

between two distributions, the *KS* approximates that divergence by just including the maximum difference at any point in the cumulative distributions. Both the *KLD* and the *KS* impose different constraints when comparing two distributions. The former measure overlooks individual points and is only concerned by the total divergence across all the points, which is opposite to what the latter measure does. The *BD* quantifies the amount of overlap between the two distributions to decide how similar they are, and the *EMD* computes the minimum cost of converting one distribution to the other.

2.4 Tracking Classifier

Given a vector of distances between the features extracted from two single-units recorded on the same channel on two different days, the tracking classifier makes the decision of whether these two single-units represent different instances of the same single-unit.

For training a waveform-based classifiers, true positives, which correspond to pairs of matched waveforms, were obtained by two experts who manually tracked the units. True negatives, which correspond to pairs of unmatched waveforms are obtained by comparing units recorded simultaneously on the same channel on any given day. Figure 2.2 illustrates this process. The same process can be applied for training classifiers that use other single-units characteristics.

Several types of classifiers were compared in this study. The following few sections give a brief summary of the classifers I compared. The input to all classifier is a vector, x, of the distances between the features extracted from two units recorded on the same channel but on different days. The length, n, of the distance vector is the number of features used in the comparison. The classifiers' decision is binary, either to assign x to class 0 (c_0), which

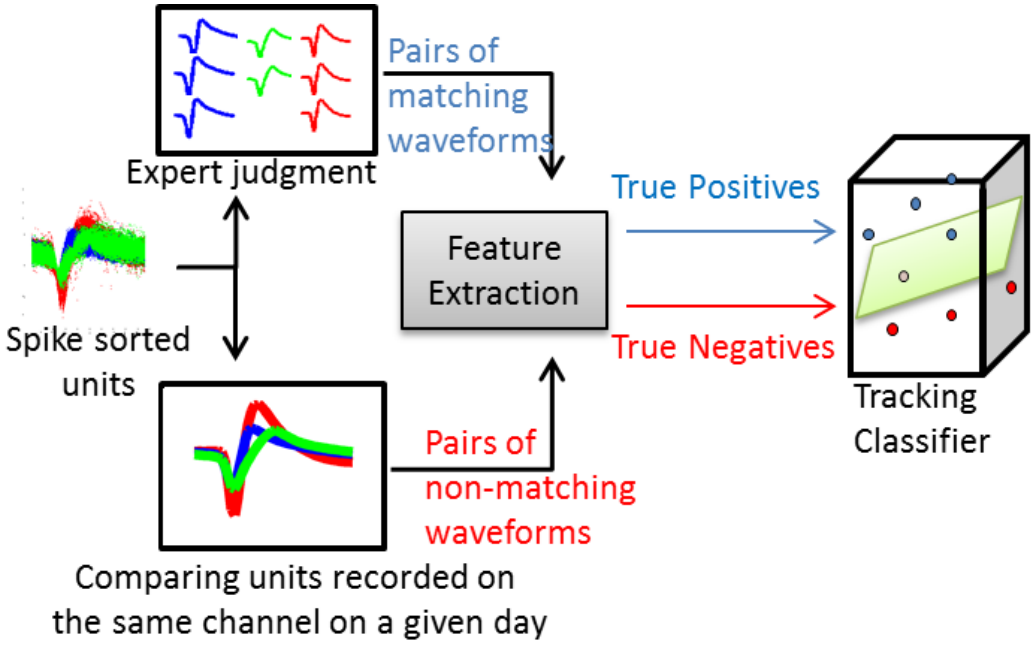

Figure 2.2: Process of training a waveform-based classifier. Using spike sorted units recorded across several days, experts track the units manually and produce pairs of matching average waveforms (true positives). Using the same set of units, pairs of non-matching average waveforms are generated by comparing spike sorted units recorded simultaneously on the same channel.

corresponds to non-matching units, or to class 1 (c_1), which corresponds to matching units.

2.4.1 Support Machine Vectors (SVM)

The SVM is a supervised binary classification method that finds a high-dimensional decision plane where the "margin" is maximized. The margin is defined as the smallest distance between the decision plane and any of the data points. The SVM model is in the form [33]:

$$\mathrm{y(x)} = \mathrm{w}^{\mathrm{T}}\phi(\mathrm{x}) + \mathrm{b} \tag{2.12}$$

where, w defines the decision plane, x is the input distance vector, ϕ(x) is projection of the input into the feature space (basis function), and b is a offset term to account for any biasedness in the training data.

Although the SVM model in Equation 2.12 is linear, non-linearity can be added to the model by the use of a non-linear basis function such a Gaussian radial basis function.

2.4.2 Relevance Machine Vectors (RVM)

The RVM is a Bayesian sparse kernel technique for classification and it shares many of the characteristics of the SVM. However, unlike the SVM, which gives the classification decision, the RVM provides the classification probability. In other words, the RVM combines the advantages of the sparse SVM and the probabilistic Bayesian methods. The model then becomes of the form [33]:

$$\mathrm{y(x)} = \sigma\left(\mathrm{w}^{\mathrm{T}}\phi(\mathrm{x}) + \mathrm{b}\right) \tag{2.13}$$

where, $\sigma(.)$ is the sigmoid function.

2.4.3 Maximum A Posteriori Classification (MAP)

The MAP is a probablistic classifier and it is used to estimate how likely a new point belongs to a distribution based on empirical data. In other words, the MAP finds the most probable class given the available training data.

Given the distance vector between two units, x, the MAP classification would be [33]:

$$p(x_1, \ldots, x_n|c_1)P(c_1) \underset{c_0}{\overset{c_1}{\gtrless}} p(x_1, \ldots, x_n|c_0)P(c_0) \tag{2.14}$$

where, $p(x_1, \ldots, x_n, |c_0)$ and $p(x_1, \ldots, x_n, |c_1)$ are the likelihood probabilities that x belongs to c_0 and c_1 respectively, and $P(c_0)$ and $P(c_1)$ are the prior probabilities of c_0 and c_1 respectively.

The likelihood and prior probability distributions are constructed using the training data, where the true positives are instances of c_1 and the true negatives are instances of c_0.

2.4.4 Naive Bayesian Classification (NB)

The NB is a simple probabilistic classifier that is based on Bayes theorem, but unlike the MAP, the NB assumes independence between features. Therefore, instead of using a joint conditional probability distribution as in the case of MAP, the NB uses multiple individual conditional probability distributions to infer its decision. Despite its unrealistic features independence assumption, the NB is surprisingly effective in practice [12].

For a distance vector between two units, x, the NB decision is [4]:

$$\prod_{i=1}^{n} p(x_i|c_1) \underset{c_0}{\overset{c_1}{\gtrless}} \prod_{i=1}^{n} p(x_i|c_0) \tag{2.15}$$

where $p(x_i|c_0)$ and $p(x_i|c_1)$ are the conditional probability distributions that the i^{th} feature of x belongs to c_0 and c_1 respectively.

As in the MAP, the conditional probability distributions are constructed using the training data, where the true positives are instances of c_1 and the true negatives are instances of c_0.

2.5 Tracking Algorithm

2.5.1 Pre-processing

The average waveforms were smoothed using a Gaussian kernel ($\sigma = 2$) to get rid of the high frequency components (which results from the short amount of recording tie I used).

2.5.2 Algorithm

As stated above, the tracking algorithm builds a database of profiles of recorded single-units. Each profile stores instances of the corresponding single-unit recorded across days. On a given day x, units on a channel c are spike sorted and passed to the feature extraction module. Pairwise distance vectors are computed between the stored profiles on c and the input spike sorted units. A distance vector is the concatenation of the distance measures computed over an input unit and an instance stored in a profile. A distance matrix is built using the pairwise distance vectors between the stored profiles and the spike sorted units. The distance matrix is then input to the tracking classifier to build a membership matrix, where each cell indicates whether an input single unit is another instance of a stored profile:

Denote the set of profiles stored on channel c as P_c, the set of spike sorted input units

on channel c as U_c, the distance between the ith profile and the jth unit as $Dist(P_{c,i}, U_{c,j})$, and the membership of the jth unit to the ith profile as $Mem(P_{c,i}, U_{c,j})$. Assume there are m stored profiles on c, and n spike sorted units on day x. The distance matrix is formed as:

$$DistMat_c(P,U) = \begin{bmatrix} Dist(P_{c,1},U_{c,1}) & Dist(P_{c,1},U_{c,2}) & \dots & Dist(P_{c,1},U_{c,n}) \\ Dist(P_{c,2},U_{c,1}) & Dist(P_{c,2},U_{c,2}) & \dots & Dist(P_{c,2},U_{c,n}) \\ \vdots & \vdots & \ddots & \vdots \\ Dist(P_{c,m},U_{c,1}) & Dist(P_{c,m},U_{c,2}) & \dots & Dist(P_{c,m},U_{c,n}) \end{bmatrix}$$

And the membership matrix is:

$$MemMat_c(P,U) = \begin{bmatrix} Mem(P_{c,1},U_{c,1}) & Mem(P_{c,1},U_{c,2}) & \dots & Mem(P_{c,1},U_{c,n}) \\ Mem(P_{c,2},U_{c,1}) & Mem(P_{c,2},U_{c,2}) & \dots & Mem(P_{c,2},U_{c,n}) \\ \vdots & \vdots & \ddots & \vdots \\ Mem(P_{c,m},U_{c,1}) & Mem(P_{c,m},U_{c,2}) & \dots & Mem(P_{c,m},U_{c,n}) \end{bmatrix}$$

Finally, the relationship between the distance and membership matrices can be modeled as:

$$DistMat_c(P,U) \Rightarrow TrackingClassifier \Rightarrow MemMat_c(P,U) \tag{2.16}$$

Units are added as instances to the profiles they belong to. If a new unit is identified, a new profile is initialized and the unit instance is added to it.

A profile is composed of all the recorded instances of the single-unit it represents. When an input unit is compared to a profile to make the decision of whether the unit is another instance of the profile, the tracking classifier needs to decide which profile instance to use in such comparison. In [11], the authors stated that once a unit remains stable (does not

experience much variability) beyond a threshold (seven days), it is more likely to maintain its stability. Based on this result, I developed a technique that I call "*First Matching Distance (FMD)*". In this technique the most recent seven instances in a profile are used in this comparison from newest to oldest. Once the tracking classifier indicates that a profile instance and the unit are recorded from the same singe-unit (a match), the comparison stops and this instance is used for distance calculations. Furthermore, two classes of profiles are defined: active and inactive. Active profiles are those profiles whose single-units have been recorded on any day within the seven days before. On the other hand, inactive profiles are those profiles whose units did not appear in the recording on any day during the seven days before, in which case I consider them "*dropped units*" and I do not include them in future comparisons.

2.5.3 Post-processing

The training classifier results in a membership matrix where each cell in this matrix indicates whether an input unit on a given day can be added to a stored profile or not. In many cases, because of the similarity of the waveform shapes of different units or misclassifications in the spike sorting algorithm, the classifier might indicate that the input unit can be added to more than one profile. However, a unit can be only added to one profile. Therefore, I developed a backtracking algorithm to find the best assignment of units instances to profiles. Backtracking is a standard general algorithm for finding all the possible solutions to a problem and then selecting one that most satisfies a set of user-defined objectives. The objectives I specified were as follows:

- **Primary objective**: Maximize the number of assigned units to already existing stored

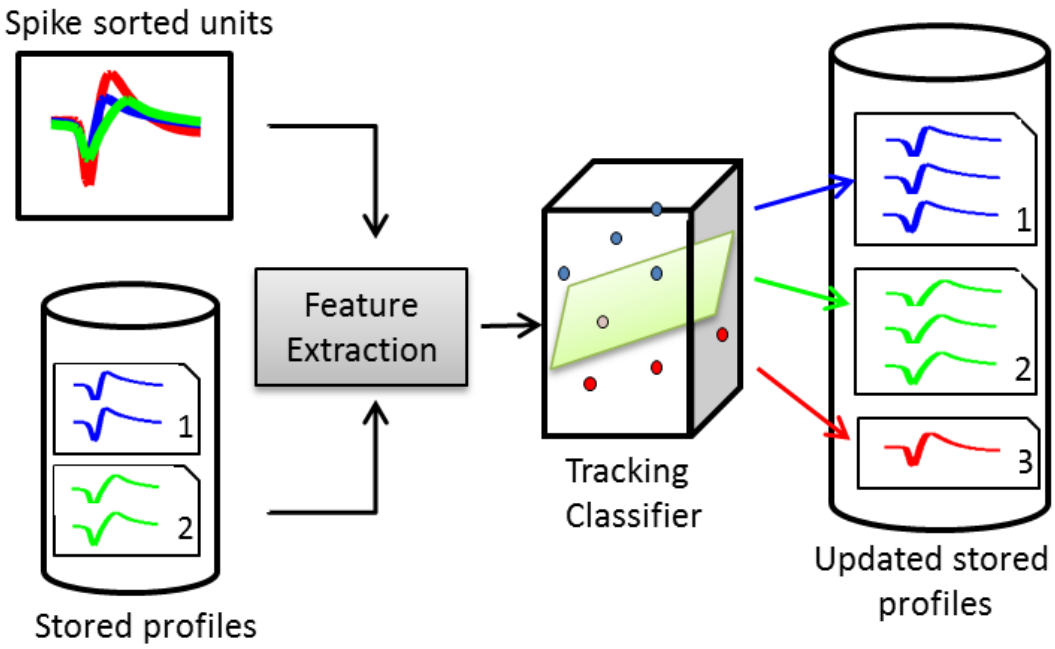

Figure 2.3: Application of a waveform-based classifier. On any given day x, the spike sorted units of a given channel c are compared against the stored profiles on the same channel c. Pairwise distance vectors are calculated from features extracted from pairs of waveforms. The tracking classifier makes the decision of whether to add a unit to an existing profile or to create a new profile for it.

profiles.

- **Secondary objective**: Maximize the total sum of distances to the decision boundary of the assigned units (since the larger the distance to the boundary, the more certain the decision of the tracking classifier is).

If two solutions have the same number of units assigned to existing stored profiles, the one with a larger sum of distances to the decision boundary is selected.

The tracking algorithm, on any given day x, is summarized in Algorithm 1.

Algorithm 1 Tracking algorithm

for each channel c **do**
  Spike sort the signal recorded on c
  Input the spike sorted units to the tracking module
  Compute distance vectors between each of the input units and the stored profiles associated with c
  Using the distance vectors computed, build a membership matrix between the input units and the stored profiles on c using the tracking classifier
  Using back-tracking, find the best assignment that maximizes the number of assigned units to existing profiles
  Create new profiles for unassigned units
end for

2.6 Results

2.6.1 Data Acquisition and Behavioral Task

Recordings were obtained using a 100-microelectrode silicon array (Blackrock Microsystems, Inc., Salt Lake City, UT) chronically implanted in the primary motor cortex (M1) of a Rhesus Macaque. The implant was performed three months before the recording of the first dataset used in this study. The surgical and behavioral procedures involved in this study were approved by the University of Chicago Institutional Animal Care and Use Committee and

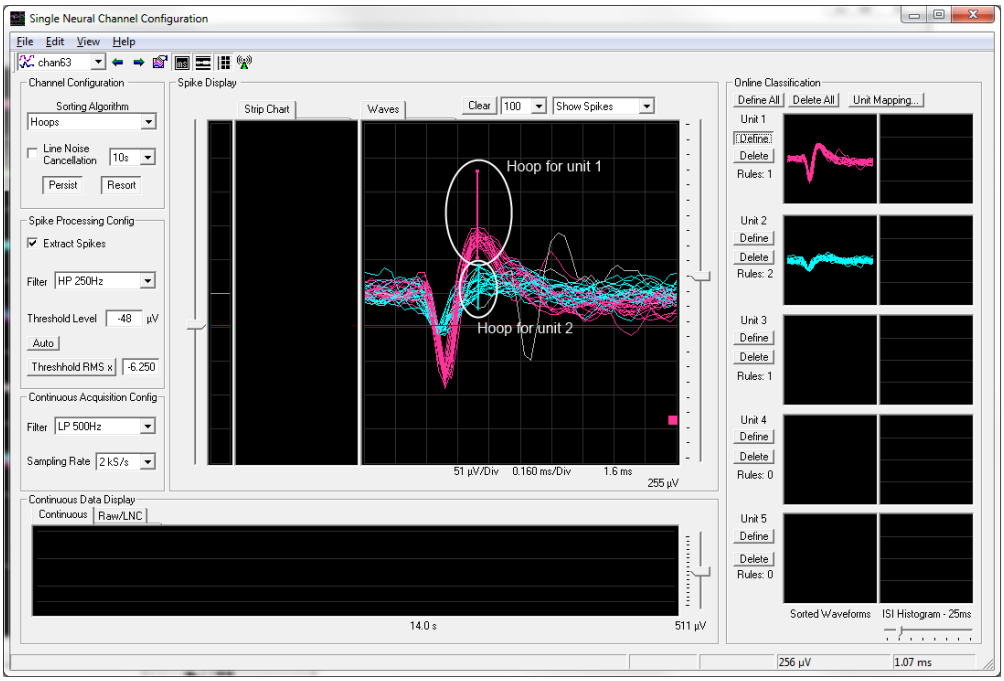

Figure 2.4: Spike sorting on the Cerebus® Online Sorter. Each single-unit is defined by a set of "hoops" where any waveform intersecting with all hoops of a given unit is added to that unit.

conform to the principles outlined in the Guide for the Care and Use of Laboratory Animals. Spike sorting was done using the Cerebus® Online Sorter and the sorting templates were updated on a daily basis. The *Hoops* method was used to perform the spike sorting where a set of bars is defined for each single-unit and when a waveform intersects with all the bars of one set, the waveform is classified as part of the unit corresponding to this set. Figure 2.4 displays a screen shot of the process of defining hoops for single-units.

The subject performed a brain-controlled reach and grasp task. Details of the behavioral task can be found in [2]. Fifteen datasets, spanning a period of 26 days, were used in training (7 datasets) and testing (8 datasets) the classifiers. I used only the first 15 minutes of each dataset.This typically corresponds to the period of updating the spike sorting templates where the subject is not engaged in any behavioral task.

2.6.2 Training Features

Figures 2.5 and 2.7 show the distributions of the waveform- and ISIH- based distance measures when applied on the true positives and true negatives training datasets.

The degree of separability between the distributions provided by each of the features I selected varied. While the peak matching distance provided the best separability between matching and non-matching waveforms, the other three waveform-based distance measures did not provide separation between classes (except in regions where it was very clear that there is a significant difference in the amplitude or time between peaks). The inclusion of the latter three features, however, was necessary to account for cases where two waveforms had very similar peak shapes but with significantly different amplitude or time between peaks or temporal dynamics, which occurred frequently in our data (see Figure 2.6).

On the other hand, all the ISIH-based distance measures did not provide the degree of separation between classes provided by the peak matching. However, their inclusion is nessecary to account for cases where the waveform shapes of two unit instances experience variability but their firing properties remain similar (see Figure 2.8).

2.6.3 Classifiers Evaluation

The performance of the classifiers was assessed based on two metrics:

1. Classification Accuracy: The percentage of correct classification made on a day-to-day basis compared to manual tracking. A correct classification happens when a unit on day x is matched with the correct unit on day $x-1$ (as compared to manual tracking).

2. Percentage of Correct Profiles: The percentage of profiles that were correctly tracked across all days of the testing datasets compared to manual tracking. A correct profile

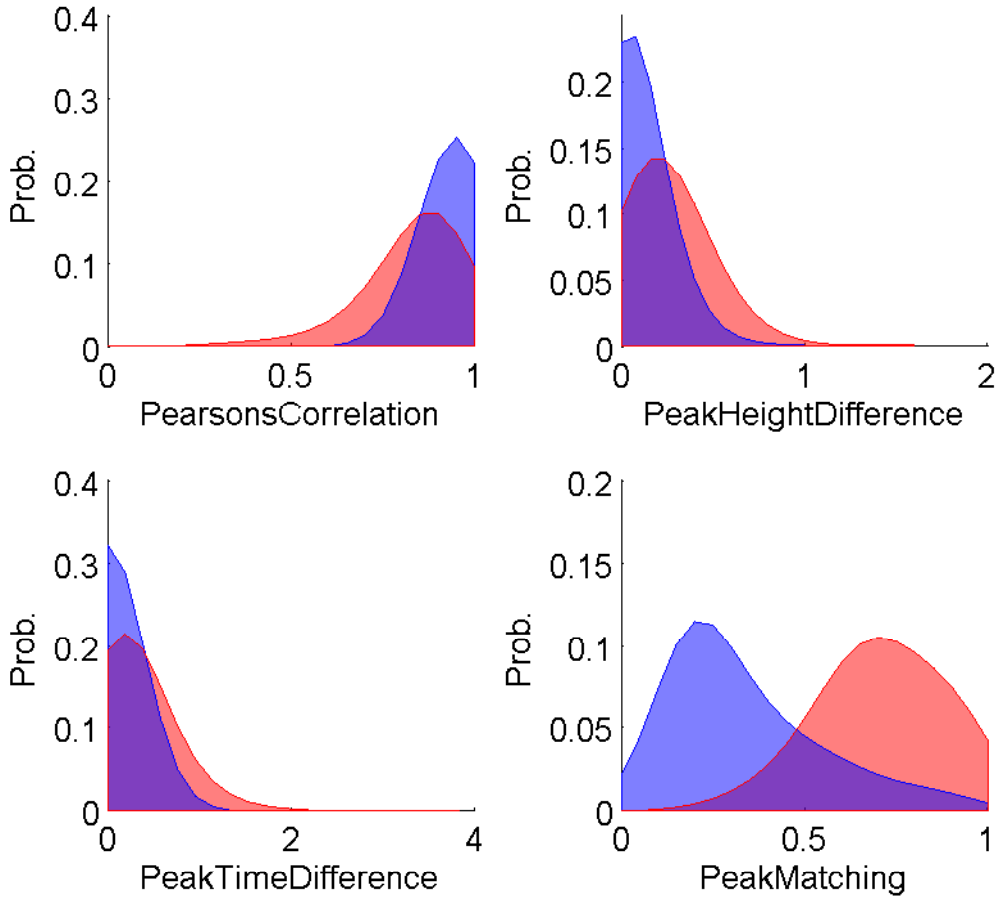

Figure 2.5: Probability distributions of the training true positives (in blue) and true negatives (in red) for each of the four wavefrom-based features.

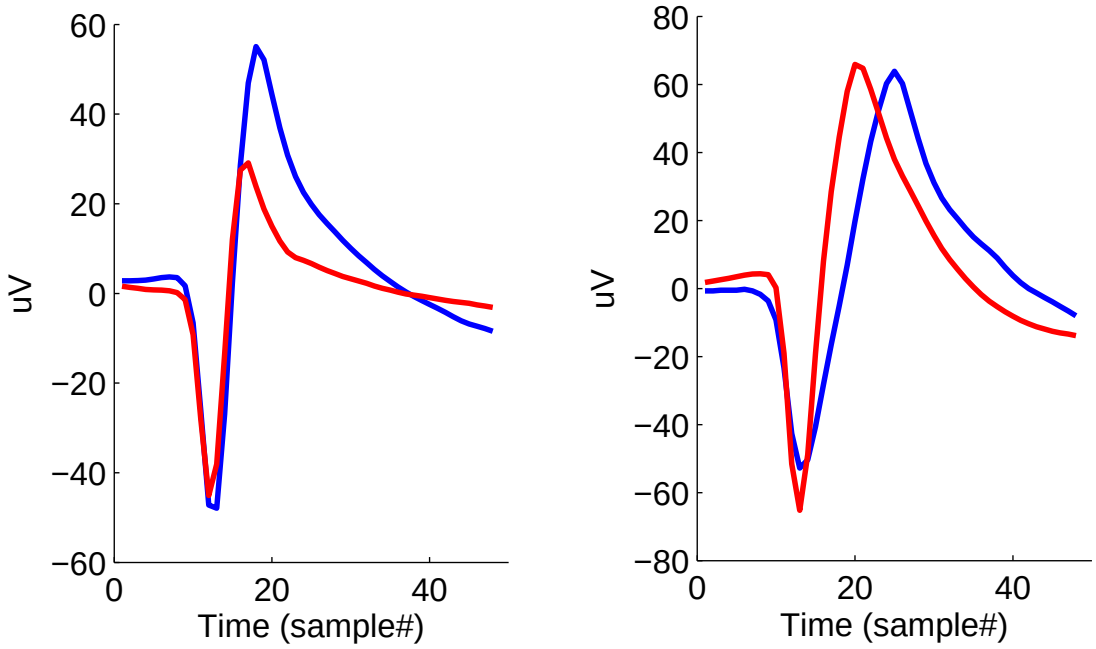

Figure 2.6: Figure a shows an example of two average waveforms that have very similar peak shapes but different peak-to-peak height, while Figure b shows an example of two average waveforms that have very similar peak shapes but different peak-to-peak time.

is the profile that matches a manually tracked profile across all days.

Classifiers implementation was provided by the newFolder liberary [47]. I used a radial basis function as the kernel for the SVM classifier. I compared the performance of the classifiers using four different sets of features:

- **Set 1**: 4 waveform-based features (PC, PH, PT, PM) and 4 ISIH-based features (KLD, BD, KS, EMD).
- **Set 2**: 3 waveform-based features (PH, PT, PM) and 3 ISIH-based features (KLD, BD, KS)
- **Set 3**: 3 ISIH-based features (KLD, BD, KS).
- **Set 4**: 3 waveform-based features (PH, PT, PM).

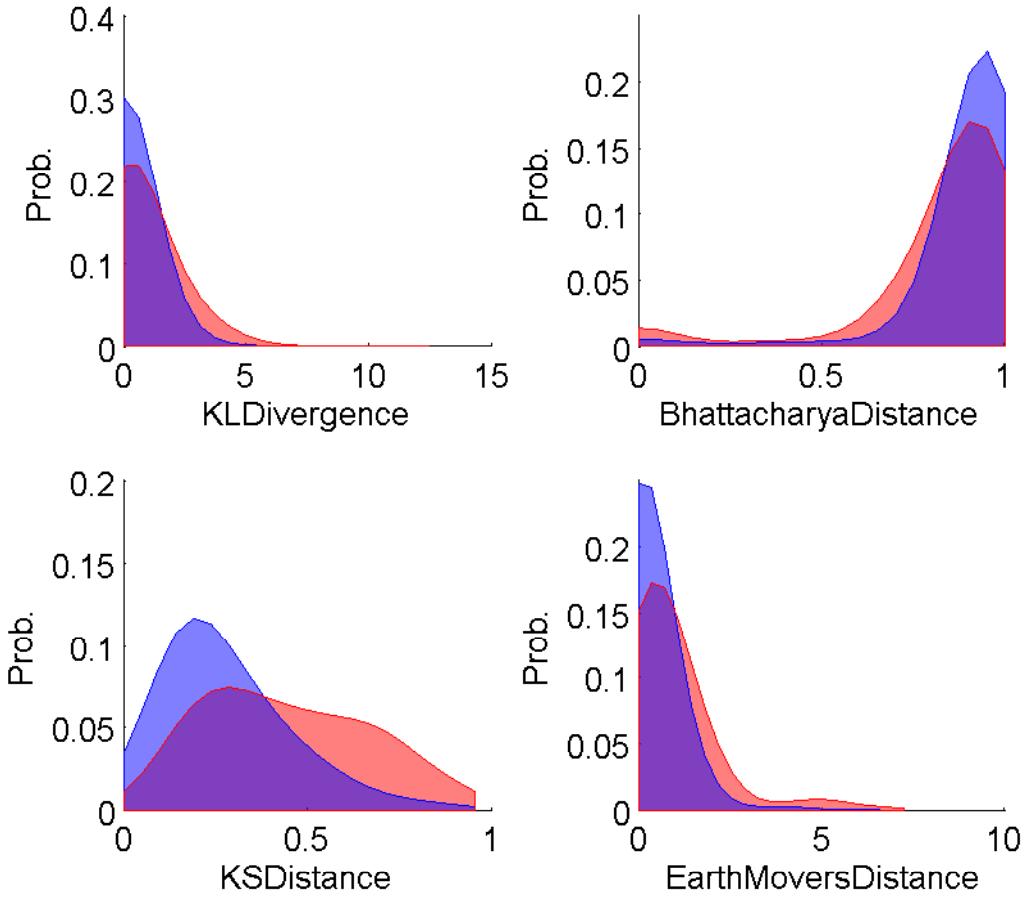

Figure 2.7: Probability distributions of the training true positives (in blue) and true negatives (in red) for each of the four ISIH-based features.

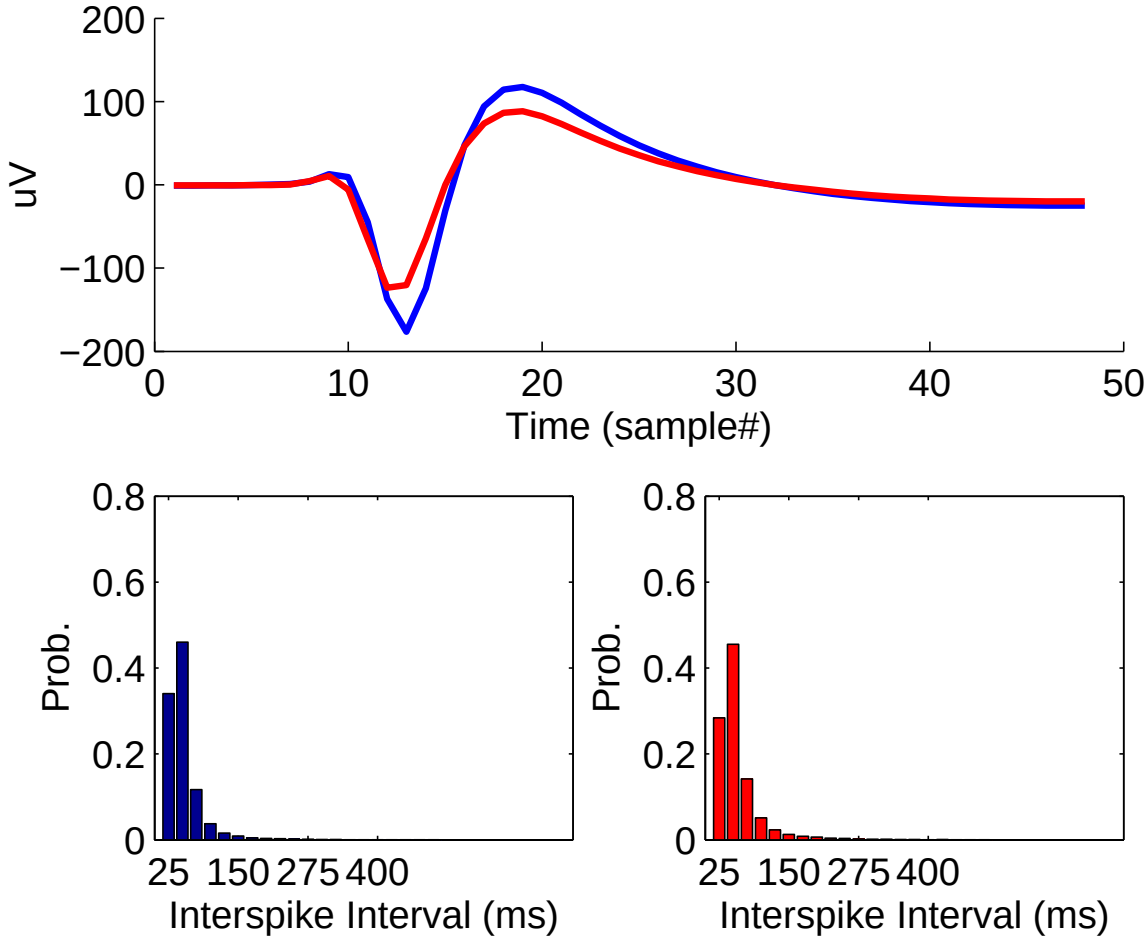

Figure 2.8: The figure shows an example of two instances of a single-unit on two successive days where the average waveform experienced some variability and the ISIH remained the same.

Performing cross validation was challenging because the order of the datasets had to be maintained. This is because changes in the characteristics of a single unit might be small over two successive datasets but much larger across longer duration. We divided the fifteen datasets (explained in Section 2.6.1) into six partitions. Each partition is divided into five training datasets and five testing datasets. The first partition contained datasets 1 to 10, the second partition contained datasets 2 to 11, and so on.

Figure 2.9 shows the classifiers performance under the four feature sets. It is clear that feature sets 1, 2 and 4 give better performance than feature set 3. This shows that adding the ISIH-based features can help improve the performance but by themselves they cannot successfully track single-units. This can be explained by the fact that the subject are not overtrained and the single-units' firing characteristics experience variability during the behavioral task learning phase [21]. Furthermore, I performed t-test to see if any of the classifiers is better than others. For feature sets 1, 2 and 4 all classifiers performed similarly. However, for feature set 3, the RVM and SVM classifiers were better than the NB and MAP classifiers ($p > 0.01$ - p-value is small because of the small number of samples (6 samples)). Figure 2.10 displays the average execution time per channel for the classifiers under different feature sets. The machine used in running the algorithm had an Intel-i7 quad-core processor (3.40GHz) and 16GB of RAM. Execution times were almost the same for the different classifiers. Feature set 1 has the largest execution time (as we might expected because of the larger number of features used) and feature set 3 has the smallest one.

Furthermore, I compared the receiver operating characteristic (RoC) curves of the four classifiers under the four feature sets. For feature set 1, the RVM performed slightly better than other classifiers. For feature set 2, the NB performed the worst. For feature set 3, the RVM and SVM were superior to the NB and MAP. Finally, all classifiers performed

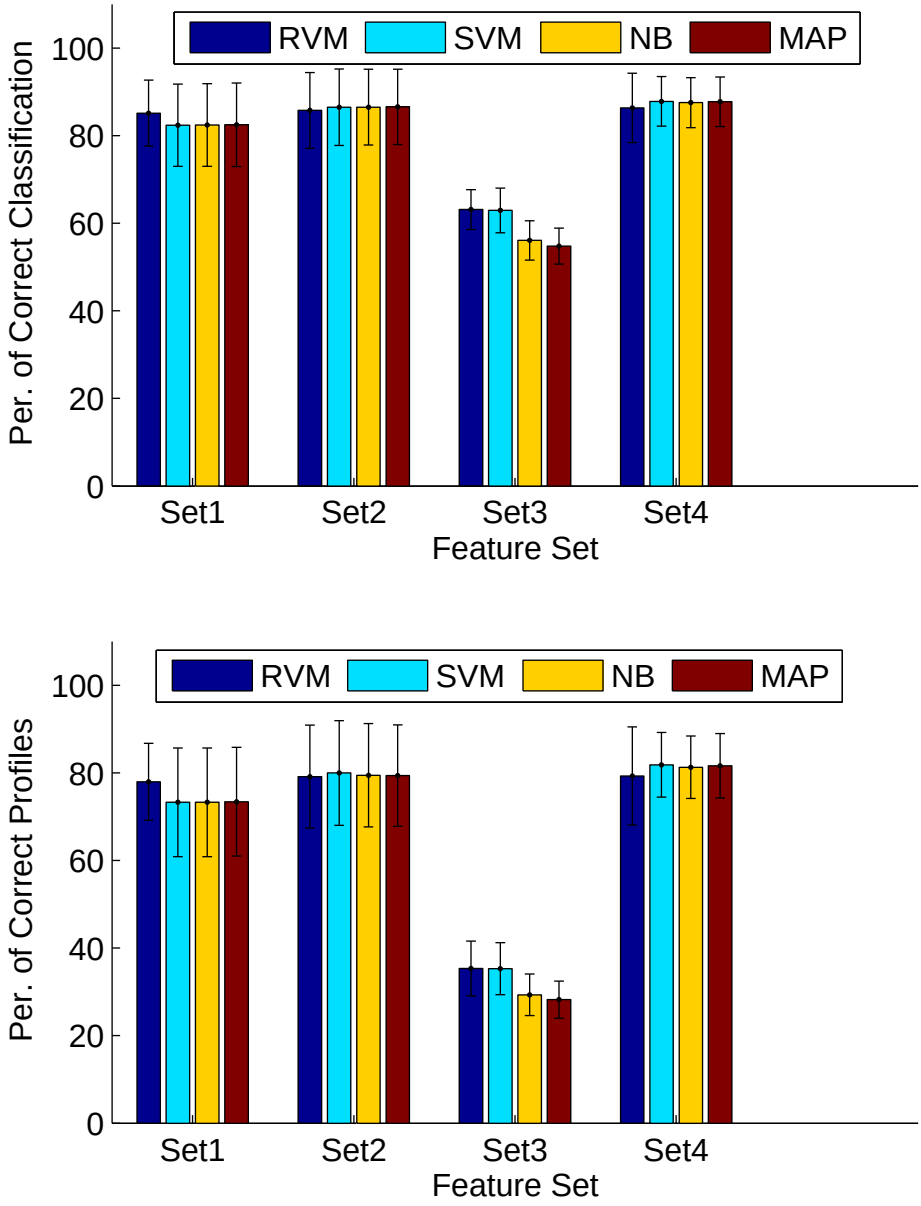

Figure 2.9: Comparison of the classifiers performance using cross validation. Four different sets of features were used. Cross validation was performed using 6 partitions (each had 5 training datasets and 5 testing datasets). The errorbars indicate the standard deviation across the 6 partitions. All classifiers performed similarly for feature sets 1, 2 and 4 (confirmed by t-test).

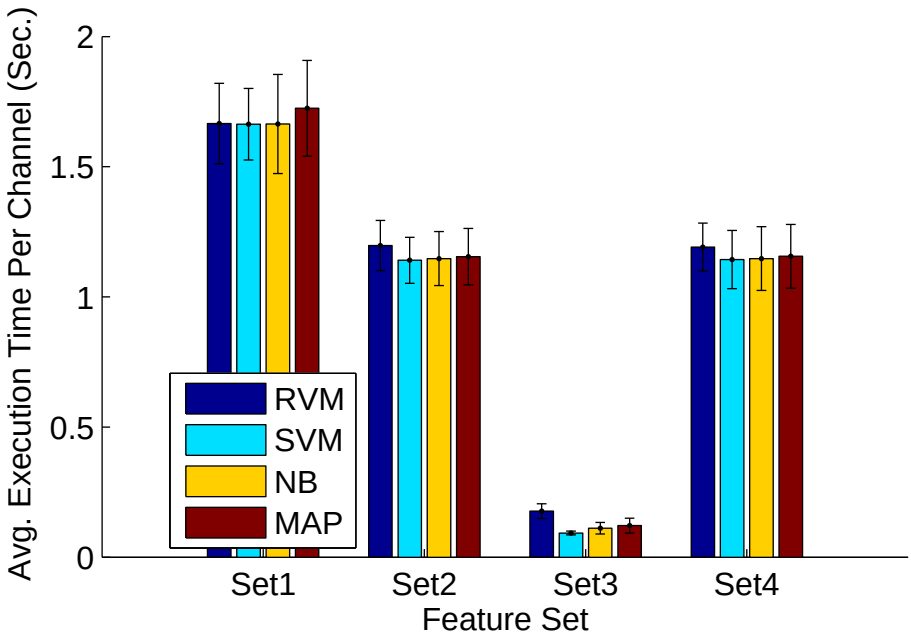

Figure 2.10: Comparison of the classifier performance based on the average execution time per channel. Cross validation was performed using 6 partitions and 4 feature sets were defined. The errorbars indicate the standard deviation across the 6 partitions. All classifiers performed similarly for feature sets 1, 2, and 4 (confirmed by t-test).

similarly under feature set 4. We also plotted the RoC curve of an RVM classifier trainined by randomly shuffling label of training datasets (we call it Rand). We repeated this process for 100 times and plotted the average RoC curve which runs along the chance level showing that my method is unbiased.

Finally, Tables 2.3 and 2.4 show the classifiers' performance measured by the metrics above when trained and tested using all 15 datasets. All classifiers performed very well, however, the RVM was superior to other classifiers in terms of both metrics. We believe that the reason behind this is that it combines the advantages of SVM and those of a probabilistic classifier. I decided to use it to perform further analysis.

Figure 2.12 shows the decision boundary of the RVM classifier under the 4 feature sets. Figure 2.13 displays two of the profiles tracked by the RVM classifier.

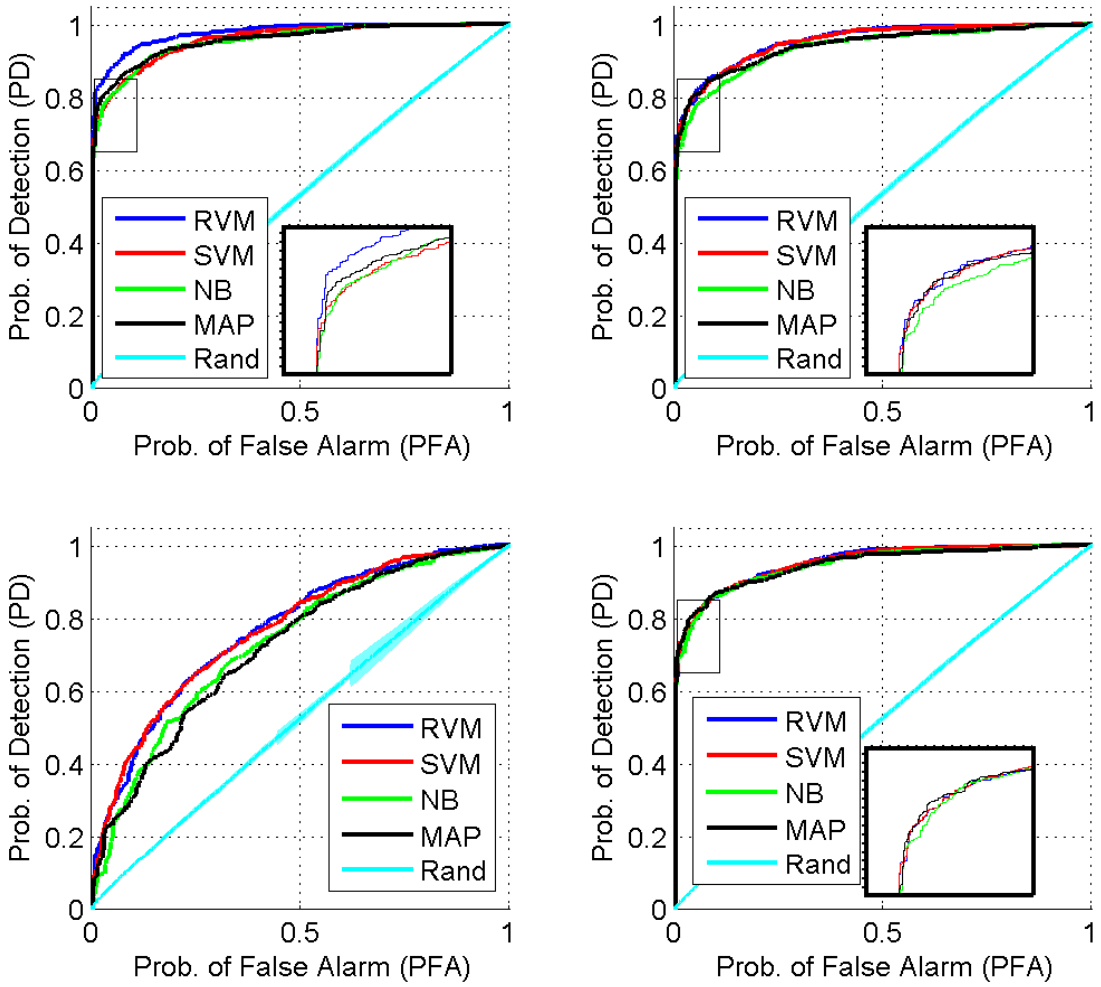

Figure 2.11: RoC curves for different classifiers using 4 different feature sets. The top left figure is for set 1, the top right figure is for set 2, and so on.

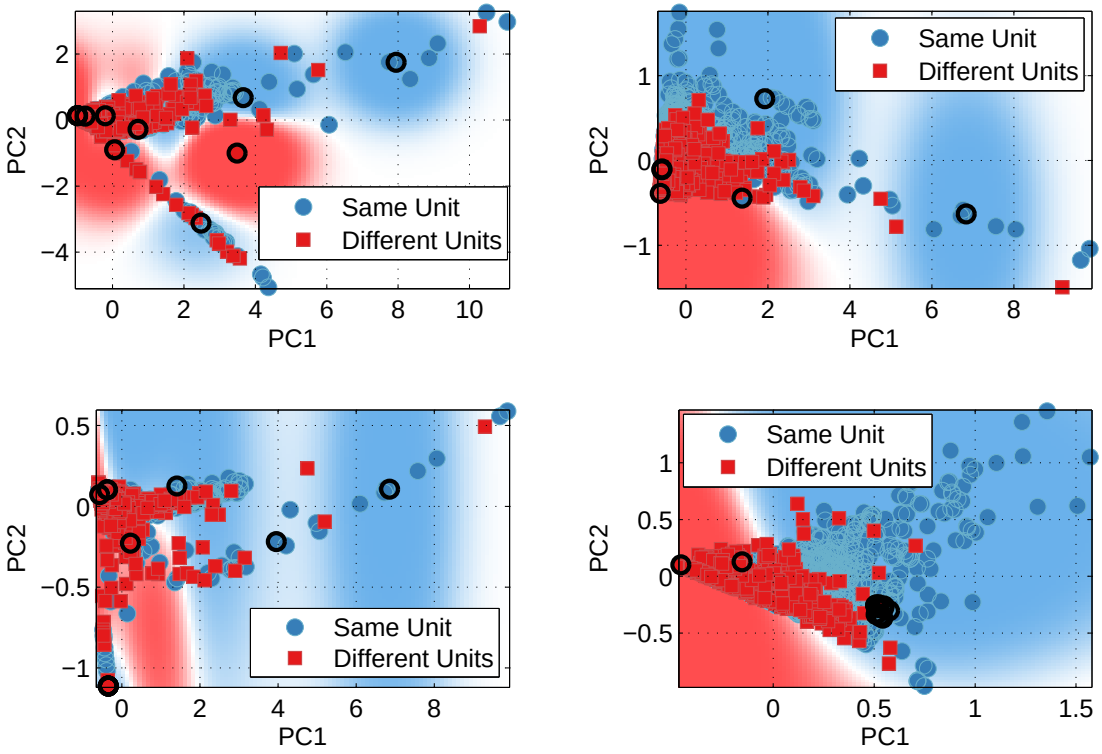

Figure 2.12: Decision boundaries for the RVM classifier using 4 different feature sets For visualization, the RVM classifier was trained on the first two principal components of the training features. The top left figure is for set 1, the top right figure is for set 2, and so on. The black circles are the basis vectors of the RVM classifier.

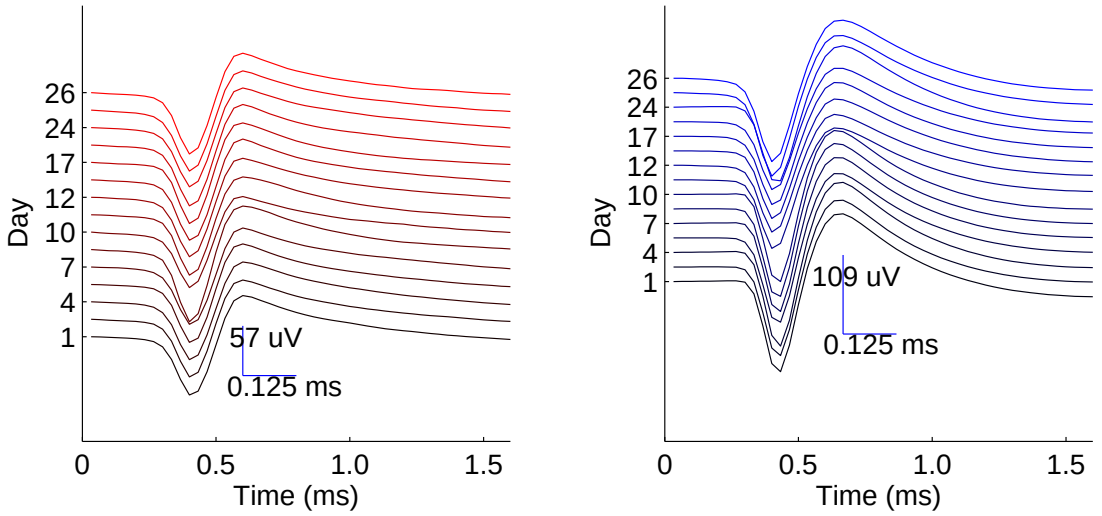

Figure 2.13: Two of the profiles tracked by the RVM classifier across 15 datasets. The profiles were recorded on channels 4 and 90 respectively.

Table 2.3: Classifiers' Performance - Classification Accuracy

Set/Class.	RVM	SVM	NB	MAP
Set 1	90.51%	87.44%	87.30%	86.62%
Set 2	89.14%	88.05%	87.91%	88.39%
Set 3	61.16%	61.92%	40.61%	41.36%
Set 4	89.62%	89.41%	89.62%	90.10%

Table 2.4: Classifiers' Performance - Percentage of Correct Profiles

Set/Class.	RVM	SVM	NB	MAP
Set 1	78.34%	73.62%	74.01%	71.65%
Set 2	74.01%	72.83%	74.01%	72.83%
Set 3	20.47%	15.74%	10.23%	9.84%
Set 4	75.98%	77.16%	76.37%	76.77%

One of the goals of this study is to develop a fast and accurate algorithm to track single-units. A trade-off exists between accuracy and speed and it can be observed clearly when comparing the performance results for feature sets 1 and 4 (See Tables 2.3, 2.4, and 2.5). Feature set 1 achieves a higher performance but at the cost of a longer execution time. On the other hand, feature set 4 has a smaller execution time but at the cost of a lower accuracy. However, the gain in using feature set 4 is greater than that in the case of using feature set 1, since the drop in accuracy is very small (1% drop) compared to a larger increase in the average execution time per channel (17% increase). Therefore, I used feature set 4 to perform any additional analysis.

Table 2.5: Average execution time per channel when the RVM was used to track 15 datasets.

Feature Set	1	2	3	4
Time (sec.)	5.24	4.03	0.62	4.48

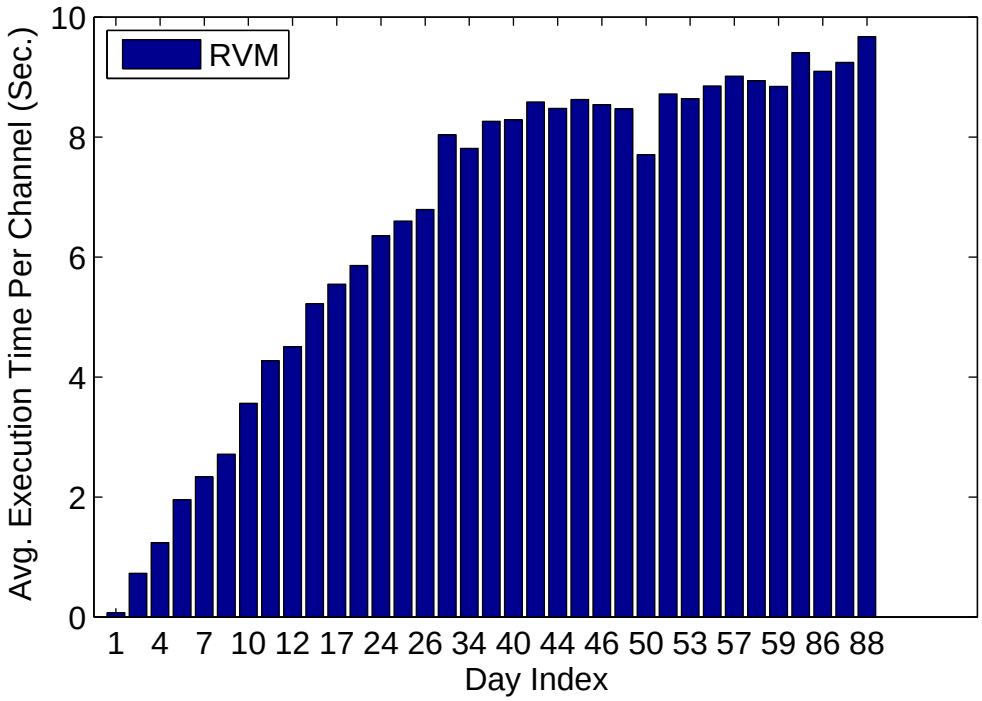

Figure 2.14: Average execution time per channel when feature set 4 and the RVM classifier were used. The classifier was run on 35 datasets spanning 88 days. Convergence in the execution time happens on day 34 of the analysis.

Figure 2.14 shows the convergence of the average execution time per channel as the algorithm is run on more datasets. The convergence happens on day 34 with an average exection time of 8 seconds per channel. This convergence occurrs because the number of active profiles stabilizes after some time. Profiles inactive for a long time (> 7 datasets) are considered droppped and are not included in the tracking.

2.7 Conclusion

In this chapter, I described the details of a single-unit tracking algorithm. I compared the performance of four different classifiers using four different sets of features that were extracted from a unit's average waveforms and firing characteristics. Using all the eight fea-

tures propsed led to the best performance but at the cost of a longer execution time. I found that using three waveform-based features achieved the best trade-off between execution time and accuracy. Using cross validation, the classifiers performed equally. However, when more training datasets were added, the RVM classifier performed slightly better than other classifiers. Finally, when the algorithm was run on 35 datasets spanning 88 days, the execution time converged on day 34 with an average execution time of 8 seconds per channel.

Chapter 3

Unit Selection Algorithm for Decoding

A neural decoding is the process of translating the firing pattern of a fixed subset of the recorded neurons into control signals to actuate an artificial limb. For maximizing success of this control, it is important to maintain a fixed mapping between the neural input space and the kinematic task space, which is known as "*Memory Motor Consolidation*". To automate the process of decoder calibration and to fix this mapping, the accurate identication of neurons to use at the start of each recording session is required, preferably when the subject (non-human primate) is not engaged in any behavioral task.

However, to apply a fixed decoder that only uses a fixed subset of the recorded units, one has to make several decisions, among these: which units to use? how to ensure the stability of these units? what to do when a unit stops showing up on recordings (*dropped*)? In this section, I propose a units selection algorithm that addresses these questions in a systematic way. The algorithms targets the use of a fixed linear decoder [49] following an operant-conditioning approach [24] where only a finite subset of units (cluster) is used for decoding a kinematic variable.

3.1 Unit Selection Criteria

Four criteria for the unit selection algorithm were specified:

1. All units must be stable.

2. Units within a cluster should be functionally connected.

3. Stable units are evenly distributed across clusters.

4. Units recorded on the same channel must belong to the same cluster.

The rationale behind each criterion is explained in the next three sections.

3.2 Tracking Unit Stability

We define a stable single-unit as a unit that was recorded on a daily basis and that maintained similar properties for at least a certain number of days. In [11], the authors found out that once a unit maintains its stability beyond a certain number of days, it is more likely to stay stable. They postulated that this threshold is 7 days. Therefore, using only stable units in decoding may ensure the reliability of decoding and the minimal inconsistency in the subject's experience with the BMI.

I used the algorithm described in Section 2.5 to track units across days. Based on the results of [11], we defined a variable which we called "*Stability Window*", and three categories of units were defined based on it:

- **Stable Units**: The units that were recorded and maintained similar properties on every day throughout the past "*Stability Window*".

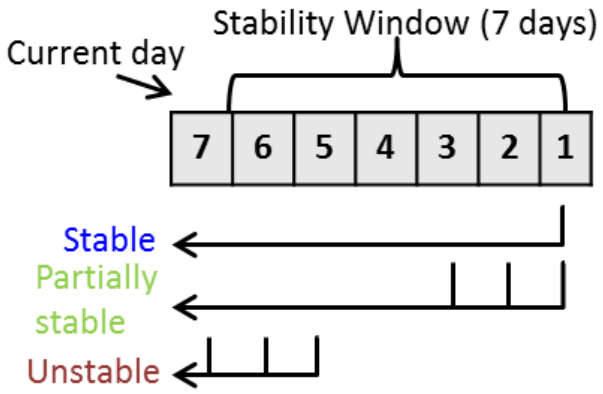

Figure 3.1: Classification of single-units. For a stability window of 7 days, stable units are those units that were identified as similar on a daily basis throughout the stability window. Partially stable units are those units that were identified throughout more that half the stability window. Unstable units are all the remaining units.

- **Partially Stable Units**: The units that were recorded and maintained similar properties for more than half of the past "*Stability Window*".

- **Unstable Units**: All the remaining units.

Figure 3.1 illustrates the process of classifying the single-units based on their stability.

Furthermore, I modified the tracking algorithm such that the comparison between a single-unit and a stored profile is performed by comparing the unit against all the profile instances that occurred within the past "*Stability Window*" of days. The distance vector resulting from this comparison is the distance between the unit and the profile instance **closest** to it (determined using the distance to the decision boundary). The intuition behind this modification is that this way I am more tolerant towards a unit that undergoes some minor instability for a short period of time and then goes back to its stable state. One of the causes of this short term instability might be the high interference from the electric circuitry used in recording on one of the days (which was observed on some channels). This tolerance comes from the fact that now the unit is compared against more than one profile instance. In

addition, this would increase the accuracy of classification since the closest matching profile instance is used in the comparison (instead of the first matching profile instance as was done in Chapter 2). I call this new technique "*Maximum Matching Distance (MMD)*".

3.3 Clusters of Functionally Connected Units

Neuronal assemblies cooperate to perform complex behavioral tasks and this determines the functional connections between neurons. Within an assembly, each neuron assumes a role depending on its tuning function, which depends on the behavioral task being performed at the time of recording [13]. It was found that changes in the tuning function of single-units occur as subjects perform the behavioral task [15]. Correlated neurons are potential cell assemblies that share common coding [22]. In [1], the authors stated that changes in functional connectivity between units and the relevance of the behavioral task are necessary for cortical plasticity to occur. Based on this result, we hypothesized that using already functionally connected units in decoding can help to speed the cortical plasticity occurring from learning the behavioral task.

3.3.1 A Multi-Scale Approach

I used the algorithm in Eldawlatly et al. [13] to infer the functional connectivity between units. The algorithm has a primary parameter and that is the number of time scales in which the correlation is calculated. The average firing rate is calculated at each time scale using wavelets. Pairwise correlations are then calculated between units' spike trains at different time scales resulting in a set of correlation matrices that are fused into one similarity matrix using SVD. Algorithm 2 provides a summary of the algorithm and Appendix B provides

more details on the computations of each step.

Algorithm 2 Functional connectivity algorithm

for $j = 1$ to J (timescale index) **do**
  for $p = 1$ to P (neuron index) **do**
    Obtain the wavelet representation s_p^j of each spike train s_p
  end for
  Compute the $P \times P$ correlation matrix $\Sigma^{(j)}$ of the spike trains at timescale j
end for
Form the block diagonal correlation matrix R from the matrices $\Sigma^{(j)}$ and obtain its spectral decomposition using SVD
Obtain the pairwise similarity of neurons using the weighted sum of the first few dominant modes of R

The result of this algorithm is an augmented correlation matrix (augmented because it is the result of fusing the correlation matrices across different time scales), where each cell in the matrix indicates the strength of functional connectivity between the two units (i.e. the augmented correlation between two units is an indicator of their functional connectivity). Note that from now on when I use the term "**correlation**", I mean the "**augmented correlation**" between the units. An additional clustering step is required to divide the units into different partitions, each controlling a degree of freedom. I used the same clustering algorithm used in [13], and that is the Spectral Clustering [32].

3.4 Balancing Units

The clusters produced from the previous step are generally unbalanced in terms of the number of units. This is undesirable because one cluster might contain a large number of units while another cluster contains a much smaller number of units. In the event of a unit drop from the smaller cluster, the subject may be unable to control the degree of freedom associated with that cluster. I developed an algorithm to balance the number of units across clusters

while maintaining the functional connectivity between units within a cluster. The aim of this algorithm is to provide the subject with the same degree of control for each degree of freedom by assigning equal number of units across clusters. This way we also divide the labor equally over units so that the control would not deteriorate in case of a unit drop.

3.4.1 Measuring the functional connectivity between units

Swapping units between clusters during the balancing process requires quantifying the connectivity of each unit to the rest of the cluster units. The **Interquartile Mean** (IQM) of the correlations between a unit and the rest of the cluster units can be used for this purpose. The IQM can be calculated by throwing away the first and last quartiles of the correlations and averaging the rest [42]. Assume that ρ is the sorted array of the correlations between a unit and the rest of cluster units and it is of length n, the IQM is:

$$IQM(\rho) = \frac{2}{n} \sum_{i=\frac{n}{4}+1}^{\frac{3n}{4}} \rho_i \tag{3.1}$$

The IQM combines the benefits of both the **average** and the **median**. It handles the effect of the outliers by computing the first and third quartiles and at the same time the result of the metric does not have to be a value in the series.

3.4.2 Balancing Algorithm

The balancing algorithm is summarized in Algorithm 3. An extra criteria is added to this algorithm where units recorded on the same channel are assigned to the same cluster. This criteria is specified because units recorded on the same electrode are located in close proximity

and it was found that neighboring neurons tend to share common inputs of the same sign [48]. This enables them to work together as a coherent functional group when their common input is activated. The modified algorithm is summarized in Algorithm 4.

Algorithm 3 Balancing Algorithm

Define a minimum number of units for each cluster
Run the clustering algorithm described in 3.3.1 and get the functional clusters
Sort the units in the clusters based on a functional connectivity metric
Determine clusters that have excess units (P)
Determine clusters that are in short of units (R)
while R is not empty && P is not empty **do**
  for each cluster C in set P **do**
    Find the connectivity of the least connected unit to the rest of the cluster units $MinCon_C$
  end for
  Select the cluster C from the set P with the minimum $MinCon_C$ (i.e. the cluster that has the least connected unit)
  Declare the least connected unit in C a free unit
  for each cluster C in set R **do**
    Find the connectivity of the free unit to its units $FreeUnitCon_C$
  end for
  Add the free unit to the cluster C with maximum $FreeUnitCon_C$ (i.e. the cluster that the free unit is most connected to)
  Update stats of sets P and R
end while
if there are clusters with excess number of units above the minimum number specified **then**
  Distribute them using the same method used above
end if

Algorithm 4 Modified Balancing Algorithm - Units on the same channel belong to the same cluster

Set a minimum number of units for each cluster
Run the clustering algorithm described in 3.3.1 and get the functional clusters
for each channel C **do**
  if units recorded on C do not belong to the same cluster **then**
    Find the average connectivity of each cluster to each of the units recorded on C
    Move all units recorded on C to the cluster they are most connected to
  end if
end for
Sort channels in the clusters based on average connectivity of each channel units to the rest of the cluster units
Determine clusters that have excess units (P)
Determine clusters that are in short of units (R)
while R is not empty && P is not empty **do**
  for each cluster C in set P **do**
    Find the connectivity of the least connected channel to the rest of the cluster units (average connectivity of channel units) $MinCon_C$
  end for
  Select the cluster C from the set P with the minimum $MinCon_C$ (i.e. the cluster that has the least connected channel)
  Declare the units of the least connected channel in C free units
  for each cluster C in set R **do**
    Find the average connectivity of the free units to its units $FreeUnitCon_C$
  end for
  Add the free units to the cluster C with maximum $FreeUnitCon_C$ (i.e. the cluster that the free units are most connected to)
  Update stats of sets P and R
end while
if there are clusters with excess number of units above the minimum number specified **then**
  Distribute them using the same method used above
end if

3.5 Unit Selection Algorithm

Figure 3.2 illustrates all the steps of the unit selection algorithm. For each channel, spike sorting is performed on the recorded signal. The spike sorted units together with the stored units profiles previously recorded on this channel are then input to the tracking module which finds matches between the units profiles and the spike sorted units. The tracking module starts its operation by extracting features from instances of single-units (spike sorted units and units profiles). It then finds pairwise distances between the features of each spike sorted single-unit and the instances of each stored profiles. After finding the best assignment, the tracking module outputs the updated profiles recorded on the channel. Those updated profiles are then input to a stability units filter, that extracts only stable units according to the criteria specified in Section 3.2. Stable units from all channels are then lumped together and input to the functional connectivity module which calculates a similarity correlation matrix using the stable single-units spike trains. A spectral clustering algorithm is then applied to the similarity matrix where the number of clusters is the number of degrees of freedom (DOFs) the subject needs to control. The resulting clusters will vary in the number of single-untis belonging to them. Therefore, a balancing algorithm is applied to the clusters to: 1) balance the number of units across all clusters, 2) maintain the functional connectivity relations between units within a cluster. The clusters output from the balancing algorithm are used to control different DOFs through the decoder.

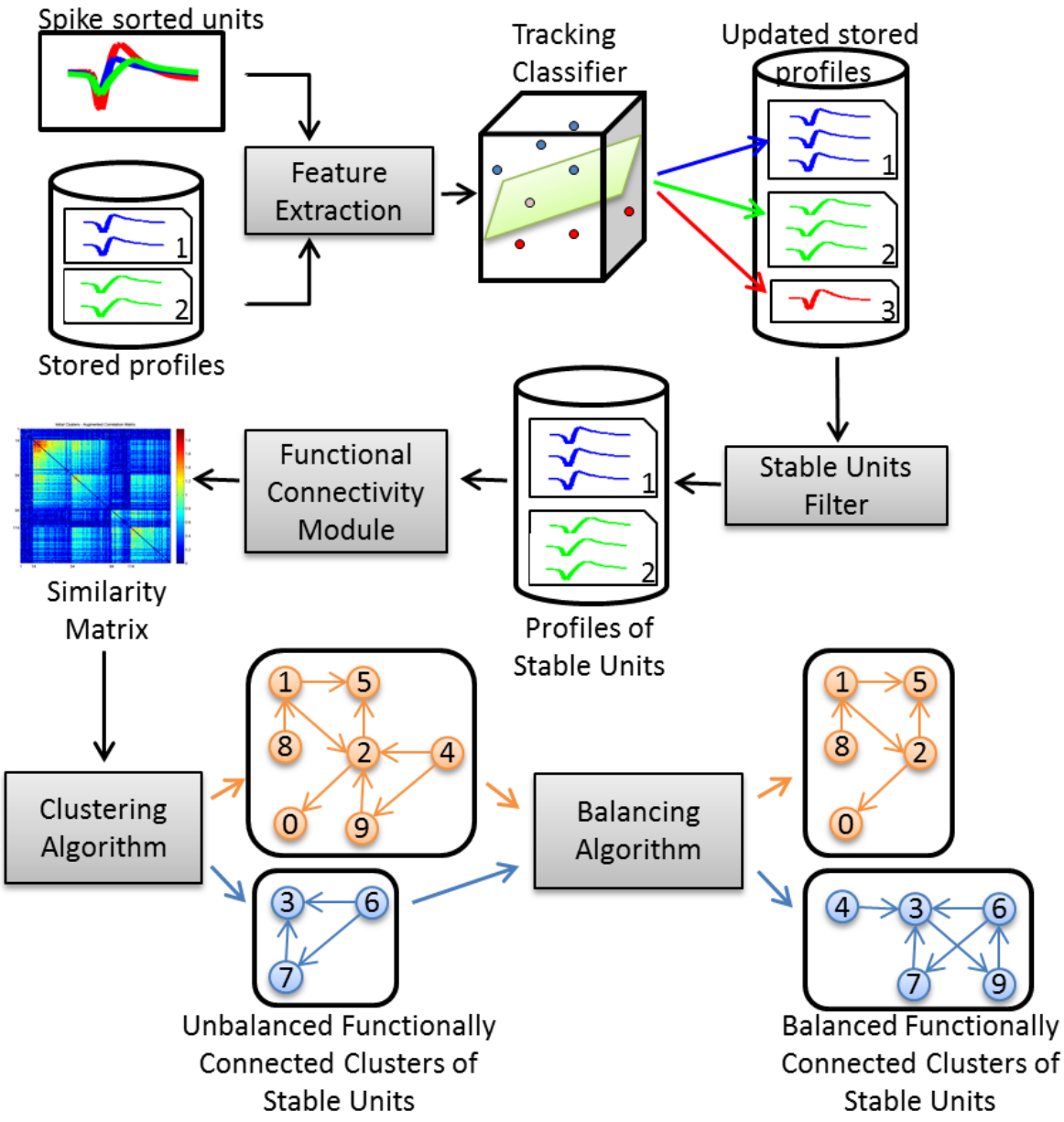

Figure 3.2: Unit selection algorithm: On each channel, spike sorted units are compared against stored profiles on the same channel. The tracking classifier determines whether to add a single-unit to an existing profile or to create a new profile for it. The updated units profiles are input to a stability filter that classifies units into stable, partially stable and unstable units. Functional connectivity matrix is calculated for stable units. The matrix is then input to a clustering algorithm that outputs a set of functionally connected clusters of single-units. However, these clusters are unbalanced in terms of the number of units. Therefore, the clusters pass through a balancing algorithm that maintains the functional connection relations between units while balancing the number of units across clusters. Each cluster is used to control a kinematic variable.

3.6 Unit Replacement Strategies

Even if I only used stable units, units will continue to drop over time [25]. Among the reasons of this drop is the formation of a glial scar consisting of reactive astrocytes and microglia around the electrodes site which affects the quality of recording [17]. The decoder should be prepared with a strategy to replace the dropped unit with another unit that would best match the dropped one to get the closest decoding performance and to make the replacement unnoticeable to the subject as much as possible One way of measuring this is by observing how much the behavioral performance dropped after the replacement.

In this section, I propose two replacement strategies,

- Strategy 1 is based on the relationship between the replacement unit with the rest of the cluster units,
- Strategy 2 is based on the relationship between the dropped unit and its replacement

Furthermore, a small modification is made to the units selection algorithm where the units in the free pool (not assigned to any cluster) are divided evenly on the clusters using the same balancing algorithm described in Section 3.4.2. These units are regarded as substitute units for each cluster (i.e. each cluster has its own free pool of substitute units).

3.6.1 Replacement Strategy 1: Unit Closest to Cluster

This replacement strategy looks for the unit from the free units pool that has the highest correlation (using the metric described in Section 3.4.1) with the rest of the cluster units. The intuition behind this strategy is the same as the one behind choosing the functionally connected units for decoding. We postulate that this would make it easier for the subject

to control the robotic arm since we pick the unit that may constitute a member of the same cell assembly as the rest of the units.

3.6.2 Replacement Strategy 2: Unit Closest to Dropped Unit

This replacement strategy finds the unit that has the highest correlation to the dropped unit. The intuition behind this strategy is to find the unit that would act the same as the dropped unit. In principle, the decoder weights corresponding to the dropped unit should be suitable for the new unit and the replacement process would be transparent to the subject. I propose two methods to apply this strategy:

- By using the correlation matrix (computed in Section 3.3.1) directly.
- By applying graph-theory algorithms on the correlation matrix (computed in Section 3.3.1).

The second method models the cluster as an undirectional weighted graph where the graph nodes are the cluster units and the weights of edges between nodes are the distances between nodes. The distance measure between two units x and y is derived from the pairwise correlation $\rho(x, y)$ such that [5]:

$$D(x, y) = \sqrt{2 * (1 - \rho(x, y)} \tag{3.2}$$

Given an adjacency correlation matrix, the model results in a fully connected graph where each node is connected to all other nodes. The weight on a given edge is the distance between the two nodes connecting the edge (distance is obtained from correlation as in equation 3.2). From the fully connected graph, the minimum spanning tree (MST) is constructed using

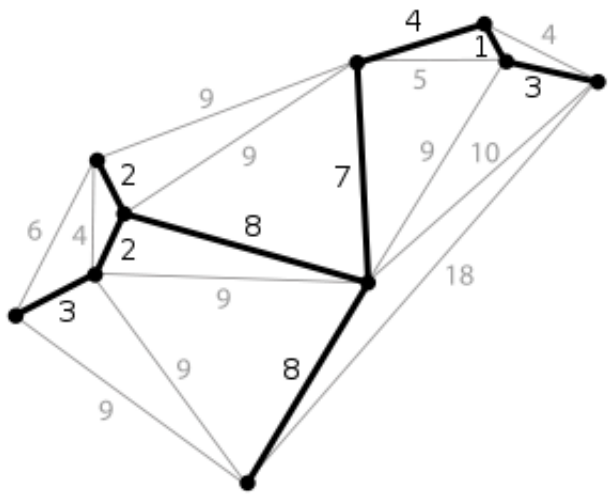

Figure 3.3: An example for an MST constructed from a graph containing 10 nodes. An MST can be constructed using either Prim's or Kruskal's algorithms. Both algorithms are greedy in the way they choose the MST edges. In both algorithms, edges are sorted ascendingly based on their weights and the lowest weight edges are added one by one as long as a loop is not formed and until all nodes are added to the tree. The boldface edges in the figure are the ones that were included in the MST. Total cost of the MST is 31. Taken from [50].

any of the well-known MST algorithms (Prim's [38] or Kruskal's [26] algorithms). A tree is an acyclic graph and a spanning tree is a subgraph that contains all of that graph's vertices and is a single tree [43]. Finally, an MST is is a spanning tree whose weight (the sum of the weights of its edges) is no larger than the weight of any other spanning tree [43]. Figure 3.3 shows an example of an MST for a 10-node graph.

The MST in our case is just a representation of the structure of the underlying network formed by the cluster nodes. It reveals the underlying network that connects all the nodes of the cluster and includes only the essential strong connections (edges with high correlation) to connect all the nodes.

My algorithm forms the MST of the cluster units and the free units using the popular Kruskal algorithm [26], and from the MST it computes all pairs shortest paths using Floyd-Warshall algorithm [28]. The free node that is closest to the dropped one is selected as the

replacement unit. This is different from merely selecting the unit that is most correlated to the dropped unit because only a small subset of the edges are included in the MST (most of the connections are considered redundant and not included in the formed tree).

3.7 Results

3.7.1 Tracking Single-Units Stability

I repeated the same analysis I did in Section 2.6.3 after applying the modification described in Section 3.2 to the tracking algorithm to verify that the modification resulted in an improved performance. I chose the stability window to be 7 days based on the results in [11].

Figures 3.4 and 3.5 compare the performance of the four classifiers using cross validation. The results were very similar to the results I obtained in Section 2.6.3.

Tables 3.1 and 3.2 compare the classifiers' performance when all the 15 datasets were used in training and testing. Again, the RVM classifier was slightly superior to other classifiers in terms of both metrics, that is why I chose to use it for the rest of the analysis. Furthermore, feature set 4 gave the best trade-off between accuracy and speed (similar to what I obtained in Section 2.6.3).

Table 3.1: Classifiers' Performance - Classification Accuracy

Set/Class.	RVM	SVM	NB	MAP
Set 1	91.67%	88.05%	88.12%	87.44%
Set 2	89.89%	88.32%	89.48%	85.87%
Set 3	62.11%	67.84%	57.26%	57.13%
Set 4	90.30%	89.55%	90.03%	89.89%

Figure 3.6 shows the same profiles I tracked in Section 2.6.3. Both methods obtained the

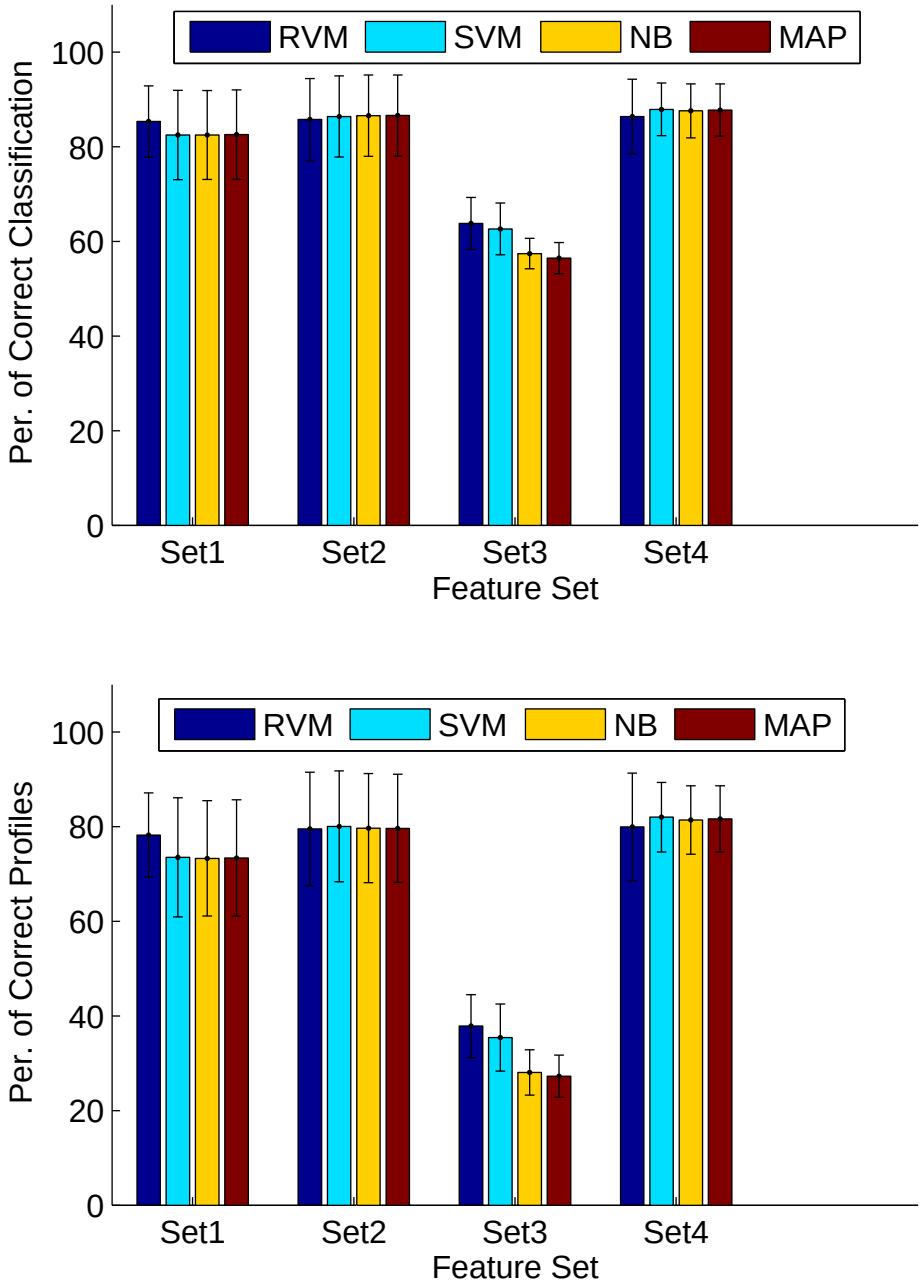

Figure 3.4: Comparison of the classifiers performance using cross validation. Four different sets of features were used. Cross validation was performed using 6 partitions (each had 5 training datasets and 5 testing datasets). The errorbars indicate the standard deviation across the 6 partitions. All classifiers performed similarly for feature sets 1, 2 and 4 (confirmed by t-test).

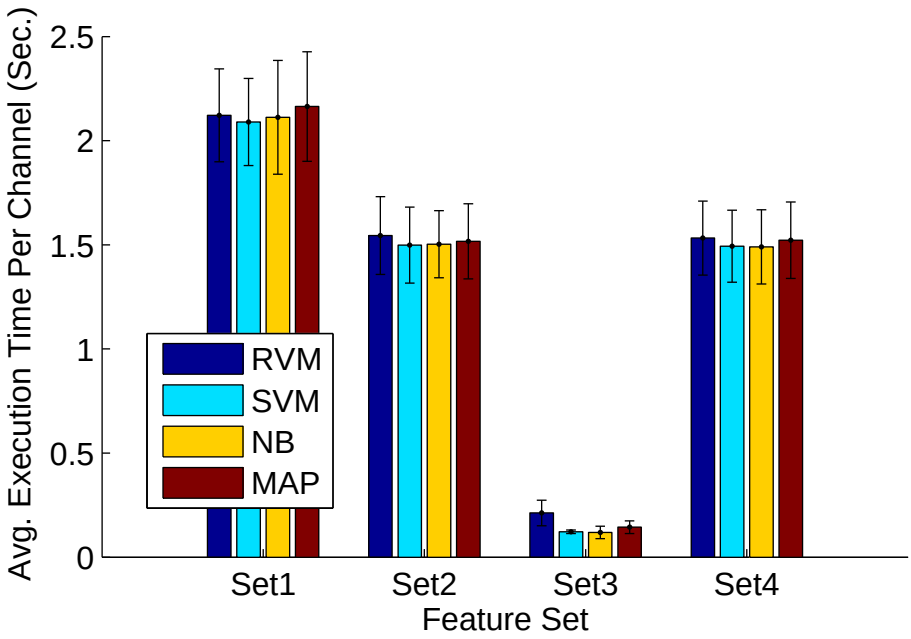

Figure 3.5: Comparison of the classifier performance based on the average execution time per channel. Cross validation was performed using 6 partitions and 4 feature sets were defined. and the errorbars indicate the standard deviation across the 6 partitions. All classifiers performed similarly for feature sets 1, 2, and 4 (confirmed by t-test).

same results. Finally, Figure 3.7 shows the average execution time per channel when the algorithm is run on 35 datasets spanning a period of 88 days. The execution time converges on day 34 of the analysis.

Comparing the results I obtained using FMD and MMD, I found the following:

- MMD has a higer average execution time per channel and that results from the additional number of comparisons the algorithm has to perform to find the closest matching unit within the "*Stability Window*" history.

- MMD has a higer accuracy than FMD and that is because the technique aims at finding the closest match to the new unit.

I decided to use MMD for the rest of the analysis.

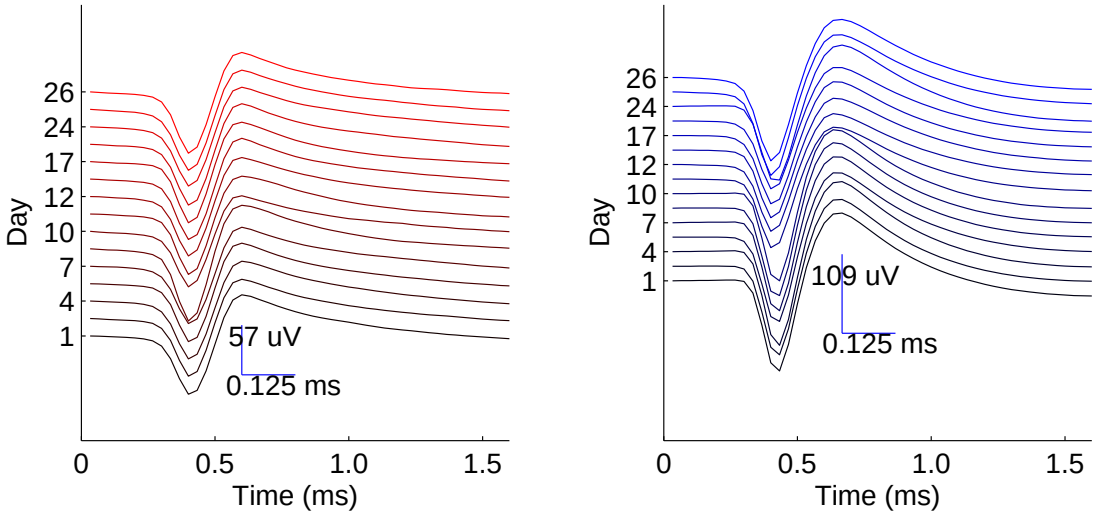

Figure 3.6: Two of the profiles tracked by the RVM classifier across 15 datasets. The profiles were recorded on channels 4 and 90 respectively.

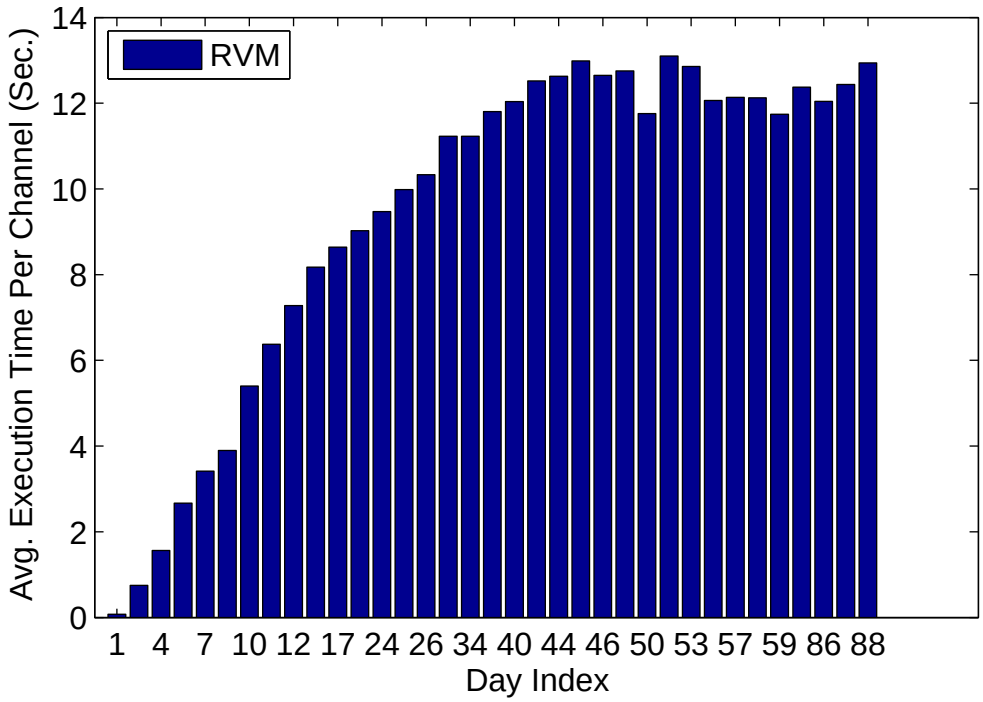

Figure 3.7: Average execution time per channel when feature set 4 and the RVM classifier were used. The classifier was run on 35 datasets spanning 88 days. Convergence in the execution time happens on day 34 of the analysis.

Table 3.2: Classifiers' Performance - Percentage of Correct Profiles

Set/Class.	RVM	SVM	NB	MAP
Set 1	82.67%	74.80%	75.19%	73.22%
Set 2	75.98%	74.80%	66.53%	65.35%
Set 3	26.37%	29.92%	16.14%	15.74%
Set 4	77.16%	75.19%	75.98%	75.19%

3.7.2 Stability Analysis

Next, I investigated the stability of the units for longer durations. I ran the tracking algorithm on 55 datasets spanning the duration of 5 months. Figure 3.8 shows different statistics about the neural population on any given day of the analysis. The number of units was almost constant across days and it shows that we still have not reached the point where the array becomes incapable of detecting spikes on a significant number of channels [25]. The duration after which the spiking activity fades away varries from one subject to another, and the reason behind this fading away is believed to be the formation of a glial scar consisting of reactive astrocytes and microglia around the electrodes site which affects the quality of recording [17]. The slow decay of the curve of the number of units that were recorded on day 1 and recorded on any given day (the blue line) confirms the results in [11] which stated that once a unit becomes stable it is more likely to remain stable.

I also plotted the number of units of each of the categories defined in Section 3.2 on any given day in Figure 3.9. As what we expected from observing Figure 3.8, the stable units form the majority of the population. Fluctuations in the total number of recorded units are reflected on the number of stable and unstable units. An increase in the number of units is reflected as an increase in the number of unstable units and a decrease in the number of units is reflected as a decrease in the number of stable units. Furthermore, Figures 3.10,

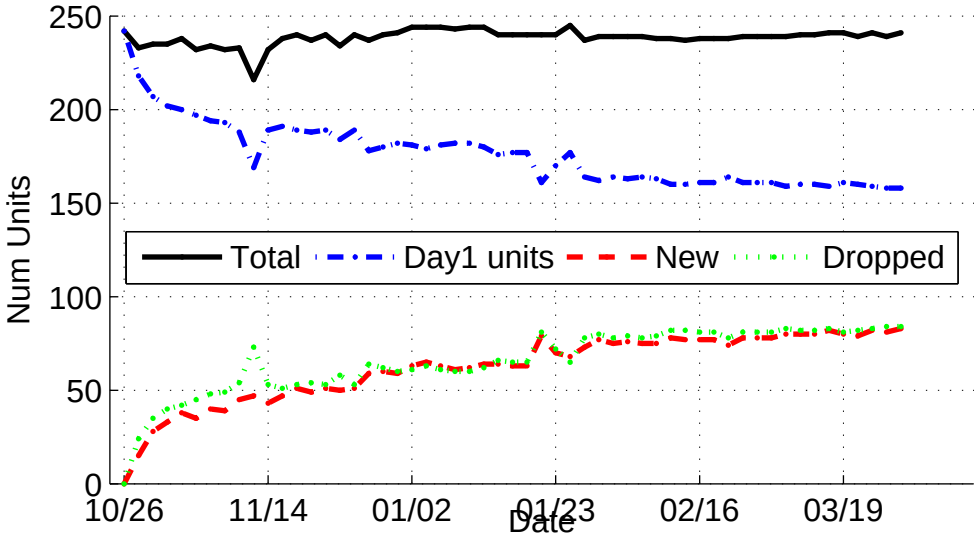

Figure 3.8: For a given day x: the black line shows total number of units recorded on day x, the blue and green lines indicate the number units that were recorded on day 1 and were also recorded or dropped respectively on day x, and the red line shows the number of new units recorded on day x relative to day 1.

3.11, and 3.12 display different samples of the profiles tracked.

Finally, I conclude the units tracking analysis by plotting a histogram of the number of contiguous days a unit is recorded on until it drops. The histogram is displayed in Figure 3.13. The mass of the histogram is mainly divided in two regions:

- Days < 20
- Days > 140

Again, my results proves that once a unit becomes stable, it is more likely to remain stable for a long time.

3.7.3 Unit Selection

We used the algorithm to select the units of five clusters for one subject. We assigned 20 units for each cluster. Figure 3.14 shows the units correlation matrix before and after

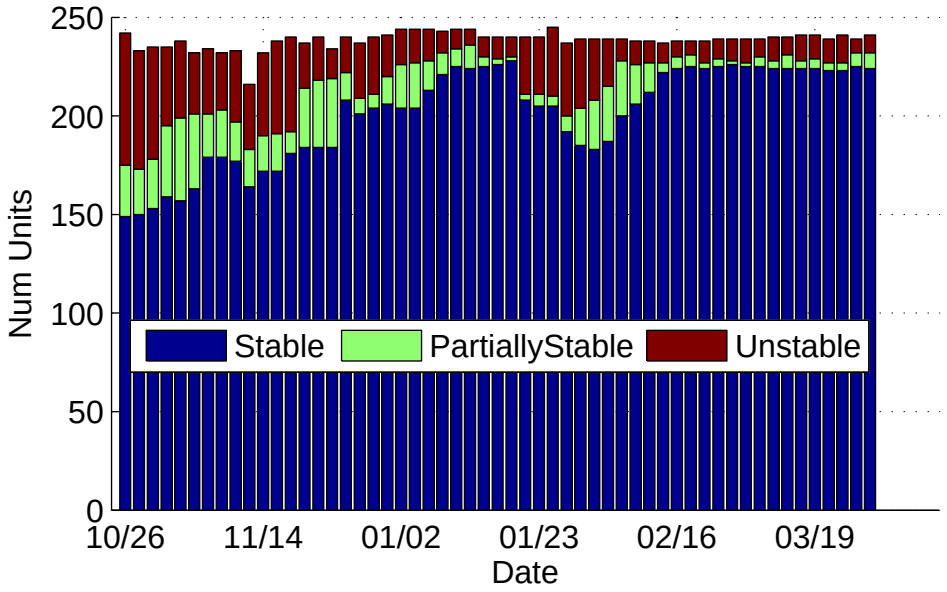

Figure 3.9: Number of stable, partially stable, and unstable units on any given day. An increase in the number of units is reflected as an increase in the number of unstable units and a decrease in the number of units is reflected as a decrease in the number of stable units.

applying the balancing algorithm. The figure shows that the correlation within a cluster is higher than that between clusters. The figure also shows that the balancing algorithm only swaps the weakly connected units between clusters and therefore, it maintains the same clusters skeleton resulting from the functional connectivity algorithm. The mapping of the clusters on the physical electrodes is displayed in Figure 3.15. Some of the clusters reflect some degree of localization on the array and this agrees with the fact that neighboring units tend to be functionally connected [48] (because there is a high probability of having common inputs.)

3.7.4 Replacement Strategies

I compared the performance of three replacement strategies using 10 datasets:

1. Finding the most correlated unit to the dropped unit (*UnitFuncCon*).

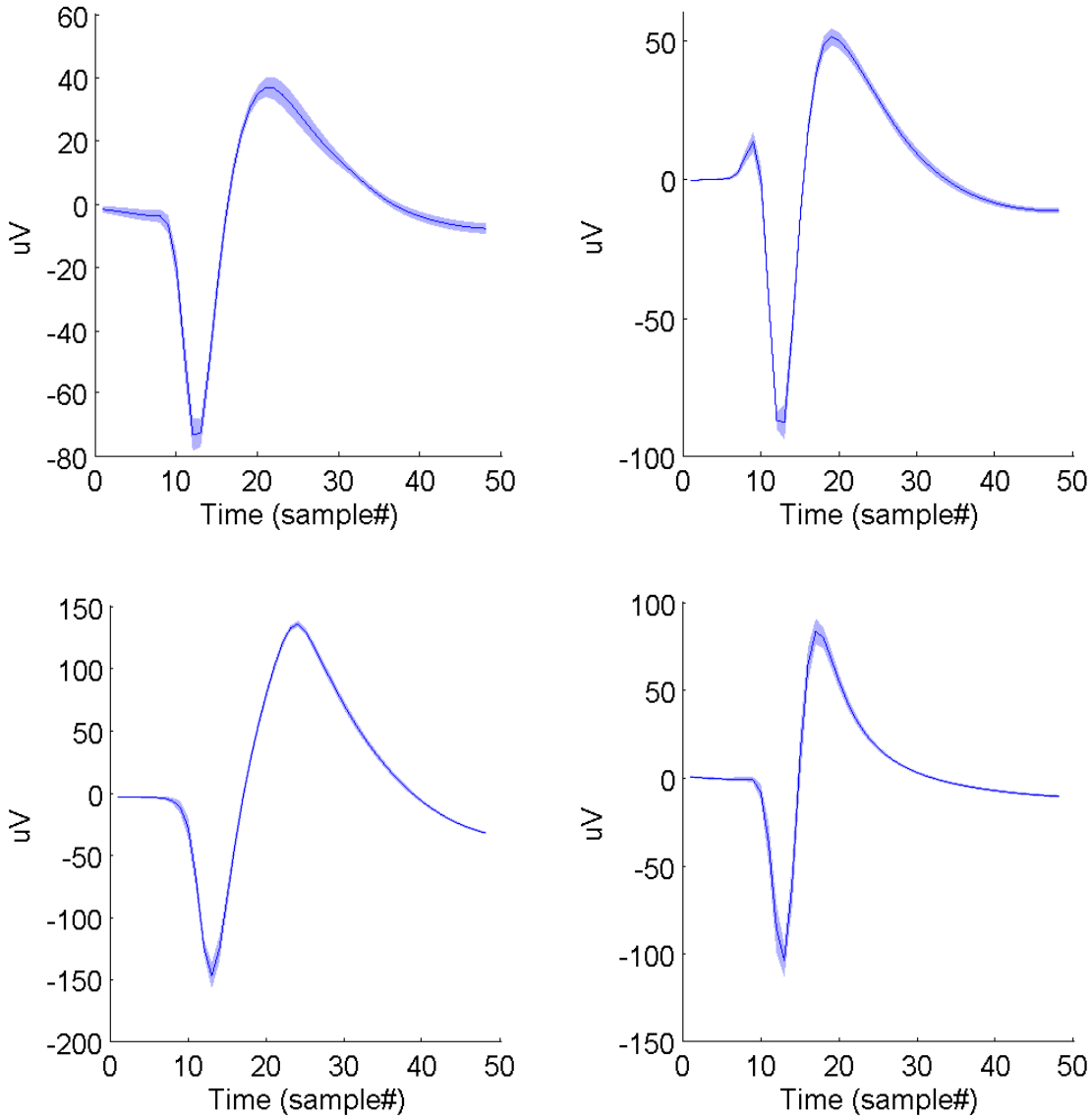

Figure 3.10: The figure shows 4 sample profiles recorded on channels 30, 32, 49, and 50 respectively. The single-units represented by the profiles were stable throughout the duration of the analysis (5 months) and were recorded on all 55 datasets. Each subplot displays the mean of the average waveforms of the instances of each profile. The color range surrounding the plots repesents the standard deviation across all instances average waveforms. We can see that all 4 single-units experienced very little changes in their average waveforms throughout the duration of the analysis.

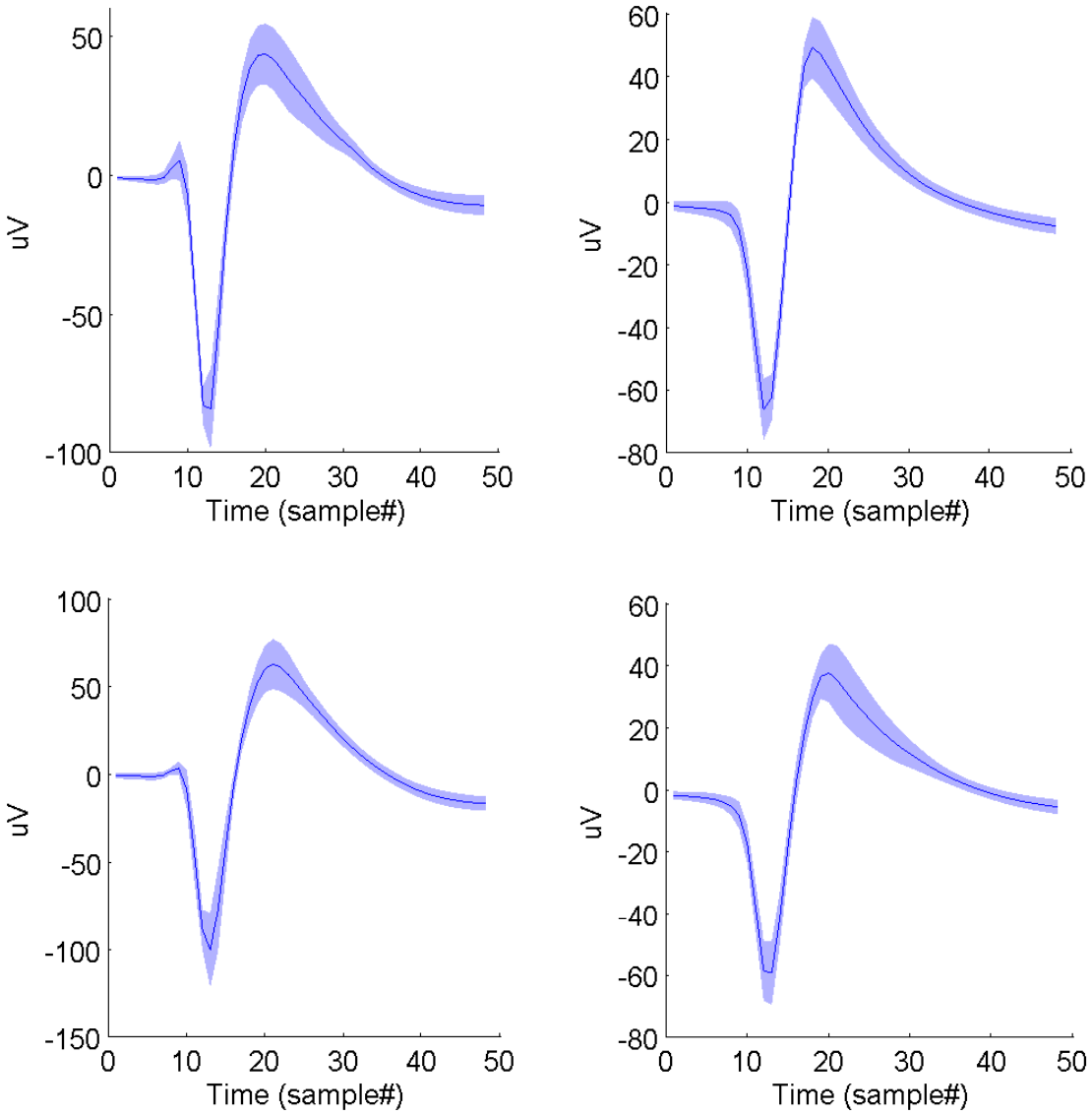

Figure 3.11: The figure shows 4 sample profiles recorded on channels 23, 25, 29, and 42 respectively. The single-units represented by the profiles were stable throughout the duration of the analysis (5 months) and were recorded on all 55 datasets. Each subplot displays the mean of the average waveforms of the instances of each profile. The color range surrounding the plots repesents the standard deviation across all instances average waveforms. The single-units in this experienced bigger changes in their average waveforms throughout the duration of the analysis than those in Figure 3.10. The algorithm was still able to capture those changes and correctly classify the single-units instances.

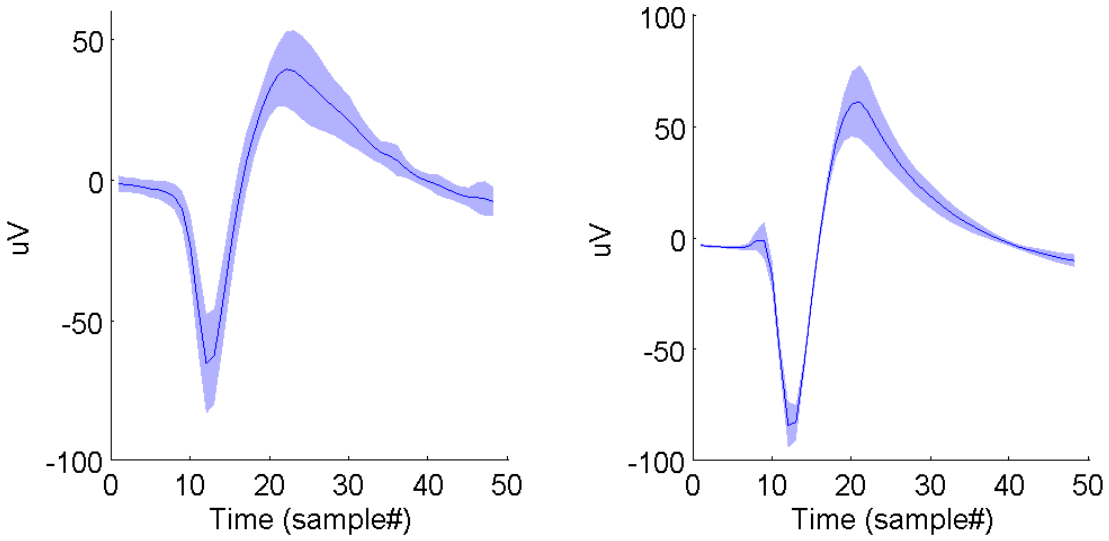

Figure 3.12: The figure shows 2 sample profiles recorded on channels 44 and 48 respectively. The single-units represented by the profiles were unstable and they were only recorded on 18 and 15 datasets respectively (out of 55 datasets). We can see that the average waveforms of both units are unstable and there is much variability across days.

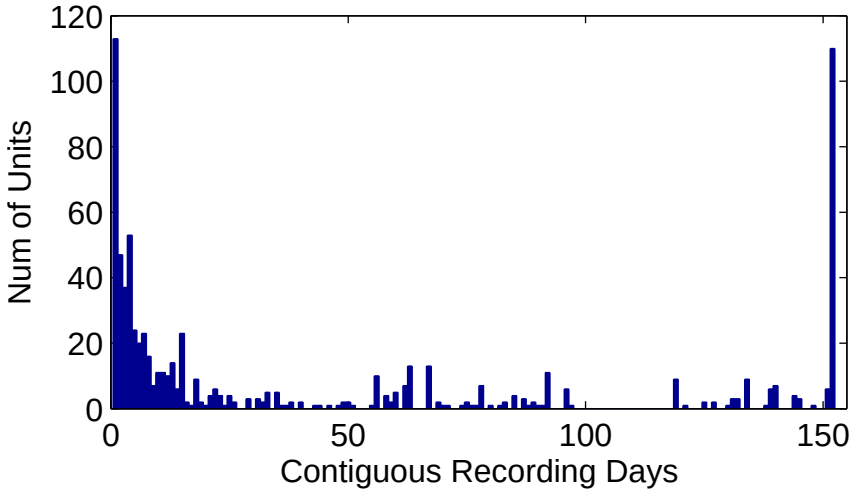

Figure 3.13: Histogram of the number of contiguous days a unit is likely to be recorded on until it drops. The large peaks on the left and right of the plot demonstrate that once a unit becomes stable, it is likely to remain stable for a long time.

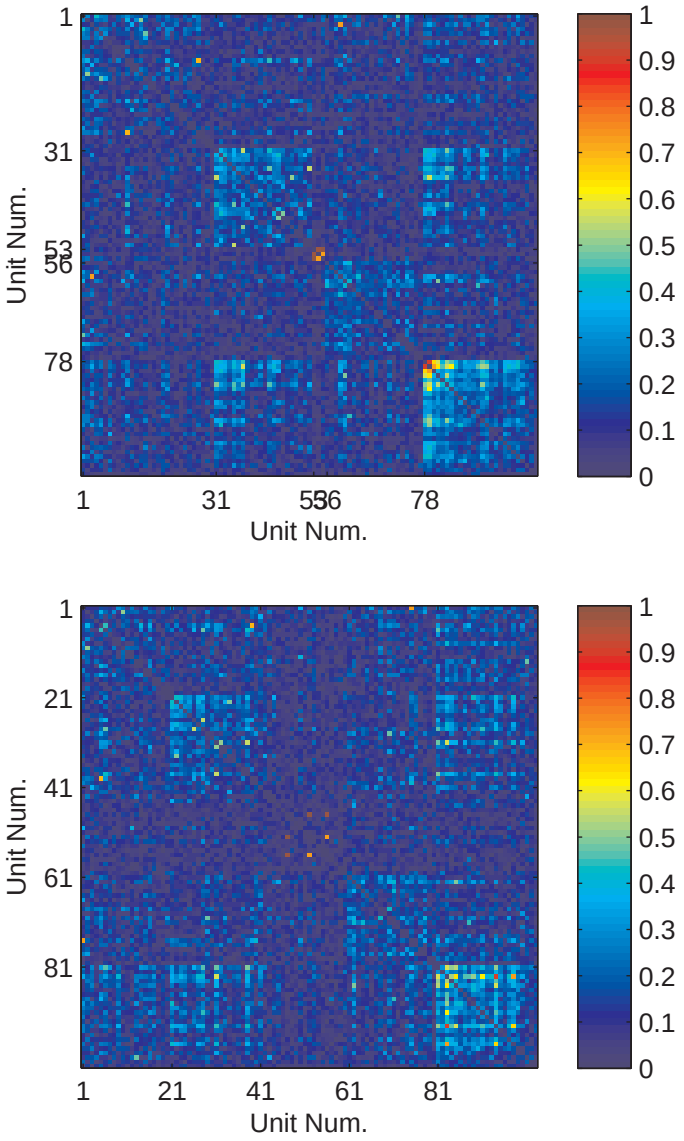

Figure 3.14: The first figure shows the correlation matrix obtained by the functional connectivity algorithm where spectral clustering was applied on the matrix to partition the units into 5 clusters. The numbers on the axes indicated the start of each cluster. It is clear that the clustering algorithm results in unbalanced clusters, this is especially true for cluster 3 that contains only 3 units while cluster 1 contains 30 units. The second figure shows that correlation matrix obtained by balancing the number of units across clusters while taking the functional connectivity between units into consideration. The strongly connected units in the initial clusters remained while the weakly connected ones were transferred to other clusters.

Figure 3.15: Mapping of the cluster units on the physical recording array.

2. Finding the most correlated unit to the dropped unit using MST as described in Section 3.6.2 (*MSTFuncCon*).

3. Find the unit that is most functionally connected to the remaining units in the cluster as described in Section 3.6.1 (*ClusterFuncCon*).

The comparison was performed by dropping each unit in the reach cluster and replacing the dropped unit with the replacement unit provided by each strategy. Two metrics were used in this comparison:

1. The average of the mean squared error (**MSE**) between the decoded velocity using the original cluster and that using the cluster with the replacement unit across all units of the cluster. The lower the average MSE, the better the replacement strategy is, because this is an indication that it provides replacement units close to the dropped one.

2. The number of times the strategy provided the replacement unit that has the least MSE among the free units pool (**CorRepl**). The higher the value of this metric, the better the replacement strategy is. This is because it is indication that the replacement strategy provides the best replacement unit available in the pool of free units a greater of number of times.

Tables 3.3 and 3.4 show the performance of the three strategies based on the two metrics. Both *UnitFuncCon* and *MSTFuncCon* provided replacement units that gave the lowest average MSE in 40% of the sessions under test. However, in 70% of the sessions *MSTFuncCon* had the highest number of correct replacements, compared to 50% of the sessions in case of *UnitFuncCon* (there was a tie in 20% of the sessions). *ClusterFuncCon* gave the

Table 3.3: Replacement Strategies Performance - average MSE between the reach velocity decoded using the original cluster and that decoded using the cluster with a replacement unit across all cluster units.

Strat./Day	1	2	3	4	5	6	7	8	9	10
UnitFuncCon	0.24	0.29	**0.28**	0.30	**0.27**	**0.23**	**0.24**	0.37	0.41	0.30
MSTFuncCon	**0.22**	**0.25**	0.29	0.33	0.28	0.31	0.38	0.30	**0.24**	**0.26**
ClusterFuncCon	0.29	0.30	0.29	**0.28**	0.28	0.25	0.28	**0.29**	0.31	0.31

Table 3.4: Replacement Strategies Performance - the number of times the strategy provided the replacement unit that has the least MSE among the free units pool.

Strat./Day	1	2	3	4	5	6	7	8	9	10
UnitFuncCon	**8**	6	**7**	**5**	5	5	**4**	**3**	2	3
MSTFuncCon	**8**	**10**	4	2	**7**	**6**	1	**3**	**4**	**5**
ClusterFuncCon	3	0	0	0	0	1	3	1	0	0

worst performance in both metrics and that is expected since this analysis was done **offline** (after recording) and in this strategy I was looking for the unit that would be in synchrony with the remaining cluster unit rather than looking for the closest unit to the dropped one. The effect of this choice can be better evaluated **online** (while the subject is controlling the robotic arm). In Figure 3.16, I show a subset of the results of the offline reach velocity decoding using each of the replacement strategies. The firing rates of the replacement unit were measured during the same interval as those of the dropped one.

I also tested the effect of replacing a unit on the decoding of the succeeding datasets and that can be achieved by using the replacement units found on day 1 in all succeeding days as well. Table 3.5 gives the MSE between the velocity decoded using the original cluster and that decoded using the cluster with the replacement unit averaged across all the units of the cluster and Figure 3.17 summarizes the testing results. As can be observed, the average MSE of all replacement strategies fluctuates around a constant value (shown by the short error

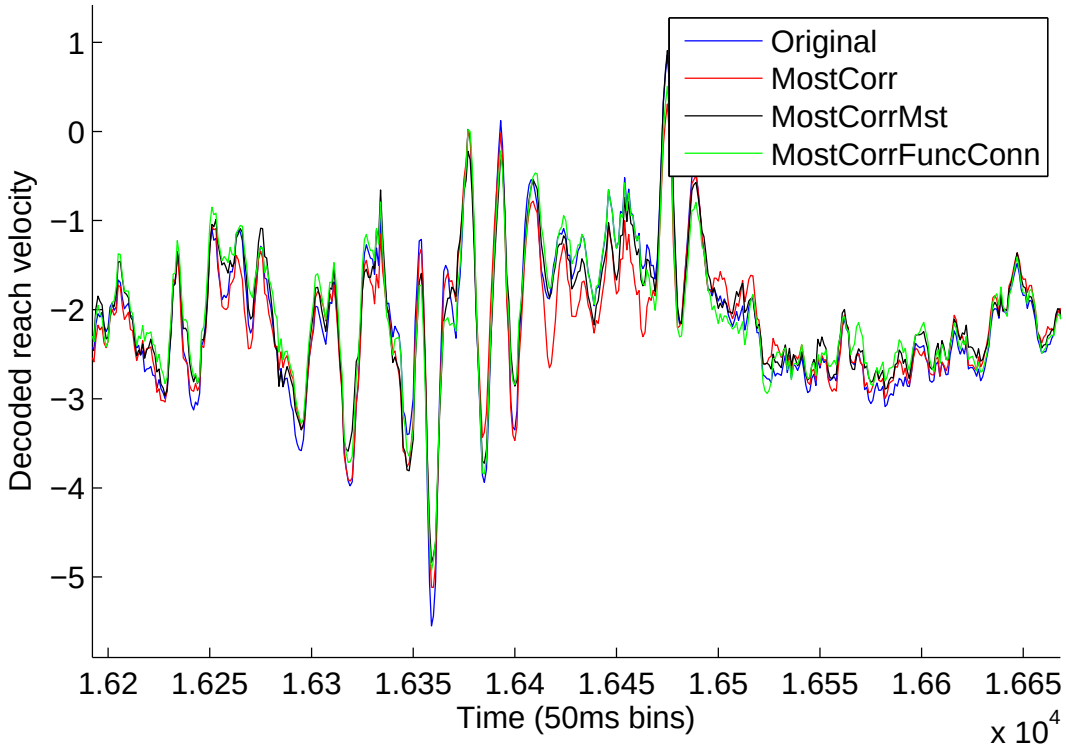

Figure 3.16: A subset of the session recorded on day 1. One unit was dropped and each replacement strategy provided a replacement unit to be used in the offline decoding of the reach velocity instead of the dropped one.

Table 3.5: Replacement Strategies Performance - average MSE between the reach velocity decoded using the original cluster and that decoded using the cluster with a replacement unit across all cluster units - Testing datasets.

Strat./Day	1	2	3	4	5	6	7	8	9	10
UnitFuncCon	0.25	0.31	0.30	0.29	0.30	0.28	0.29	0.31	0.29	0.26
MSTFuncCon	0.23	0.29	0.29	0.28	0.30	0.28	0.29	0.34	0.31	0.26
ClusterFuncCon	0.30	0.32	0.31	0.28	0.32	0.29	0.33	0.35	0.36	0.35

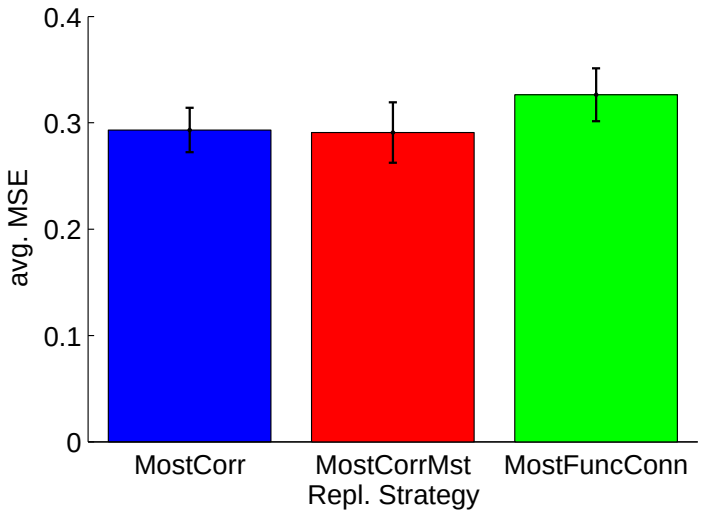

Figure 3.17: The figure summarizes the performance of the three replacement strategies when tested on 9 unseen datasets. The replacement unit in each strategy was selected on the first day and then it was used on the 9 subsequent days. The error bars are the standard deviation of the average MSE across days.

bars) and does not deteriorates with time which proves the effectiveness of the replacement strategies. I also performed a t-test to see if one strategy performed better than the other. Both the *UnitFuncCon* and *MSTFuncCon* were better than the *ClusterFuncCon* ($p > 0.004$ and $p > 0.008$ respectively).

Finally, I show the correlation matrix of the cluster and it corresponding MST on day 1 in Figure 3.18. Using the formula defined in equation 3.2, pairwise correlations are converted into distances. This is a non-linear transformation where a zero-correlation is mapped to $\sqrt{2}$ distance and a unit-correlation is mapped to zero-distance. A correlation of -1 is mapped to

a distance equals 2. Therefore, the boundaries of the calculated distance is $[0, 2]$. The MST construction algorithm is greedy and it includes edges with small weights (high correlation) as long as it does not form a loop in the MST. Looking at the correlation matrix, we can expect certain edges to be present in the MST, such as (5,15) and (5,6) since they have a higher correlation than other edges.

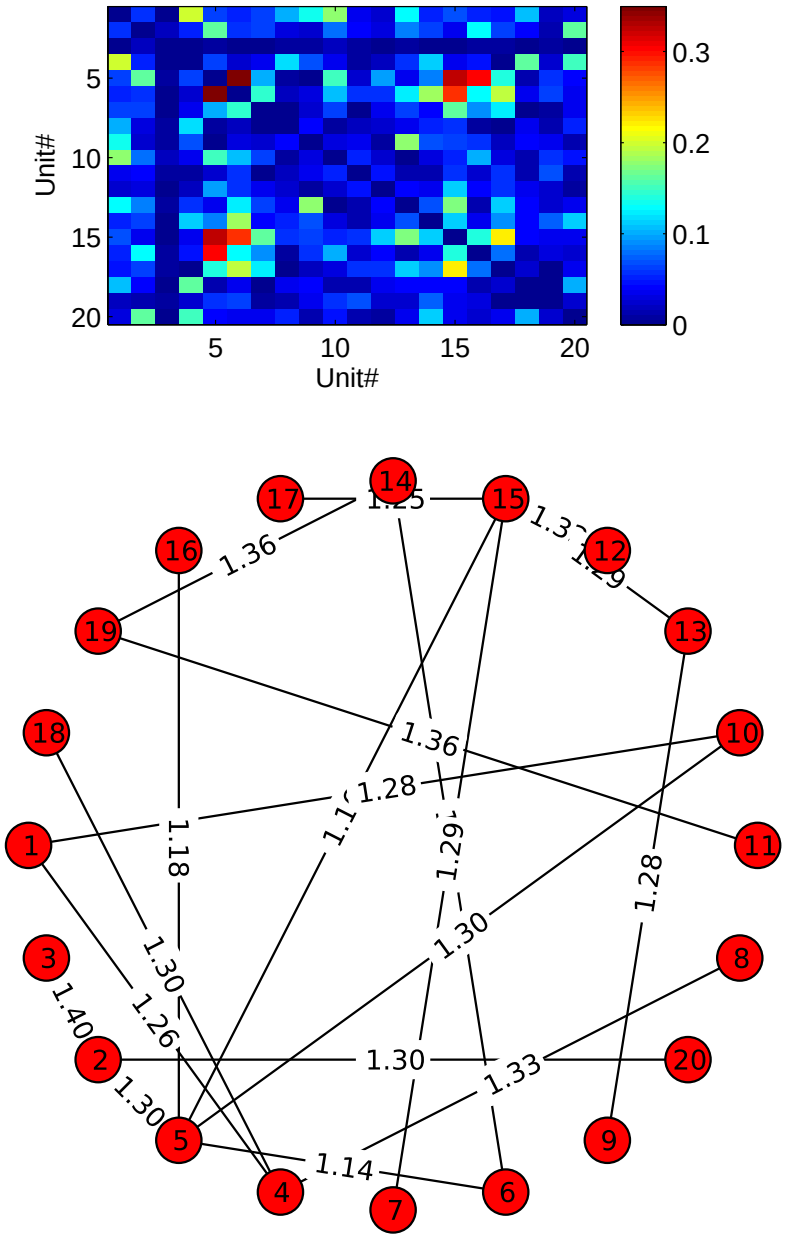

Figure 3.18: Figure a displays the augmented correlation matrix of the cluster units on day 1. Figure b shows the MST formed from the correlation matrix.

Chapter 4

Discussion

In this chapter, I give a summary of the topics presented in this study, list the advantages and drawbacks of the techniques used, and give directions to future studies and improvements.

In this study, I presented a single-unit tracking algorithm that is based on waveform features and firing characteristics of single-units and that does not require the subject to be engaged in any behavioral task. It also requires a short recording time (15 minutes of recording time) to perform the tracking. I also specified a set of criteria to assess the stability of a single-unit and to classify it into one of three classes: stable, partially stable, and unstable. I tracked the stability of single-units over the duration of 150 days where the first session was recorded three months after implant. The number of recorded units was almost constant during this duration and there were many stable units. This shows that the arrays has not reached the stage where it is unable to detect spike on a significant number of its electrodes. I also used the stability criteria to boost the performance of the tracking classifier by comparing the newly recorded unit against more than one instance of the previously recorded single units on a given channel and selecting the instance that is closest to the new unit. When the algorithm was tested on eight datasets, the accuracy of classification reached 91% when using all the features I proposed and 90% when using only 37% of the features. 82% of the profiles tracked by the algorithm matched the experts' manual tracking. Furthermore, the running time was short and the convergence occurred

at about 12 seconds per channel. It can also reach 8 seconds per channel at a slightly reduced accuracy. This makes the algorithm suitable for real-time BMI applications such as identifying units at the start of a recording session to be used in brain-control decoding.

One possible improvement to the tracking algorithm is to perform a two-level classification paradigm. The current paradigm is based on one classifier that is used for all channels. However, different channels have different properties, for example units might experience more variability on one channel than on others. A human expert can capture the variability of a channel across days by looking at the channel's waveforms, while an automated method cannot because the training points extracted from this channel would contaminate the classification decision or might be considered outliers. In the two-level classification paradigm, a custom classifier for each channel can be trained using training features extracted from this channel's units only. The challenges in this paradigm include the need for a large number of training datasets to generate enough data points for classifiers' training and ways to handle the introduction of new single-units on the channel. These challenges should be addressed before implementing this paradigm.

A drawback of my tracking technique is that it does not include any tuning features, which are considered a strong proof of whether the recordings on two different days were from the same unit or not. Although each unit recorded on an electrode has its characteristic waveform, it was shown that the shape of the waveform depends on the relative location of the recording electrode to the cell [20]. It may happen that the electrode lies exactly midway between two cells and the same waveform shape is recorded for both cells. Also, the micro-movement occurring in the brain might have a major effect on the shape of the recorded waveform if a different part of the cell body is closer to the recording electode. However, there are some drawbacks in using the tuning features in tracking units. First of all, the

subject has to be engaged in a behavioral task and the tuning features depend on this task Second, changes might occurr to the tuning function of units as the subject learns the task.

I also proposed a unit selection algorithm for BMI decoding as an application of the unit tracking algorithm. For a fixed decoder, the mapping from the neural space to the kinematic parameters should be fixed, and in order to do so, the same units have to be used with the same decoder coefficients across days. I use the tracking algorithm to identify units at the start of each recording session for this purpose. But then comes the question of how to select the units to use for decoding. I specified four criteria for this and explained the reasoning behind each:

1. Units have to be stable to guarantee the reliability and long lasting of those units (based on the results of [11]).

2. Units within a cluster have to be functionally connected because functionally connected units are potential cell assemblies that share common coding. We also hypothesized that using functionally connected units can speed the cortical pasticity reflecting learning of the behavioral task [1].

3. The number of units across the clusters should be the same to provide the subject with the same degree of control for each degree of freedom by assigning equal number of units across clusters. This way, we also divide the labor equally over units so that the control would not deteriorate in case of a unit drop.

4. Units recorded on the same electrode should be put in the same cluster to control the same degree of freedom. This is due to the high probability of them sharing the same source of input [48] and to minimize the effect of misclassifications in spike sorting and units tracking by making the units still control the same degree of freedom.

Some of clusters that resulting from the unit selection algorithm show some degree of localization on the physical recording array. The balancing algorithm also maintained the same connectivity strength in the clusters output from the functional connectivity algorithm and it only swapped the loosely connected units.

I also described the details of three strategies that provide replacements for dropped decoding units and they are all based on the augmented correlation matrix resulting from the functional connectivity algorithm. Using an offline analysis, selecting the most correlated replacement to the dropped unit using graph-theory algorithms provided the best results. However, furhter verification of these replacement techniques have to be done online.

One possible strategy to replace units is to use a longer lasting signal than the spikes, the local field potentials (LFP). The LFPs are a much slower wave than the APs and they are the aggregate of synchronized activity of the ensemble of neurons relatively away from the voltage sensing device ($> 250\mu\ m$) [36]. In [39], the authors report that they were able to infer the spiking activity of single-units from LFP in the primary visual cortex (V1). Zhuang et al. in [51] report that LFP can be used to decode hand 3-D position and grasp aperture. LFP can be used to build what might be called a "Pseudo Unit" by training a classifier or an artificial neural network (ANN) to predict the firing rates of the dropped unit from features extracted from the LFP signal. The LFP features include the signal power and phase across different frequency bands. One challenge towards the use of ANN to predict firing rates from LFP is how to quantize the resulting output to reflect the discrete-valed firing rates. K-means clustering is a possible solution to this problem, but it requires the knowledge of the maximum firing rate in advance.

Another possible application of the tracking algorithm is to track brain-mediated plasticity. This plasticity occurrs while the subject is learning the task where the tuning functions

of units and their functional connections with other units may change over time. However, to do so we need the accurate identification of units which can be provided by the tracking algorithm.

Finally, I point out that the techniques discussed here promotes the practicality of BMI system. Using the tracking algorithm, units can be accurately identified across days and this enables us to maintain a fixed mapping between the neuronal space and decoded movement parameters (i.e. fixed decoder coefficients). In addition, using the replacement strategies discussed, the system can react automatically to units dropping and maintain a relatively fixed mapping that is unnoticeable to the subject. This alleviates the need of a human operator that performs decoder calibration on a daily basis. Also, since the tracking algorithm does consume much time for its operation, any automated decoder calibration can run fast.

APPENDICES

Peak Matching

The *peak matching* finds the distance between two signals using the maximum and minimum peaks in both signals. The purpose of the *peak matching* is to provide comparable results to those given by an expert. It focuses on the features an expert pays attention to in comparing signals and the main ones of those are the maximal and minimal peaks.

Preprocessing

Cubic spline interpolation is performed on the signal to generate more data points. This aims at considering the signal a continuous one.

Algorithm

The algorithm finds the peaks of both signals, assigns weights to them (based on their shape) and finally quantifies the pairwise distance between peaks from both signals and summarizes them in one final distance.

Given two signals, $H(x)$ and $L(x)$, each of length n, the similarity of $H(x)$ to $L(x)$, $\tilde{\mathrm{r}}(H, L)$, is:

$$\tilde{\mathrm{r}}(H, L) = \mathrm{K}_1(H, L)\mathrm{K}_2(H, L) \tag{1}$$

where $\tilde{\mathrm{r}}(H, L)$ is the asymmetric similarity of H to L, and $\mathrm{K}_1(H, L)$ and $\mathrm{K}_2(H, L)$ are factors defined in Equations 2 and 4.

The first factor, $\mathrm{K}_1(H, L)$, quantizes the absolute distance between the two signals using the Manhattan distance and can be viewed as a scaling factor:

$$\mathrm{K}_1(H, L) = exp\left(\frac{\mathrm{S}(H, L)}{\bar{S}}\right) \tag{2}$$

where $\bar{S}$ is a parameter and $\mathrm{S}(H, L)$ is the Manhattan distance between the two signals and is defined as:

$$\mathrm{S}(H, L) = \sum_{x=1}^{n} |H(x) - L(x)| \tag{3}$$

where $H(x)$ and $L(x)$ are the x^{th} points in signals H and L respectively.

The second factor, $\mathrm{K}_2(H, L)$, finds a weighted distance between the peaks of the two signals using some heuristic measures.

$$\mathrm{K}_2(H, L) = \sum_{i} \mu_i(H)\mathrm{c}_i(H, L) \tag{4}$$

where $\mu_i(H)$'s are a set weights given to the peaks of $H(x)$ (see Section) and $\mathrm{c}_i(H, L)$'s are heuristic closeness factors between peaks in $H(x)$ and the closest peaks to them in $L(x)$ (see Section).

Peak Weights Calculation

In this section, I show how the weights given to peaks of a signal are calculated. The value of the weight given to a peak depends on its height compared to its neighboring points.

The first and second derivates of a signal $H(x)$ are defined as:

$$H^{(1)}(x) = \delta H(x)/\delta x \tag{5}$$

$$H^{(2)}(x) = \delta^2 H(x)/\delta x^2 \tag{6}$$

A peak at a point x_i is defined as a point in which the second derivative has a local minimum (this definition includes "shoulders" as well). Define also two points related to each peak, x_i^{left} and x_i^{right} as the nearest points on the left and right of each peak respectively where the second derivate is zero.

Finally, the weight of a peak at point x_i is:

$$\mu_i = \frac{\left| H^{(2)}(x_i) l_i \right|}{\sum_{j=1}^{\eta(H)} H^{(2)}(x_j) l_j} \tag{7}$$

where l_i is the maximal distance from the points on the signal in the interval $[x_i^{left}, x_i^{right}]$ and the line joining x_i^{left} and x_i^{right}. $\eta(H)$ is the total number of peaks in signal H.

Closeness Factor Calculation

The closeness factor (similarity) between peak i in signal $H(x)$ and peak j in signal $L(x)$, $\tilde{\mathrm{c}}_{ij}$, is the multiplication of three factors:

$$\tilde{\mathrm{c}}_{ij} = \mathrm{c}_{ij}^{x} \mathrm{c}_{ij}^{y} \mathrm{c}_{ij}^{form} \tag{8}$$

The first factor is the distance between the x-location of the two peaks:

$$\mathrm{c}_{ij}^{x} = exp\left\{-\left(\frac{\triangle x_{ij}}{\delta_x}\right)^2\right\} \tag{9}$$

where δ_x is a parameter, and

$$\triangle x_{ij} = x_i - \tilde{x}_j \tag{10}$$

x_i is the x-location of peak i in signal $H(x)$, and $\tilde{x}_j$ is the x-location of peak j in signal $L(x)$.

Similarly, the second factor is the distance between the y-location of the two peaks:

$$\mathrm{c}_{ij}^{y} = exp\left\{-\left(\frac{\triangle y_{ij}}{\delta_y}\right)^2\right\} \tag{11}$$

where δ_y is a parametre, and

$$\triangle y_{ij} = H(x_i) - L(\tilde{x}_j) \tag{12}$$

Finally, the third factor aims at finding the distance between the shapes of the two peaks:

$$\mathrm{c}_{ij}^{form} = exp\left\{-\frac{\mathrm{f}_{ij}\bar{\nu}}{\eta(H) + \eta(L)}\right\} \tag{13}$$

$$\mathrm{f}_{ij} = \frac{\left|H^{(1)}(x_i^{left}) - L^{(1)}(\tilde{x}_j^{left})\right|}{\left|H^{(1)}(x_i^{left}) + L^{(1)}(\tilde{x}_j^{left})\right| + \varepsilon_1} + \frac{\left|H^{(1)}(x_i^{right}) - L^{(1)}(\tilde{x}_j^{right})\right|}{\left|H^{(1)}(x_i^{right}) + L^{(1)}(\tilde{x}_j^{right})\right| + \varepsilon_1} + \frac{\left|(x_i^{right} - x_i^{left}) - (\tilde{x}_j^{right} - \tilde{x}_j^{left})\right|}{\left|(x_i^{right} - x_i^{left}) + (\tilde{x}_j^{right} - \tilde{x}_j^{left})\right| + \varepsilon_2} \tag{14}$$

where ε_1 and ε_2 are parameters.

The closeness factor of a peak is defined as the maximum closeness factor it has with any of the peaks in the other signal:

$$\mathrm{c}_i = \max_{j \in J}(\tilde{\mathrm{c}}_{ij}) \tag{15}$$

The calculated similarity is asymmetric and the symmetric *peak matching* similarity between two signals is calculated as the geometric mean of the asymmetric ones:

$$\mathrm{r}(H, L) = \sqrt{\tilde{\mathrm{r}}(H, L)\tilde{\mathrm{r}}(L, H)} \tag{16}$$

and the *peak matching* distance is:

$$\mathrm{D}(H, L) = 1 - \mathrm{r}(H, L) \tag{17}$$

Figure 1 shows an illustration of the previous steps to calculate the peak matching similarity of one signal to another.

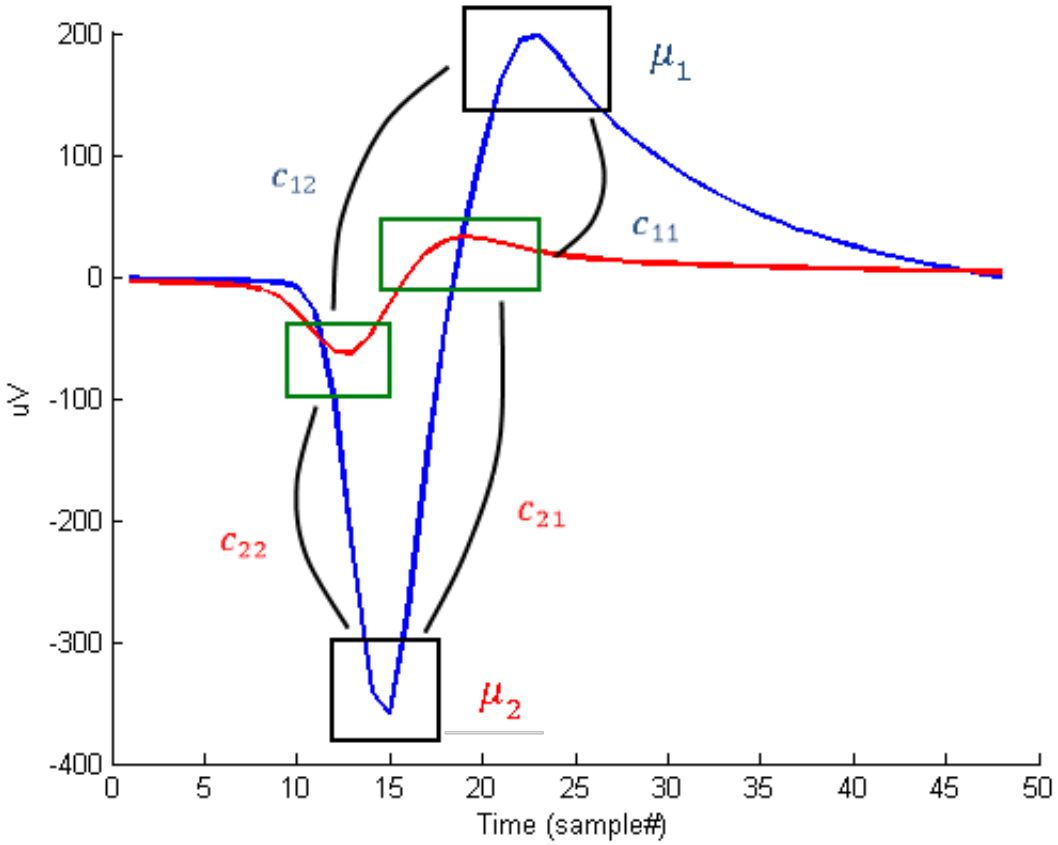

Figure 1: The figure shows the steps of calculating the peak matching similarity of a signal x (blue) to another signal y (red). Both x and y have two peaks. Weights are given to the peaks of x, μ_1 and μ_2. Closeness factors are calculated between pairwise peaks in both signals. In this case, there are 4 closeness factors to calculate (c_{11}, c_{12}, c_{21}, and c_{22}). For each signal in x, we select the maximum closeness factor it has with any of the peaks in y. We end up with 2 factors, c_1 and c_2, one for each peak in x. The total similarity of x to y is then the sum of the peak weights multiplied by the closeness factors for all peaks of x (i.e. $\mu_1 c_1 + \mu_2 c_2$).

Functional Connectivity Algorithm

The following are the details of computing each step of the functional connectivity algorithm used in Section 3.3.1. The details below are mostly adapted from [13].

At any given timescale j, the spike train of a neuron p can be expressed as:

$$s_p^j(t) = W^{(j)} s_p(t), j = 0, 1...J \tag{18}$$

where $s_p(t)$ is the spike train (0's and 1's)of neuron p, $W^{(j)}$ is a lumped matrix representing the Haar wavelet transform and J is the total number of timescales. For instance, equation shows the Haar wavelet transform lumped matrix for the first time scale, $W^{(1)}$.

$$W^{(1)} = \sqrt{2} \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & -1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & -1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & -1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & -1/2 \end{bmatrix}$$

"Multiplying by the top half of $W^{(1)}$ by the spike train s_p gives the probability of firing in successive bins that are two samples in size. Multiplying by the bottom half of $W^{(1)}$ by the spike train s_p gives the relative change in the firing pattern in successive bins that are two samples in size."

The sample cross-correlation between neurons p and q at timescale j can be expressed as:

$$\sigma^j_{pq}(t_i, t_j) = E\{s^j_p(t_i)s^j_q(t_j)\} \tag{19}$$

The full correlation matrix at timescale j, $\Sigma^{(j)}$, can be formed using equation 19 for all $p, q \in \{1, ..., P\}$, where P is the number of neurons in the population.

A block diagonal matrix can then be formed as:

$$R = \begin{bmatrix} \Sigma^{(0)} & & & \\ & \Sigma^{(1)} & & \\ & & \ddots & \\ & & & \Sigma^{(J)} \end{bmatrix} \tag{20}$$

Next, R is decomposed using SVD:

$$R = \sum_{i=1}^{Q} \lambda u_i u_i^T \tag{21}$$

and finally, the pairwise similarity (augmented correlation) of the neurons is obtained using the weighted sum of the first few dominant modes of R.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Ahissar, Vaadia, Ahissar, Bergman, Arieli, and Abeles, *Dependence of cortical plasticity on correlated activity of single neurons and on behavioral context*, Science **257** (1992), no. 5075, 1412–1415.

[2] Balasubramanian, Southerland, Vaidya, Qian, Eleryan, Fagg, Sluzky, Oweiss, and Hatsopoulos, *Operant conditioning of a multiple degree-of-freedom brain-machine interface in a primate model of amputation*, Engineering in Medicine and Biology Society, 2013. EMBS'13. 35th Annual International Conference of the IEEE, IEEE, 2013, p. To appear.

[3] Bhattacharyya, *On a measure of divergence between two statistical populations defined by their probability distributions*, Bull. Calcutta Math. Soc **35** (1943), no. 99-109, 4.

[4] Bishop et al., *Introduction to information retrieval*, vol. 1, Cambridge University Press, 2008.

[5] Borgatti, *Multivariate statistics*, University Lecture, 2007.

[6] Carmena, Lebedev, Henriquez, and Nicolelis, *Stable ensemble performance with single-neuron variability during reaching movements in primates*, The Journal of neuroscience **25** (2005), no. 46, 10712–10716.

[7] Chen and Fetz, *Characteristic membrane potential trajectories in primate sensorimotor cortex neurons recorded in vivo*, Journal of neurophysiology **94** (2005), no. 4, 2713–2725.

[8] Chestek, Batista, Santhanam, Byron, Afshar, Cunningham, Gilja, Ryu, Churchland, and Shenoy, *Single-neuron stability during repeated reaching in macaque premotor cortex*, The Journal of Neuroscience **27** (2007), no. 40, 10742–10750.

[9] Chestek, Gilja, Nuyujukian, Foster, Fan, Kaufman, Churchland, Rivera-Alvidrez, Cunningham, Ryu, et al., *Long-term stability of neural prosthetic control signals from silicon cortical arrays in rhesus macaque motor cortex*, Journal of neural engineering **8** (2011), no. 4, 045005.

[10] Dayan, Abbott, and Abbott, *Theoretical neuroscience: Computational and mathematical modeling of neural systems*, Taylor & Francis, 2001.

[11] Dickey, Suminski, Amit, and Hatsopoulos, *Single-unit stability using chronically implanted multielectrode arrays*, Journal of neurophysiology **102** (2009), no. 2, 1331–1339.

[12] Domingos and Pazzani, *On the optimality of the simple bayesian classifier under zero-one loss*, Machine learning **29** (1997), no. 2-3, 103–130.

[13] Eldawlatly, Jin, and Oweiss, *Identifying functional connectivity in large-scale neural ensemble recordings: a multiscale data mining approach*, Neural computation **21** (2009), no. 2, 450–477.

[14] Fraser and Schwartz, *Recording from the same neurons chronically in motor cortex*, Journal of Neurophysiology **107** (2012), no. 7, 1970–1978.

[15] Ganguly and Carmena, *Emergence of a stable cortical map for neuroprosthetic control*, PLoS biology **7** (2009), no. 7, e1000153.

[16] Gilletti and Muthuswamy, *Brain micromotion around implants in the rodent somatosensory cortex*, Journal of neural engineering **3** (2006), no. 3, 189.

[17] Griffith and Humphrey, *Long-term gliosis around chronically implanted platinum electrodes in the¡ i¿ rhesus macaque¡/i¿ motor cortex*, Neuroscience letters **406** (2006), no. 1, 81–86.

[18] Hatsopoulos and Donoghue, *The science of neural interface systems*, Annual review of neuroscience **32** (2009), 249.

[19] Hazewinkel, *Encyclopaedia of mathematics, supplement iii*, vol. 13, Springer, 2001.

[20] Henze, Borhegyi, Csicsvari, Mamiya, Harris, and Buzsáki, *Intracellular features predicted by extracellular recordings in the hippocampus in vivo*, Journal of neurophysiology **84** (2000), no. 1, 390–400.

[21] Hikosaka, Nakamura, Sakai, and Nakahara, *Central mechanisms of motor skill learning*, Current opinion in neurobiology **12** (2002), no. 2, 217–222.

[22] Humphries, *Spike-train communities: finding groups of similar spike trains*, The Journal of Neuroscience **31** (2011), no. 6, 2321–2336.

[23] Jackson and Fetz, *Compact movable microwire array for long-term chronic unit recording in cerebral cortex of primates*, Journal of neurophysiology **98** (2007), no. 5, 3109–3118.

[24] Jackson, Mavoori, and Fetz, *Long-term motor cortex plasticity induced by an electronic neural implant*, Nature **444** (2006), no. 7115, 56–60.

[25] Krüger, Caruana, Dalla Volta, and Rizzolatti, *Seven years of recording from monkey cortex with a chronically implanted multiple microelectrode*, Frontiers in neuroengineering **3** (2010).

[26] Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proceedings of the American Mathematical society **7** (1956), no. 1, 48–50.

[27] Kullback and Leibler, *On information and sufficiency*, The Annals of Mathematical Statistics **22** (1951), no. 1, 79–86.

[28] Leiserson, Rivest, Stein, and Cormen, *Introduction to algorithms*, The MIT press, 2001.

[29] Linderman, Gilja, Santhanam, Afshar, Ryu, Meng, and Shenoy, *Neural recording stability of chronic electrode arrays in freely behaving primates*, Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE, IEEE, 2006, pp. 4387–4391.

[30] Liu, McCreery, Carter, Bullara, Yuen, and Agnew, *Stability of the interface between neural tissue and chronically implanted intracortical microelectrodes*, Rehabilitation Engineering, IEEE Transactions on **7** (1999), no. 3, 315–326.

[31] Lütcke, Margolis, and Helmchen, *Steady or changing? long-term monitoring of neuronal population activity*, Trends in neurosciences (2013).

[32] Von Luxburg, *A tutorial on spectral clustering*, Statistics and computing **17** (2007), no. 4, 395–416.

[33] Manning et al., *Pattern recognition and machine learning*, vol. 1, springer New York, 2006.

[34] Nicolelis, Dimitrov, Carmena, Crist, Lehew, Kralik, and Wise, *Chronic, multisite, multielectrode recordings in macaque monkeys*, Proceedings of the National Academy of Sciences **100** (2003), no. 19, 11041–11046.

[35] Nicolelis and Lebedev, *Principles of neural ensemble physiology underlying the operation of brain–machine interfaces*, Nature Reviews Neuroscience **10** (2009), no. 7, 530–540.

[36] Oweiss (ed.), *Statistical signal processing for neuroscience and neurotechnology*, 1st ed., Academic Press, 2010.

[37] Polikov, Tresco, and Reichert, *Response of brain tissue to chronically implanted neural electrodes*, Journal of neuroscience methods **148** (2005), no. 1, 1–18.

[38] Prim, *Shortest connection networks and some generalizations*, Bell system technical journal **36** (1957), no. 6, 1389–1401.

[39] Rasch, Gretton, Murayama, Maass, and Logothetis, *Inferring spike trains from local field potentials*, Journal of neurophysiology **99** (2008), no. 3, 1461–1476.

[40] Reich, Mechler, Purpura, and Victor, *Interspike intervals, receptive fields, and information encoding in primary visual cortex*, The Journal of Neuroscience **20** (2000), no. 5, 1964–1974.

[41] Rousche and Normann, *Chronic recording capability of the utah intracortical electrode array in cat sensory cortex*, Journal of neuroscience methods **82** (1998), no. 1, 1–15.

[42] Salkind, *Encyclopedia of research design*, vol. 1, Sage Publications, Incorporated, 2010.

[43] Sedgewick, *Algorithms in java, parts 1-4*, vol. 1, Addison-Wesley Professional, 2002.

[44] Strelkov, *A new similarity measure for histogram comparison and its application in time series analysis*, Pattern Recognition Letters **29** (2008), no. 13, 1768–1774.

[45] Suner, Fellows, Vargas-Irwin, Nakata, and Donoghue, *Reliability of signals from a chronically implanted, silicon-based electrode array in non-human primate primary motor cortex*, Neural Systems and Rehabilitation Engineering, IEEE Transactions on **13** (2005), no. 4, 524–541.

[46] Tolias, Ecker, Siapas, Hoenselaar, Keliris, and Logothetis, *Recording chronically from the same neurons in awake, behaving primates*, Journal of neurophysiology **98** (2007), no. 6, 3780–3790.

[47] Peter Torrione, Sam Keene, and Kenneth Morton, *PRT: The pattern recognition toolbox for MATLAB*, 2011, Software available at http://newfolderconsulting.com/prt.

[48] Vaadia, Haalman, Abeles, Bergman, Prut, Slovin, and Aertsen, *Dynamics of neuronal interactions in monkey cortex in relation to behavioural events*, Nature **373** (1995), no. 6514, 515–518.

[49] Warland, Reinagel, and Meister, *Decoding visual information from a population of retinal ganglion cells*, Journal of Neurophysiology **78** (1997), no. 5, 2336–2350.

[50] Wikipedia, *Minimum spanning tree*, July 2013.

[51] Zhuang, Truccolo, Vargas-Irwin, and Donoghue, *Decoding 3-d reach and grasp kinematics from high-frequency local field potentials in primate primary motor cortex*, Biomedical Engineering, IEEE Transactions on **57** (2010), no. 7, 1774–1784.

[52] Zumsteg, Kemere, O'Driscoll, Santhanam, Ahmed, Shenoy, and Meng, *Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems*, Neural Systems and Rehabilitation Engineering, IEEE Transactions on **13** (2005), no. 3, 272–279.