HEURISTIC SPIKE SORTING TUNER FOR THE DETERMINATION OF AN OPTIMAL PARAMETER SET FOR A GENERIC SPIKE SORTING ALGORITHM

by

David Atle Bjånes

B.S. in Electrical and Computer Engineering, Cornell University, 2012

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

David Atle Bjånes

It was defended on

May 9th, 2014

and approved by

Robert A. Gaunt, Ph.D., Assistant Professor, Dept. of Physical Medicine and Rehabilitation

Patrick J. Loughlin, Ph.D., Professor, Dept. of Bioengineering

Zhi-Hong Mao, Ph.D., Associate Professor, Dept. of Electrical and Computer Engineering

Thesis Advisor: Douglas J. Weber, Ph.D., Associate Professor, Dept. of Bioengineering

Thesis Co-Advisor: Ervin Sejdić, Ph.D., Assistant Professor, Dept. of Electrical and Computer Engineering

HEURISTIC SPIKE SORT TUNER FOR THE DETERMINATION OF AN OPTIMAL PARAMETER SET FOR A GENERIC SPIKE SORTING ALGORITHM

David Bjånes, M.S.

University of Pittsburgh, 2014

Extracellular microelectrodes frequently record neural activity from multiple sources in the vicinity of the electrode. Spike sorting generally describes the process of labeling each recorded spike waveform with the identity of its source neuron, which is required to conduct any further analysis of the neuronal spiking patterns. This process for spike sorting or isolating neural activity is often approached from an abstracted mathematical perspective such as calculating the Euclidean distance between spike waveform features in some lower dimensional space or using probability distributions to describe the isolation of neural activity or recorded spikes. However, these approaches ignore neurophysiological realities and result in the loss of important information that could improve the accuracy of these methods. Furthermore, standard algorithms typically require manual selection of at least one free parameter, which can have significant effects on the ultimate quality of the spike sorting and all resulting neurophysiological inferences.

We describe a Heuristic Spike Sorting Tuner (HSST) which determines the optimal choice of the set of free parameters for a given spike sorting algorithm. A set of heuristic metrics computes a neurophysiologically-based qualification score of an algorithm's output across a range of parameters. This qualification score measures unit isolation and signal discrimination, allowing HSST to select the best set of parameters for a sorting algorithm, resulting in high sort quality.

We demonstrate the power of this spike sorting framework, by comparing its performance across many existing spike sorting methods while using HSST to set their free parameters. The algorithm is robust over varied data (signal-to-noise ratio, number of units, relative size of units to each other, etc), and importantly; this approach requires no human supervision.

With simulated datasets, HSST reliably selects the optimal set of free parameters for many different sorting algorithms, allowing simple clustering techniques (such as K-Means) whose performance is highly dependent on correct parameter settings to outperform more complex algorithms. HSST outperforms expert manual sorters and is more robust at parameter estimation than other unsupervised algorithms, achieving this without sacrificing speed or stability.

Rather than being a spike sorting algorithm in its own right, HSST is a general framework that can incorporate any existing spike sorting algorithm, has an extendable set of heuristics and can be integrated in any existing neural signal processing stream. HSST makes use of known neurophysiological priors while simultaneously taking advantage of the power of abstract mathematical tools. We believe that this approach enables unsupervised spike sorting that exceeds the performance of previous methods, thereby enabling principled processing of large data sets without the significant confound of human intervention.

Key Words: **Heuristic**, **Spike Sorting**, **Sort Quality, Unsupervised**

TABLE OF CONTENTS

LIST OF EQUATIONS

LIST OF FIGURES

PREFACE

ACKNOWLEDGEMENTS

Many thanks to the entire Rehab Neural Engineering Lab (RNEL), specifically Dr. Rob Gaunt and Dr. Lee Fisher for guidance, direction and editing of the manuscript. Special thanks to Dr. Steve Chase at CMU for advice on statistical methods and exposure to the spike sorting topic.

Special thanks to my parents, Atle and Laura, for their constant support and love, propelling me early in life to succeed by never tiring of my "why" questions and encouraging me to always think critically, live in the moment and never stop pursuing my dreams. Thanks to my fiancée, Kim, for enduring many long frustrating nights when I was banging my head against a wall and sticking with me through it all. You are an inspiration to me!

FUNDING

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) Microsystems Technology Office under the auspices of Dr. Jack Judy (jack.judy@darpa.mil) through the Space and Naval Warfare Systems Center, Contract No: N66001-11-C-4171

1.0 INTRODUCTION

The fundamental goal of many neural recordings from the brain, spinal cord, and peripheral nervous systems is extracting the timing information of the firing rate of individual neurons. In the case of extracellular electrodes, existing technologies provide the ability to record the summed voltages generated by multiple neurons in the vicinity of the electrode [39]. This makes it difficult to distinguish between the activities of each neuron close to the electrode tip and to remove/ignore background noise generated by neurons further away. For our research, it is important to identify individual neural activity and detect/remove neural signals whose amplitudes are too low to confidently isolate their activity. Isolated activity from a single neuron is called a single unit [39]. Extracting this information is a multistep process [12] usually including 1) filtering the raw waveform, 2) detecting neural activity (*see Figure 1.1.A*), 3) extracting features of the waveform near the point of detected activity (*see Figure 1.1.B*), 4) clustering those extracted features into either single units or multi-unit activity (*see Figure 1.1.C*). Interesting hypothesis about the behavior of the nerve cells can then be tested on the basis of the information extracted [32].

1.1 NEUROBIOLOGY AND ELECTRICAL OVERVIEW

At the very foundation of the nervous system, a single neuron (or nerve cell) is the basic building block for communication. Nerve cells become electrically excited (or inhibited) by chemical inputs from other nerve cells, and after performing some operation on the incoming signals, they send a chemical output across a synapse to other neurons, triggered by an internal electrical pulse (known as an action potential). This exchange of information between neurons forms one of the most complex networks ever examined, containing multiple possible levels of abstraction for studying the different functions performed. For this discussion, let's start within an individual cell as the first layer of abstraction and examine the chemical changes which help produce a measurable electrical pulse, called a spike, or action potential.

The following information was extracted from various sections in the "Principles of Neuroscience" textbook by Kandel, Schwartz, and Jessell [64]. The cycle of triggering an action potential starts by a chemical reaction at a synapse which connects two neurons (*see Figure 1.2.C*). Neurotransmitters, many of which are proteins, released from other neurons, pass across the synaptic cleft (the small gap which electrically isolates neurons), located between the axon terminals of the transmitting neuron and the dendrites of the receiving neuron.

These neurotransmitters bind themselves to certain ion channels which regulate the flow of potassium (K^+), sodium (Na^+), calcium (Ca^{2+}) and other elements through the cell wall, thus changing the local permeability of the cell membrane. The flow of these charged elements into the cell changes the membrane potential and creates an unbalanced charge between the inside and outside of the cell (*Step 1 in Figure 1.2.A*). The nerve cell has a resting potential of about -70mV, balanced by a steady inward/outward flow of sodium and potassium. Voltage controlled ion channels regulate the flow of charged particles in and out of the cell to maintain this balance;

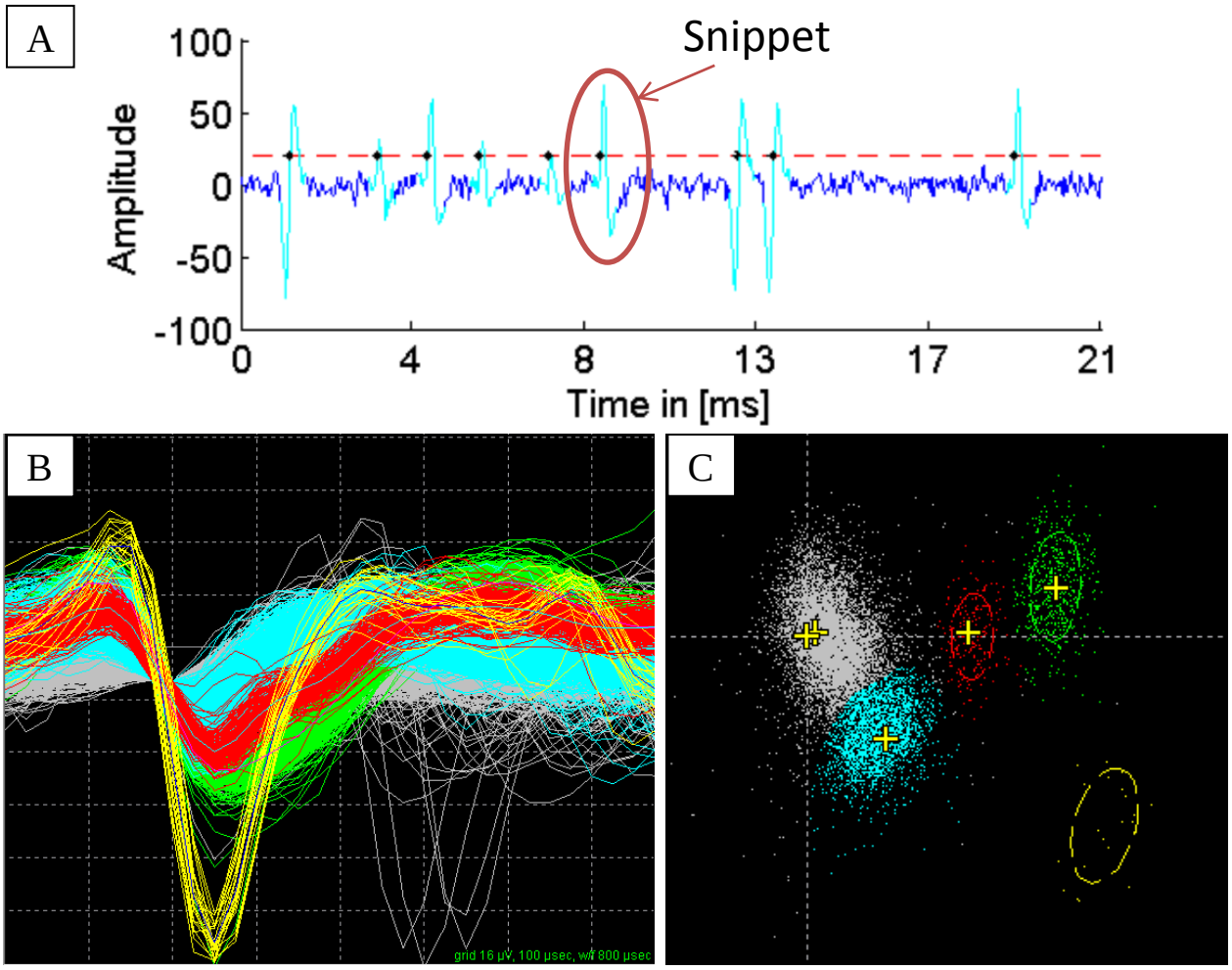

Figure 1.1: Snippet Extraction from Raw Waveform, Feature Space

Part A: A generated cartoon example of a continuous recording using a thresholding technique to detect action potential events. A one millisecond window is extracted from the waveform about the point of detection (black dots). This extracted waveform is called a snippet**. Part B:** shows aligned snippets from an actual recording from the spinal cord of a cat (*see Chapter 1, Motivation for Spike Sorting in Current Research)*. **Part C:** Those same waveforms show in Part B are clustered in 2D principle components (PCA) space [56].

if the membrane becomes depolarized, the voltage controlled ion channels which allow more potassium to flow, will open, lowering the membrane potential back to the resting potential and vice versa.

However, while the chemically gated ion channels flood the cell with sodium, the membrane potential increases, depolarizing the cell too quickly for the slowly acting voltage

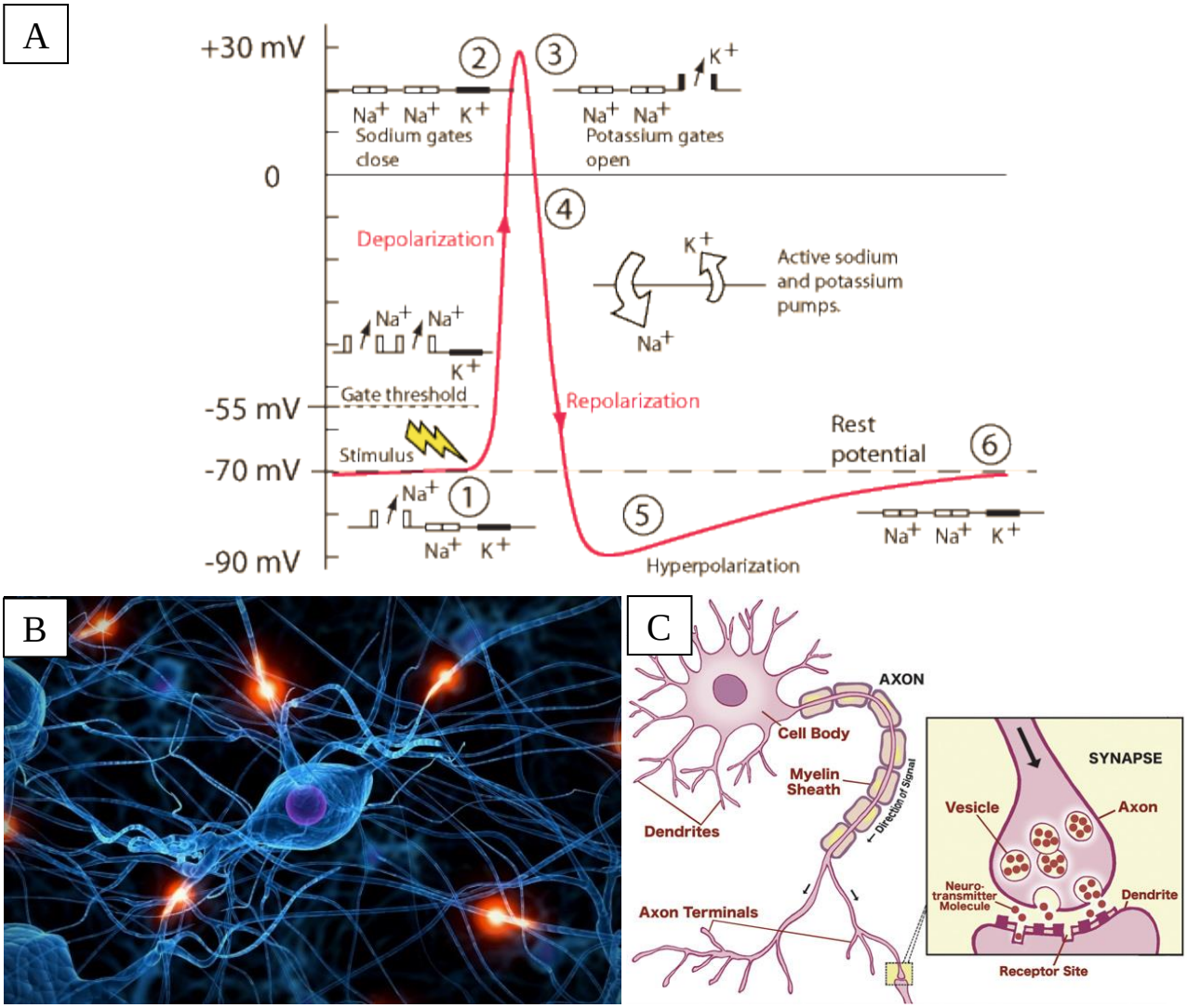

Figure 1.2: Neurons in the Body

Part A: [23], shows the 6 step cycle of an action potential. **Part B:** [21], shows an artist rendition of a single neuron. **Part C:** [24], shows the structure of a neural cell and the chemical release of neurotransmitter which links neurons together.

gated potassium channels to respond. The increase in voltage potential opens up more voltage gated sodium channels, spawning a chain reaction. Once the potential reaches approximately -55mV (depends on the type of nerve cell), the cell membrane has become too depolarize to stabilize itself, and an action potential is fired. Once all voltage-gated sodium channels have opened (*Step 2 in Figure 1.2.A*), the transmembrane potential reaches a peak, since the cell can't

depolarize any further. The slow responding potassium channels are now wide open, due to the high level of depolarization of the cell potential (Step 3 in Figure 1.2.A). Potassium flows into the cell and the rush of potassium ions hyperpolarizes the cell, lowering the electrical potential to below its resting potential (*Step 4 in Figure 1.2.A*). The potential dips below its resting potential, into a hyper polarized state (*Step 5 in Figure 1.2.A*), triggering the opening of a few voltage gated sodium channels, and eventually restoring the cell to its resting potential (*Step 6 in Figure 1.2.A*). This brief hyperpolarized phase is called relative refractory period and it is more difficult for the cell is to fire another action potential during this phase, because greater levels of sodium are required to raise the membrane potential high enough to trigger a spike. The absolute refractory period is due to inactivation of the sodium channels after they have all been opened. These voltage-gated sodium channels remain inactive until the cell has hyperpolarized and thus is impossible for the cell to trigger a spike during this phase. This sets a fundamental limit on the minimum firing rate of a neuron (usually 1 millisecond – although other types of neurons have a relative refractory periods of 5-20 milliseconds). This cycle of depolarization and repolarization continues down the axon, until the electrical potential reaches the axon terminals, where the increased potentials trigger the release of neurotransmitters, which travel across the synaptic cleft (*see Figure 1.2.C*) starting the process all over again in nearby neurons.

The second level of abstraction we can discuss pertains to the network level, examining a population of single neurons and their communication between each other [37] [50] in response to some network input. We can map which neurons fire a spike in correlation with many different inputs. We can also analyze the output of the network to see what processing of the input signal occurs. For complex brain functions involving many regions of the brain, we may not be able to see the whole picture at this level of abstraction and thus only have a narrow

window into the network behavior [38]; however, this layer of abstraction is excellent for understanding neural network dynamics for nearby neurons [37] or less complex networks in the spinal cord or peripheral nervous system.

The third level corresponds to two networks (populations of neurons) and examining how they relate to each other). We can easily see where this line of reasoning is headed. At the very top layer, we have an enormously complex bi-directional system including the brain (with thousands of populations of neurons) which communicates with the spinal cord which communicates with the peripheral nervous system and back again.

The reason for using a multi-tiered abstraction process for looking at this structure is the sheer enormity of the network. An average cortical neuron has on the order of 10,000-100,000 neural inputs, and 100-1000 outgoing neuron connections. The entire complex system is further expanded when we can estimate the total number of neurons in the cortex totals more than 20 trillion [25], with 10^14 synaptic connections between them all. As a whole, attempting to map the system with this level of detail is clearly entirely unmanageable to study in an organized and constructive way. In order to break down this complex structure into a smaller understandable scale, it would be advantageous to capture the inputs and outputs of a small group of neurons and study their interactions with each other, while learning how signals propagate through the network.

Many different methods (*see Figure 1.3*) exist to map neural activity, including measuring the electrical potential in a population of neurons via a Utah Array [60], measuring the flow of blood in a larger population of the brain using fMRI [61], surface (non-penetrating) electrodes recording electrical activity through the skin, a technique called electrocorticography (or ECoG) [52], and measuring magnetic fields in response to electrical current traveling through

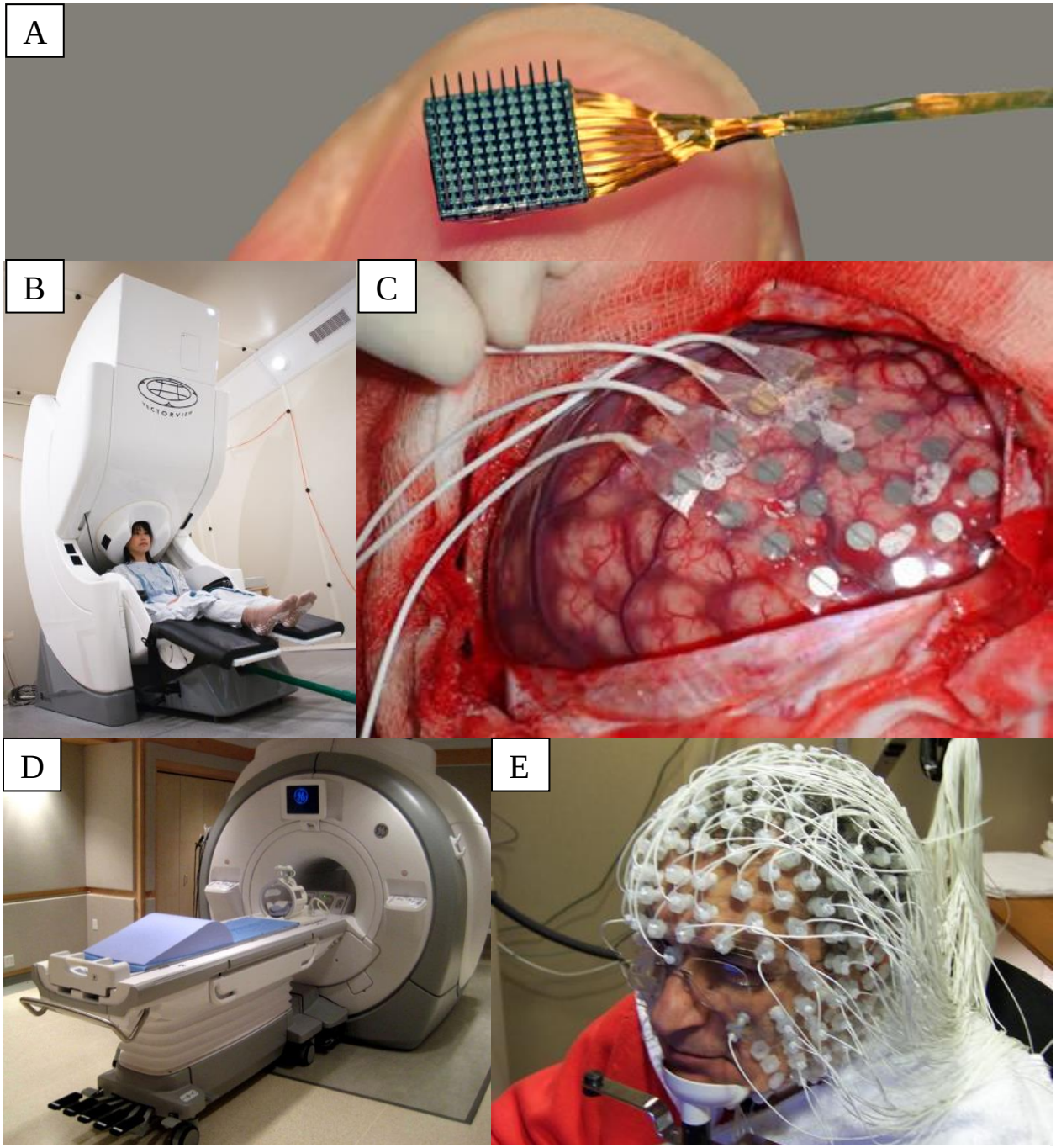

Figure 1.3: Mapping Neural Activity

Part A: Image: [26], A Utah array of penetrating electrodes arranged in a 10x10 mm square grid. **Part B:** Image: [28], Magnetoencephalography [57] measures small changes in the magnetic field caused by electrical current in the brain. **Part C:** Image: [27], Electrocorticography [52] measures field potentials by electrodes which lie on the surface of the brain**. Part D:** Image: [57], An fMRI [61] machine which measures blood flow. **Part E:** Image: [58], A subject wears an EEG (electroencephalography) grid [59] of surface electrodes measuring electrical activity.

the brain, magnetoencephalography (or MEG) [62]. Each method has advantages and disadvantages, but we will focus on an attempt to measure the electrical properties of a population of neurons (second layer of abstraction) to give information about which neurons are firing and when they are active.

Extracellular electrodes are a common method to capture neural activity. These electrodes capture the changing electrical fields associated with action potential generation near neural cell bodies or axons by penetrating the tissue surrounding the neural populations [39] [43]. A reference electrode (placed in a location far from the target recording area) is used to record a differential signal to remove large activity found in both signals, reducing common mode noise. Extracellular electrodes are able to measure disturbances in the local electrical field, meaning they will record very small electrical pulses for neurons relatively far way (greater than 200um) or large electrical pulses for close neurons (closer than 50um) [36] [37]. This means

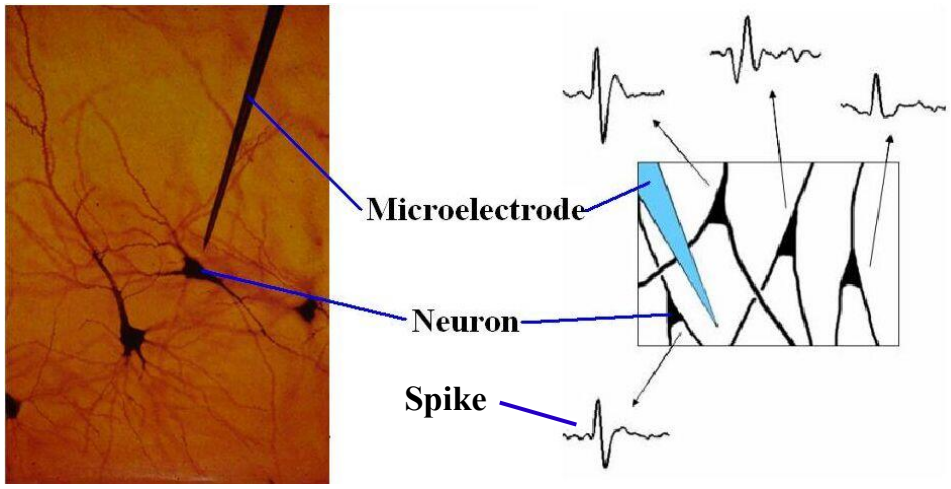

Figure 1.4: Extracellular Electrodes

An electrode measures action potentials elicited from several neurons in its vicinity. Based on the distance from the neuron to the electrode, conductivity of the tissue, and geometry of the cell, each neuron disturbs the electrical field with a slightly different signature. The differences in the disturbance allow us to identify the number of neurons, and when each neuron fires [22] [35].

multiple neurons could contribute to the electrical field measured by a single extracellular electrode.

Since we would like to isolate single unit activity on each electrode in our target recording area, it is important to understand the electrical characteristics of the system we are recording. Pettersen et al. found that neural signal amplitudes decayed at a rate of 1/r^n, where 1<n<2 for nearby cells and n>2 for further away cells [47]. These results were based from analytical models of neurons as current sources in a conductive medium, derived from Maxwell's equations and confirmed with other numerical methods. Due to the linearity of Maxwell's equations, Rall's feed-forward model [46] of the recorded extracellular potential accounts for multiple neurons' electrical activity. This electrical activity is generated by the neuron's action potential electric discharge and post-synaptic electrical activity. The model conveniently allows for the superposition principle to dictate that all neural electrical activity sums linearly [45] when contributing to the extracellular electrical potential. In this generally accepted model, each neuron now contributes a continuous electrical signal, and our recorded extracellular potential is a linear combination of all local neural signals.

Based on the distance from each neuron to the electrode, conductivity of the tissue, and geometry of the cell, each neuron has a slightly different signature waveform [35] [44]. The differences in the waveforms of these electrical pulses (or spikes) allow us to identify the number of neurons, and when each neuron fires. This means our target recording area size is limited by our ability to isolate and identify single unit activity [35] [36] [37]. Since the amplitude of neural electrical signals from cell bodies decays at a rate of 1/r^n (r = distance) [47], we are limited to a spherical volume with an approximate radius of 150-200um from the tip of the electrode (based on our sorting algorithm's discriminatory abilities - *see Chapter 4*). Neurons further away

contribute to background neural noise, known as multi-unit activity, which can be detected when many neurons fire simultaneously; however, in our particular experimental context (*see Chapter 1: Motivation for Spike Sporting in Current Research*) unless the recorded neural activity on our electrodes can be isolated into a single unit activity, this information isn't useful.

1.2 THE PROBLEM: SPIKE SORTING

How do we decompose the summed electrical fields generated by the time varying activity of many neurons at many distances into the contributions of individual neurons close to the electrode tip? This is the problem of spike sorting. Using the data available, the characteristic waveform shapes extracted from the raw waveform, the timing information of spike detection and the complete raw waveform give us our inputs to the spike sorting algorithm. Since we now have a set of numbers representing the waveform and timing information, we can abstract the problem of spike sorting to a multi-class classifier, a problem rigorously studied in computer science and machine learning. We could now use mathematical machine learning techniques, statistical analysis methods or probability based modelers do the classification. However, existing mathematical methods frequently do a very poor job of classifying individual waveforms. The standard in many labs is still to manually spike sort each dataset, which is an extremely time consuming and subjective process [5].

1.2.1 Physical Difficulties with Spike Sorting

Some of the main difficulties with spike sorting result from the volatile nature of electrical recordings. Extracellular electrodes are often implanted as part of arrays containing up to several hundred electrodes. These arrays can move over time (or in conjunction with a moving body part), leading to scar tissue build up around the arrays, which results in signal quality degradation. This means the electrical signals become smaller and smaller, making it more and more difficult to classify the number of unique signature action potentials and harder to distinguish between two (or more) neural waveform shapes.

1.2.2 Mathematical Difficulties with Spike Sorting

Every sorting algorithm has a set of parameters (whether they are exposed to the end user or not) which directly determine the results of the algorithm. These parameters correspond to latent (or observable) variables within the data extracted from the neural signal. For example, a standard machine learning classifier called K-Means [6] has a parameter K, which determines the number of clusters identified in a dataset. The effect of changing this parameter is fairly obvious; the higher K value, the more clusters will be found in the dataset. A more complex example is the parameter “corr_sig”, found in the M-Sorter algorithm [19]. This required parameter, allows the user to specify the minimum “distance” (in the normalized correlation space) between the generated template waveform and an individual waveform when deciding to assign that waveform to a cluster. The results of changing this parameter are not immediately obvious, making it more difficult to estimate. In both cases, these parameters need to be optimally tuned for each individual dataset. While the parameters may be appropriate for the given dataset they

were estimated for, any other datasets in which any number of variables has changed (number of units, number of waveforms, SNR, etc), the classification result is very often wrong.

Many techniques and algorithms have been developed to address all or parts of this problem [1] [2], and these methods range from supervised (completely manual sorting by visual inspection of the waveforms) to semi-supervised (manual inspection of generated clusters) to

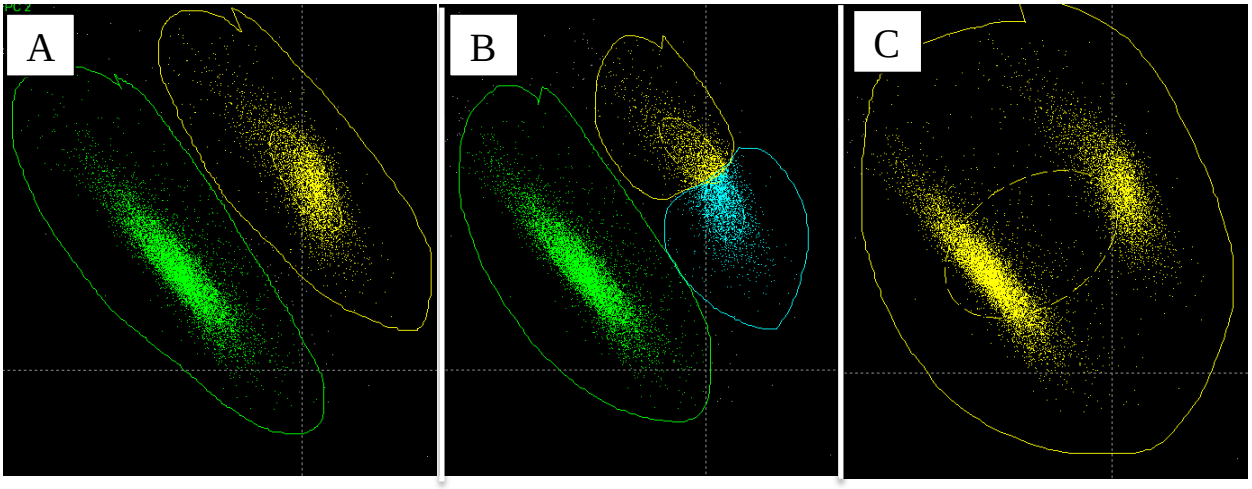

Figure 1.5: Over/Under Sorting

Extracted waveforms from a neural recording (*for methods see Figure 1.1.A-B*) shown PCA space [56]. **Part A:** Shows an accurate reflection of the data showing two clusters (shown in yellow and green). This clustering scheme matches the underlying data well. This is a well sorted dataset. **Part B:** Shows the same dataset with a different clustering scheme. Here, the dataset is over-sorted, split into more clusters than actually exist. **Part C:** Shows an under-sorted clustering scheme. Too few clusters are used to describe the data.

fully unsupervised (online and offline analysis performed solely by a computational algorithm). Manual sorting can be highly variable, based on subjective criterion [4] [5], as well as being time intensive. Often, this method is impractical due to the large quantity of data recorded in an experiment. For completely unsupervised algorithms, assumptions about the parameters of the data have important implications on performance, often leading to poorly sorted data, with waveforms from the same neuron classified as different clusters (**over-sorting**) or waveforms from the multiple neurons classified as same cluster (**under-sorting**) (*see Figure 1.5*). In these

cases, information which could have been extracted is either wrong or lost. This poorly sorted data must then be manual inspected and resorted, effectively losing the advantage of an unsupervised method.

For a truly unsupervised method to perform effectively with a wide variety of data, the algorithm must be able to "learn" the parameters which change the underlying assumptions about the data, adding to the complexity of classification. One classic problem which general algorithms must face is the tradeoff between well isolated clusters and number of clusters. It is often extraordinarily difficult to *a priori* determine how many clusters are present, since this number is highly sensitive to the specific physical location and geometry of neurons near the extracellular electrode, which can change drastically from dataset to dataset. To return to an earlier example, the K-Means algorithm [6] has a parameter *K*, which fixes the number of clusters discovered by the algorithm, and some initial starting point of each of the *K* clusters. If *K* (or the initial cluster centers) is incorrectly set, K-Means can fall into a local minimum and will not converge to the global optima (defined as the lowest possible negative log likelihood of the classification results). For a more nuanced example, one can imagine an algorithm which separates the data into Gaussian clusters, each with a mean and covariance in some high dimensional space. This algorithm may have a parameter to determine the maximum allowed difference between the estimated parameters and the ideal values (as decided by the algorithm) of a cluster. It could decide to split this cluster in half once the difference exceeds some threshold, until all the data is classified into Gaussian clusters. The number of clusters this algorithm will find is highly dependent on this tunable parameter and correctly setting the parameter would be essential to achieving good performance.

1.3 PREVIOUS SPIKE SORTING METHODS

Some sort quality metrics, used to judge the accuracy of parameter estimation for other spike sorting methods, have been previously developed [7] [4] [8] [30]; however, most are based on measures in a low dimensional representation of the data (such as principle component (PCA) [56] space or some other dimensionality reduction kernel) or applicable only to a particular sorting algorithm's feature space. For instance, a metric which determined unit isolation by Euclidean distance between those two units in PCA space only makes sense if the algorithm sorts the data in the PCA space. Obviously, dimensionality reduction algorithms inherently discard information deemed to be unimportant or not as important as other information. While necessary in some applications, it is difficult to avoid discarding useful information when the variability of the datasets is high. Some investigators have suggested manual inspection of sorts to determine quality [9] [10]; these are highly subjective, time intensive, prone to human error and have poor inter-rate repeatability.

1.4 MOTIVATION FOR SPIKE SORTING IN CURRENT RESEARCH

In this current research, we are recording the activity of sensory and motor neurons in the spinal roots, which provide a connection from the spinal cord to the actuators (muscles) and sensors (muscle spindles, golgi tendon organs, cutaneous receptors) in the hind limb of cats. The purpose for studying and understanding this neural network is for the eventual development of a brain computer/machine interface (or spinal cord interface) (*see Figure 1.6*). We would like to interface with the spinal cord to decode intent (i.e. to move a motor prosthetic) and communicate

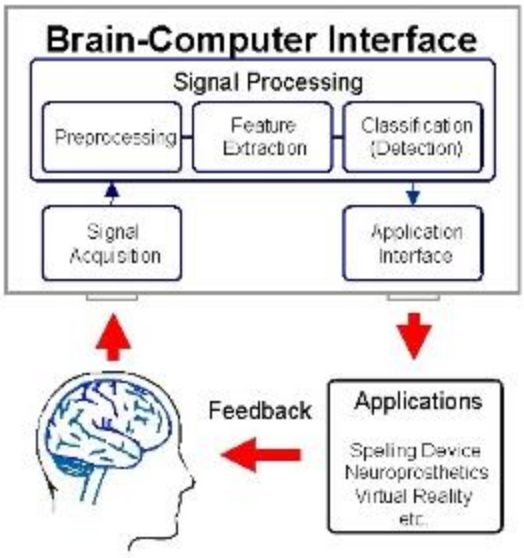

Figure 1.6: A Brain Machine/Computer Interface

The machine records neural activity, does some signal processing, and figures a command to be sent to the application (a robotic arm, computer game, wheel chair, etc) while the user can gain feedback of the signals "sent" through visual or tactile means. [29]

sensory feedback (i.e. position of that motor prosthetic), by capturing all the network events and attempt to correlate its activity with some motor, sensory and/or thought pattern. These spinal roots have little to no somatotopical organization [48], and therefore it is essential that we are able to distinguish between the different neural activities on a single electrode. We often implant a Utah array [41] a 10x10 grid of uniform length electrodes, spaced at a distance of 400um from each other, greatly reducing the likelihood of two electrodes recording the same neuron [47].

The depth of granularity when examining a neural population is context specific. For example, for BCIs recording from the motor cortex from monkeys or humans, there is good reason to believe that the entire population, whose activity is captured by a Utah array (or other array), is responding similarly to the same inputs, due to the somatotopical organization of the motor cortex [63]. The population of neurons is quite small with respect to the total group of neurons contributing to making an arm or leg move. In this case, it may be fine to abstract the

group of neurons recorded by a single electrode to a single entity with one firing rate. The brain has so many different groups of neurons, for an experiment examining many parts of a complex network, it might be impossible (or impractical) to process such detail. However, for areas with little or no known somatotopical organization such as the Dorsal Root Ganglion, or DRG, (*see Figure 1.7*) [48], it is absolutely necessary to separate individual neuron activity, since two neurons side by side could respond completely opposite to the same input. For example, if we recorded activity from two neurons which each controlled opposing muscles, any analysis assuming the activity from these neurons came from the same neuron could lead to drastically erroneous results.

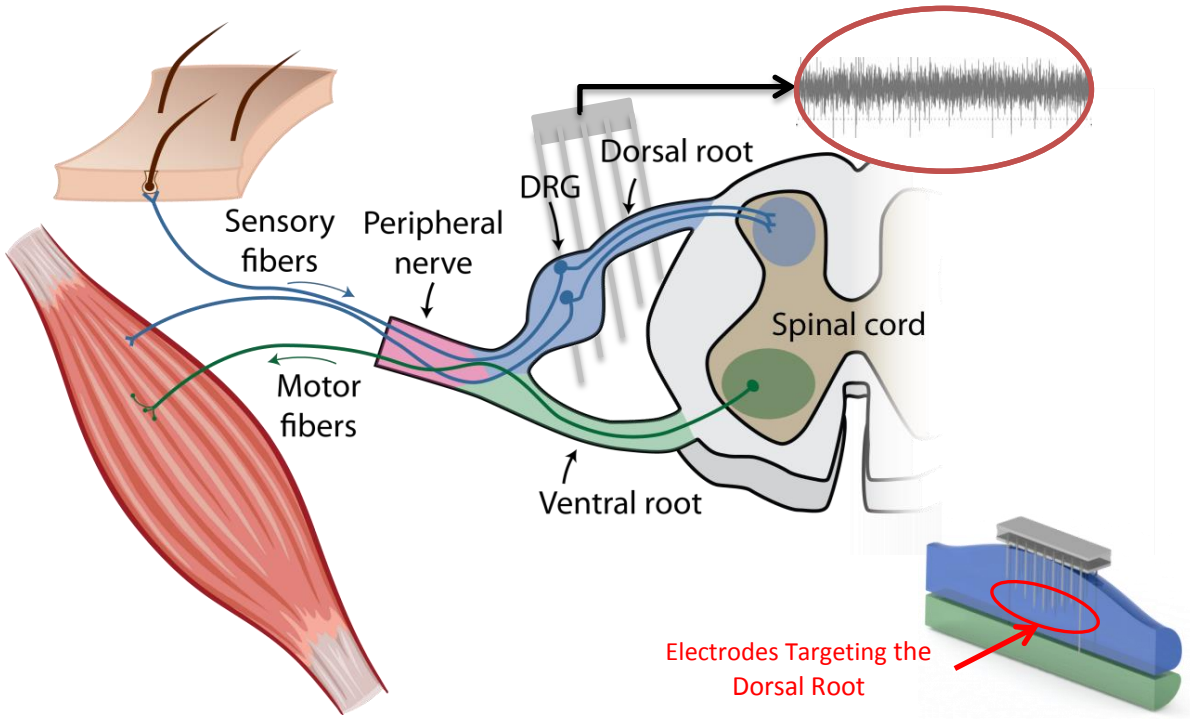

Figure 1.7: Experimental Setup

We target the Dorsal Root Ganglion (DRG) which contains cell bodies relaying sensory information back to the brain and spinal cord. The ventral root contains axons for neurons transmitting motor commands to muscles in the hind limb of a feline. These roots present a rare opportunity where the motor and sensory bundles are split. Any further down in the leg and these neurons are interspersed.

In order to understand the workings of the leg, how it is controlled, what information is being sent back to the brain, what neurons are responsible for relaying the output of the electromechanical sensors in the leg, back to the brain, and other questions, we need to map the activity of the root neural network while the leg is moving and experiencing sensory stimuli. We implant felines chronically and perform controlled behavioral experiments to provide an input to the neural population. Our targets are the cell bodies in the DRG (*see Figure 1.7*) which pass sensory information back to the brain and spinal cord. Another neural population we record is located in the ventral root, containing axons from motor neurons controlling the muscles in the leg. By recording spikes from both these populations, we can see how the networks respond under various behavioral conditions.

To properly map the neural activity we have recorded, we propose an algorithm, Heuristic Spike Sort Tuner (or HSST) to provide a validation score, generalizable to any sorting algorithm, for the comparison between parameters sets of a sorting algorithm, to iteratively select the optimal parameter set in an offline algorithm, or to initialize parameters for an online algorithm. This algorithm will classify the incoming neural information into unique groups (representing the unknown number of distinct neural input(s)), thus giving us the timing information of each neuron's fire activity. By using many heuristics developed to mimic human understanding of accurately clustered neural recordings of electrical potentials, we can combine many weak metrics which examine some features of the recorded data, contributing to strong classifier [11]. HSST will score a data set based on a focused criterion (i.e. signal-to-noise ratio (SNR), inter-spike interval (ISI) violations, similarity of waveform shape, and others). Using the generated score, this classifier will determine the best sort results of an algorithm from a given set of parameters.

2.0 BACKGROUND OF METHODS

Many machine learning techniques and statistical measures were used in the metrics to evaluate some feature space of the data, and thus we provide a brief overview of these concepts to give the reader some familiarity with the technical tools used in the algorithm. These tools are the building blocks on which the algorithm was built.

2.1 MULTI-MODALITY FUNCTION

Often, when examining a feature space, it is helpful to determine the number of modes in that feature space, by generating a statistical distribution of values (through some analysis of that feature space) and determining the number of peaks. To simplify this calculation, a standard way of detecting bimodal (or multi-modal) data was determined. First a histogram of the raw data values is generated, and normalized such that the total area of the histogram equaled 1, and smoothed with a moving average window. Insignificant bins (those contributing less than 1% of bin area) trailing and leading are removed from analysis.

All peak/valley pairs are identified and normalized between the maximum peak amplitude and the 1% significant bin area threshold. Any peak/valley distance less than 25% is considered insignificant, such than only peak/valley distances greater are candidates for multiple

modes in the data. Any valley discovered with both sides greater than 40%, means the data is not uni-modal. For valleys with sides between the 25% and 40% distance, a Gaussian mixture model is generated using the number of local peaks as the number of components. If more than one local peak is found in the combined distribution, the data is not uni-modal. Figure 2.1 illustrates the individual steps to detect multi-modality within a feature space of a dataset.

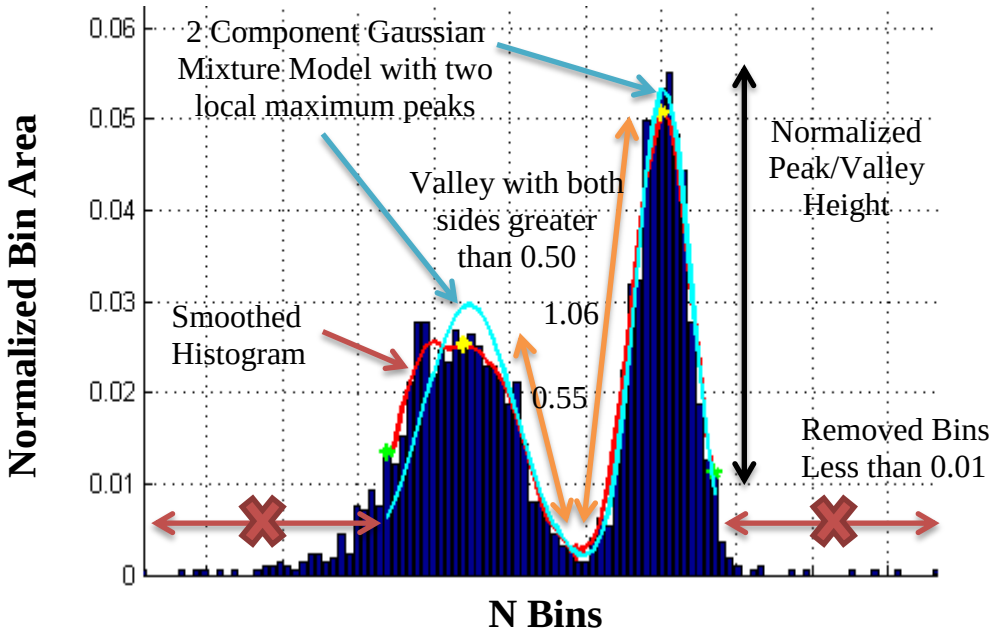

Figure 2.1: Multi-Modality Detector

Histogram is has N bins (N is the square root of the number of data points [16]). Histogram is normalized by the sum of the bin area. A smoothed histogram generated by choosing each bin value as the median of the bin before and after it [17]; the output is filtered by a moving average (window size = 5 bins). All preceding and trailing bins less than 1 percent are removed from analysis. A Gaussian Mixture Model is generated with the number of components equal to the number of local max peaks of the smoothed histogram, and if the resulting distribution has more than one local maximum peak, data is not uni-modal.

2.2 EXPECTATION MAXIMIZATION ALGORITHM

The EM algorithm [31] is a well-known and studied machine learning classification tool. It is an iterative method which relies on a 2 step process (see *Figure 2.*2) for determining classification of a data set. The expectation (or 'E') step provides the log-likelihood function of the internal model given the current parameters, while the maximization step (or 'M') step, updates the parameters to maximize that log-likelihood function (see Equations 2.1,2,3). The Gaussian Mixture Model [33] is an excellent example of an implementation of this algorithm, which uses a Gaussian distribution as the internal model. The parameters updated are the mean and covariance matrix. K-Means [6], a special case of the Gaussian Mixture Model, has equal

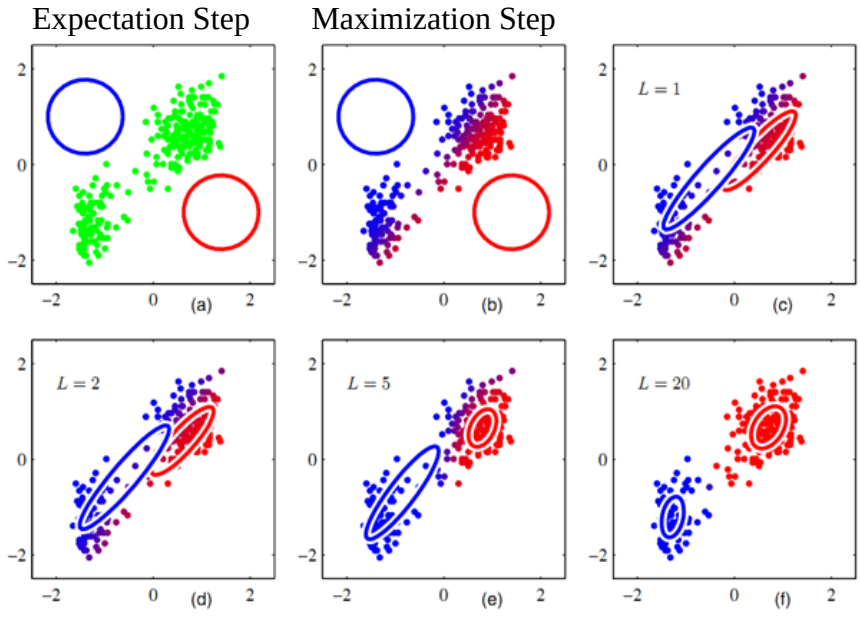

Figure 2.2: Gaussian Mixture Model

At each iteration, two steps are performed, illustrated by the first two panes. An Expectation Step determines the means and covariance of the clusters (shown as red/blue ellipses in the figures above). A Maximization step assigns each element of the data to maximize the log-likelihood of the current cluster locations. These steps are repeated until the log-likelihood converges. [49]

fixed covariance matrices or each cluster. Each element in the covariance matrix is equal and it has only has non-zero elements on the diagonal.

$$p(x|\theta) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i) \text{ where } \theta = \{w_i, \mu_i, \Sigma_i\} \text{ and } i = 1, \dots, M$$

Equation 2.1: GMM – Probability Function

The probability of x, given θ parameters is equal to the sum of each multivariate Gaussian component weighted by some w_i vector, whose values all sum to 1.

$$\log\big(L(\theta|X)\big) = log \prod_{j=1}^{N} p(x_j|\theta) = \sum_{j=1}^{N} log\left(\sum_{i=1}^{M} w_i g\big(x_j|\mu_i, \Sigma_i\big)\right)$$

Equation 2.2: GMM – Log-Likelihood Function

The log-likelihood function is maximized with respect to each variable in θ to compute the update laws for each variable. Once the log-likelihood value converges (it is computed at each E step), GMM stops updating.

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left\{-\frac{(x-\mu_i)^T {\Sigma_i}^{-1}(x-\mu_i)}{2}\right\}$$

Equation 2.3: GMM – Multi-Variate Gaussian Probability

x is a D dimensional vector of observations, μ_i is the mean vector, Σ_i is the covariance matrix.

2.3 SIGNAL TO NOISE RATIO DEFINITION

The signal-to-noise ratio is often discussed when comparing the power of a transmitted (or detectable) signal to the power of background noise, which could possibly corrupt or interfere with the desired signal. In biological terms, we discuss signal-to-noise ratio as a measure of discriminability between the electrical activities of nearby neurons and far away ("background")

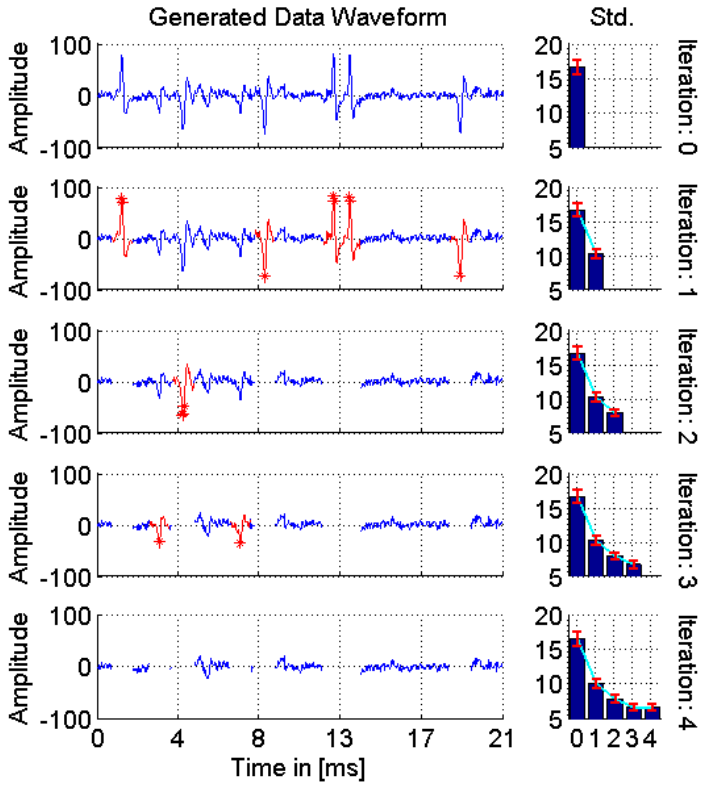

Figure 2.3: Iterative Noise Floor Estimation

Iteration 0 shows the raw generated waveform. During each iteration, the standard deviation is estimated. The red dots indicate data points which lay more than four standard deviations from the mean of the entire waveform (shown in blue). One millisecond of the waveform is removed about that point (shown in red), and the next iteration estimates the updated distribution and removes more snips. This continues until the estimate for the standard deviation of the noise converges (shown on the right hand panels).

neurons. This signal-to-noise ratio can be determined many ways; however, we have chosen to define it as the ratio of mean amplitude of the signal to the mean amplitude of the noise.

We calculate the average noise amplitude using an iterative method. The entirety of our recorded waveform is modeled as a Gaussian distribution. Iteratively, we remove waveform snippets (1 millisecond around any point(s)) which are greater than 4 standard deviations from the mean of our Gaussian fit. These outliers which distort the empirically measured distribution

are iteratively removed until the estimated standard deviation converges (*see Figure 2.3*). Signal is measured as the average absolute peak amplitude of the removed waveform snippets.

2.4 D' (SENSITIVITY INDEX)

The Sensitivity Index (also called D' – pronounced 'dee-prime'), is a measure of isolation between two Gaussian distributions [15]. It provides a decimal value (ranging from 0 to positive infinity). A value of less than one is an indicator of significantly overlapping Gaussian distributions. It can also be calculated using the hit rate and false alarm rate of a binary classification problem [34].

$$X_S = [x_s(1), x_s(2), \dots x_s(N)], \quad X_N = [x_N(1), x_N(2), \dots x_N(M)]$$

$$d(\theta)' = \frac{\mu_S - \mu_N}{\sqrt{\frac{1}{2}(\sigma_S^2 - \sigma_N^2)}} \text{ where } \theta = \{X_S, X_N\}$$

$$\mu_S = mean(X_s),\ \sigma_S = var(X_s)^{1/2},\ \mu_N = mean(X_N),\ \sigma_N = var(X_N)^{1/2}$$

Equation 2.4: The D' Statistic (or Sensitivity Index)

Symbols μ_S and μ_N are the mean of the signal and noise distribution, respectively. σ_S and σ_N are the first standard deviation of the signal and noise distributions.

2.5 PRICINPLE COMPONENTS ANALYSIS (PCA)

Principle Component Analysis [56] is an unsupervised algorithm which is often used as a dimensionality reduction technique to examine high dimensional data. Mathematically, it uses an

orthogonal transformation to covert a set of possibly correlated observations into a set of linearly uncorrelated vectors. From linear algebra, we can describe these vectors as eigenvectors of the matrix 'X', containing all our observations. Simply put, PCA identifies a set of vectors describing the dimensions of maximum variability in a high dimensional space; it ranks those orthogonal dimensions according to vectors describing the most variability, to those describing the least variability. These vectors are also known as principle components.

For example, in Figure 2.4, the black dots represent our observations in a 2 dimensional space (represented by $f(E_1)$ and $f(E_2)$). The vector describing maximum variability of data is show as eigenvector 1. Eigenvector 2 shows an orthogonal vector describing the dimension of next greatest variability. (Since we have only 2 dimensions, eigenvector 2 is the only possible orthogonal vector; however in less trivial cases, all possible orthogonal vectors to eigenvector 1 are examined). If we were to plot the data in PCA space, we would transform our observations (black dots) into another coordinate space along the eigenvectors shown. Due to limitations of singular value decomposition, we cannot identify more dimensions than number of dimensions of our observations. Our observational data points only have 2 dimensions, so we can only have 2 principle components.

Principle component analysis is widely used in many fields where it similar to these other analyses: the discrete Karhunen–Loève transform (KLT) [66], singular value decomposition (SVD) [65], eigenvalue decomposition (EVD), factor analysis and many others.

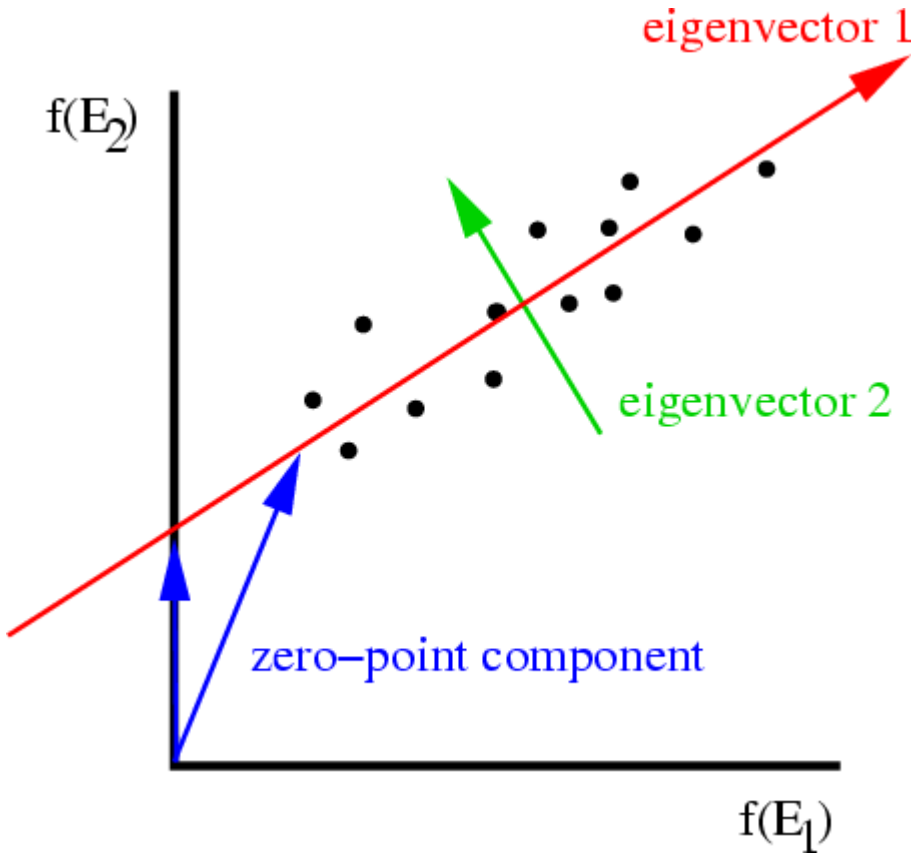

Figure 2.4: Principle Component Analysis

Image: [55], We see our observation data (black dots) shown in 2 dimensional space. PCA has identified an orthogonal rotation of these axes such that the data variability is maximum along the axis of specified by the principle components. If the data points are expressed as a matrix 'X' of observations, we can identify this rotation through singular value decomposition to identify the eigenvectors of the matrix X.

2.6 NORMALIZED MEAN SQUARED ERROR

Normalized Mean Squared Error is a measure of the "goodness" of fit between a probability distribution and some empirically measured observations. We use this to see how closely some of our probabilistic models fit some feature space within our data.

$$\hat{Y} = [\hat{y}_1, \hat{y}_2, \ldots \hat{y}_N], \;\; Y = [y_1, y_2, \ldots y_N]$$

$$NMSE(\theta) = \frac{MSE}{y_{max} - y_{\min}} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}}{y_{max} - y_{\min}} \quad \text{where } \theta = \{\hat{Y}, \; Y\}$$

Equation 2.5: Normalized Mean Squared Error

Symbol ŷ is the estimated value; y is the true value; N is the number of samples; y_{max} and y_{min} are the maximum and the minimum value of y. NMSE is often expressed as a percent.

3.0 METHODS

Spike sorting is a multi-tiered process roughly broken down into three steps: 1) detection, 2) extraction, and 3) clustering [12]. One of the goals of this study was to develop a method that could be generalized to a variety of spike sorting approaches, all working in any feature space. As such, anyone can insert their favorite algorithm for detection and/or sorting, and the Heuristic Spike Sort Tuner (HSST) (*see Figure 3.1*) can be used to identify the best set of free parameters of that given sorting algorithm.

3.1 TERMINOLGY

Before we describe the HSST algorithm, we need to clarity a few ambiguous terms. A **dataset** refers to a group of waveforms, which have been extracted from some **raw waveform** at the point of spike detection and whose features are captured in some N-dimensional space (*see Figure 1.1*). A **sort** refers to a classification scheme assigned to dataset. That classification scheme assigns each waveform an integer identification number (referring to the source neuron), determined by the output of a sorting algorithm whose inputs include a set of parameters. A **unit**, a subset of a sort, refers to the group of waveforms with the same classification identification number. One waveform cannot belong to two units; it can only be assigned a single classification

identification number. A dataset will have multiple sorts, each generated by a different parameter set.

3.2 HSST ALGORITHM OVERVIEW

The HSST algorithm is designed be as general as possible with regard to the spike detection and clustering algorithms used. This maximizes its potential use and feasibility of integration with existing systems. As such, HSST was designed to work with any detection and feature extraction/clustering algorithm which relies on tunable parameters.

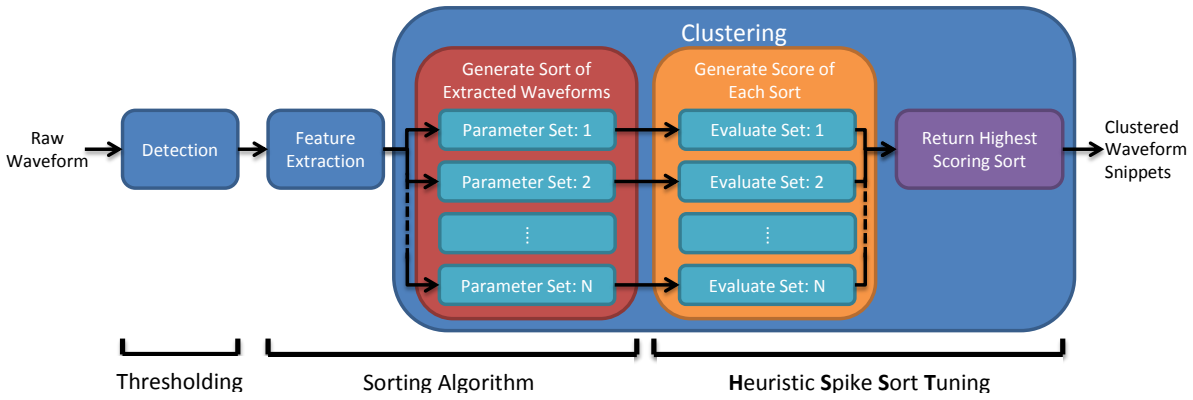

Figure 3.1: Block Diagram of HSST Algorithm

The blue blocks illustrate the three generic steps, while the black labels at the bottom show the specific technique used in our sorting algorithm.

In interest of simplicity, for this study, a basic amplitude threshold technique detected spike times. Similarly, a simple K-Means sorting algorithm clustered spikes, extracted from the continuous recorded waveform at the spike detection times. The true advantage of the HSST algorithm comes from the autonomous parameter selection which allows even a simple K-Means algorithm to be used at maximum effectiveness across many datasets with widely varying SNR

values and changing number of neurons. Our HSST algorithm is the key to determining which parameters yield an accurate result.

3.2.1 Sort Score Generation

The HSST algorithm returns a **sort score** for each sort in a dataset, evaluating each sort on the basis of several metrics which help determine the neurophysiological feasibility of the given sort. The highest scoring sort is selected as the optimal way to classify the dataset and classification scheme becomes the output of HSST.

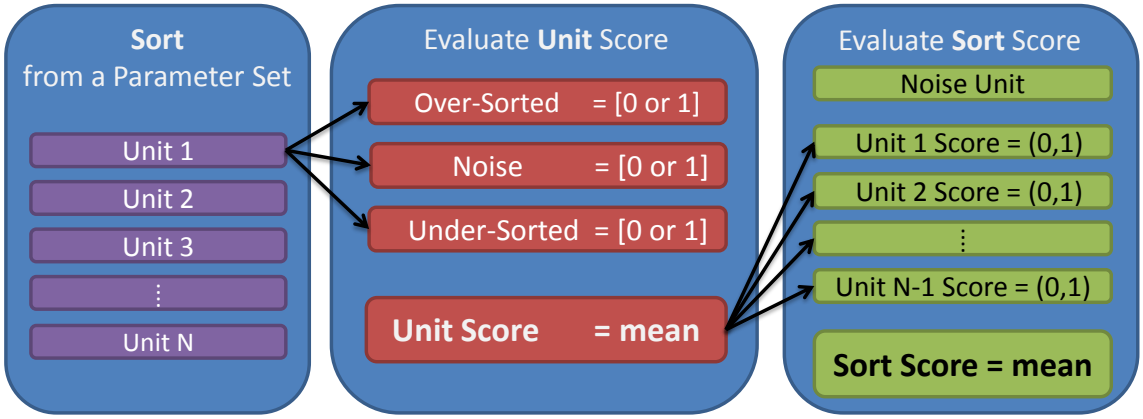

Figure 3.2: Block Diagram of Sort Score Generation

Each Unit of the sort receives an individual score. The average of these unit scores results in the sort score. If identified, a single noise unit will removed from the average.

The **Sort Score** is generated from each of the unit scores in that sort. The **Unit Scores** are based on whether that unit is Over-Sorted, Under-Sorted, or Noise. Each unit receives a decimal score ranging from 0 (lowest sort quality) to 1 (highest sort quality). The **Sort Score** (also ranging from 0 to 1) is the weighted average of all the **unit scores** (weighted by the number of waveforms in that unit). If a noise unit within a sort is identified, the noise unit's score is not

included in the final average (*see Figure 3.2*). If multiple units are labeled noise units, only the noise unit with the lowest score is removed from the final average.

3.2.2 Unit Score Generation

In order to generate the **Sort Score**, each unit within a sort must be given a score. The overarching goal of our HSST method is to accurately classify and separate individual neural activity in a single dataset (which can contain neural activity from multiple neurons). Each **unit score** reflects whether the unit contains information from a single neuron. To make these judgments, we need to identify three failure modes of a clustered unit, **under-sorted** units (a single unit which contains information from multiple neurons) (*see Figure 1.5.C*), **over-sorted** units (when the information from a single neuron is spread between two or more units) (*see Figure 1.5.B*), and **noise** units (units with irrelevant data). If a unit does not fall into one of these three categories, it is a well sorted unit. We use several metrics (*see Figure 3.3*) to help us determine which category the unit belongs in.

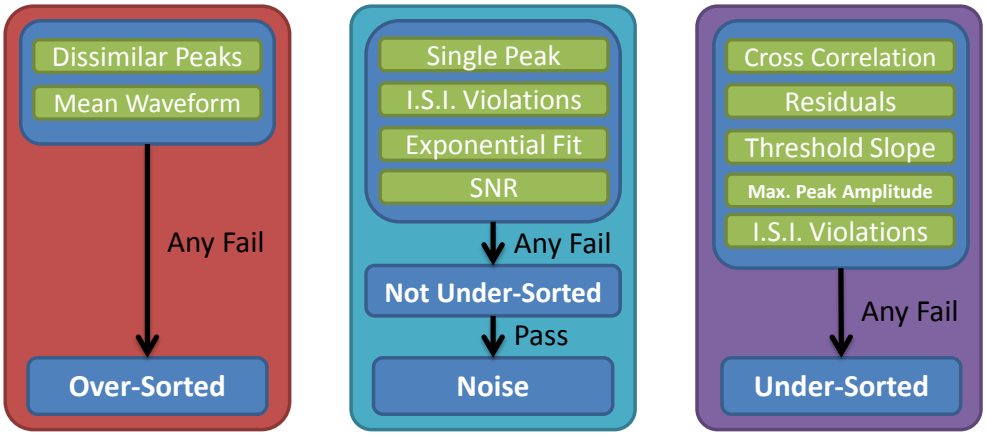

Figure 3.3: Block Diagram of Unit Score Generation

Many heuristical metrics are used to determine the category values of **over-sorting**, **under-sorting**, and **noise.** The diagram above shows the Boolean logic used to determine each category value.

3.2.3 Generated Example of HSST Parameter Selection

In Figure 3.4, we show a generated example of the HSST method. For each sort while K = 2, 4, or 6, the units have been scored according to the output of the metrics. These scores contribute to the Sort Scores shown at the bottom of the figure. The sort with the highest score is most similar to the known result on the far left.

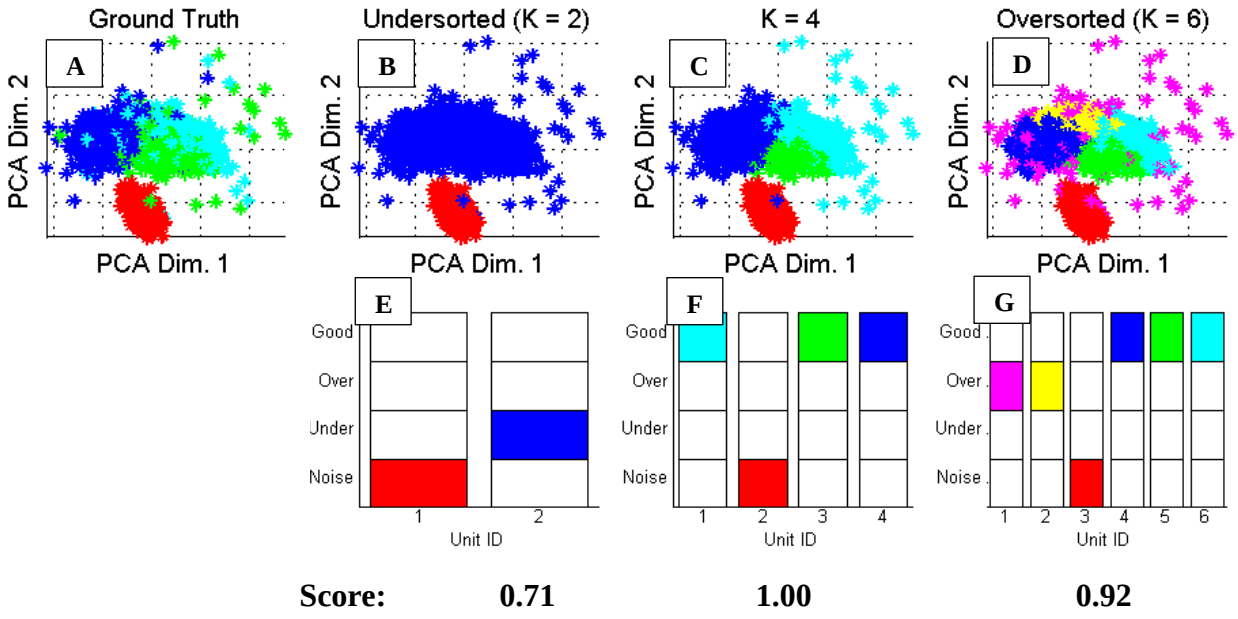

Figure 3.4: Illustration of Iterations of HSST using Generated Dataset

See *Chapter 4: Datasets* for how data was generated. **Part A:** The "Ground Truth" PCA scatter plot. **Part B-D:** Results of the Gaussian Mixture Model sort with the K parameter set to 2, 4, and 6. **Part E:** Colors correspond to unit in the PCA plots. Colored boxes show the unit status (Good, Over-sorted, Under-sorted, Noise) according to HSST's Unit Score. This sort is under-sorted. **Part F:** We see the noise cluster (Unit 2), and three good units. This sort is the most similar to the known result and scores the highest. **Part G:** This result is over-sorted. As a result, Units 2 and 4 are overlapping, and Unit 1 overlaps with all the units. Unit 3 is correctly identified as noise.

3.3 METRICS

We have developed a series of metrics to determine the **unit score**, representing the sort quality of a unit. These metrics are explained in detail below, but briefly, to identify under-sorting we quantify the residuals when the mean waveform is subtracted from individual snippets and examine a similarity distribution to identify units with multiple peaks. To identify over-sorting, we quantify the degree of separation between the minimum and maximum peaks of sorted units to quantify similarity in their shape.

3.3.1 Noise Metrics

These metrics are used to identify multiple unit clusters which are background noise (either neural data whose amplitude makes it impossible to confidently separate from other neural data or artificial measurement noise).

3.3.1.a Signal to Noise Ratio (SNR)

Since our primary objective is to determine well isolated neural activity, using the signal to noise ratio can be a discriminating feature in the data. We calculate the noise floor from the raw waveform by modeling the entirety of our recorded waveform as a Gaussian distribution and iteratively removing waveform snippets which distort the empirically measured distribution until the estimated standard deviation converges (*see Figure 2.3, Background of Methods*). If raw data is not available, the user can provide an estimate of the noise. Signal is measured as the average absolute peak amplitude of the removed waveform snippets. If the average signal to noise ratio of a unit is above a threshold value (determined by the user), the unit passes this metric.

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points}\ n = 1, \dots, N$$

$$SNR_j = \frac{\frac{1}{M}\sum_{i=1}^{M} \max\left(abs\left(f_{ij}\right)\right)}{\sigma_{noise\_estimate}}$$

Equation 3.1: Signal-to-Noise Ratio

SNR_j is the SNR value of unit j, which contains M snippets. $f_i(n)$ is a vector (N sample points) of values representing the waveform snippet i in unit j. $\sigma_{noise\_estimate}$ is the estimated noise level (see *Figure 2.3, Background of Methods*).

3.3.1.b Inter-Spike Intervals (ISI)

A count of inter-spike interval (ISI) refractory violations is a reliable metric for a poorly sorted unit. Biologically, most types of neuron have absolute refractory periods greater than one millisecond, meaning a neuron cannot fire at a rate greater than 1 kHz. In practice, neuronal firing rates are much lower, since many types of neurons have a relative refractory periods ranging from 5-20 milliseconds.

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points}\ n = 1, \dots, N$$

$$r_j(i) = recorded\ time\ [ms]\ of\ f_{ij}(n = 1) \quad \textbf{(1)}$$

$$a_j(i) = \begin{cases} 1 \text{ iff } r_j(i) - r_j(i+1) < 1ms \\ 0 \text{ iff } r_j(i) - r_j(i+1) > 1ms \end{cases} \quad \textbf{(2)}$$

$$ISI\_VIOL\_PCNT_j = \frac{\sum_{i=1}^{M-1} a_j(i)}{M-1} \quad \textbf{(3)}$$

$$Unit\ j's\ Metric\ Score = \begin{cases} 1 \text{ iff } ISI\_VIOL\_PCNT_j > 5\% \\ 0 \text{ iff } ISI\_VIOL\_PCNT_j \leq 5\% \end{cases} \quad \textbf{(4)}$$

Equation 3.2: ISI Violations

Eqn. 1 shows the time information of a recorded snippet. If the number of ISI values which are less than 1 millisecond (Eqn. 2), is less than 5% (Eqn. 3), the unit j passes the metric.

For our study with generated data, we assumed our generated neural activity fired at a random rate less than 1 kHz. If a unit had more than 5% of its spikes occurring within 1 millisecond of another detected spike, the unit failed this metric (*see Figure 3.5*, *Equation 3.2*).

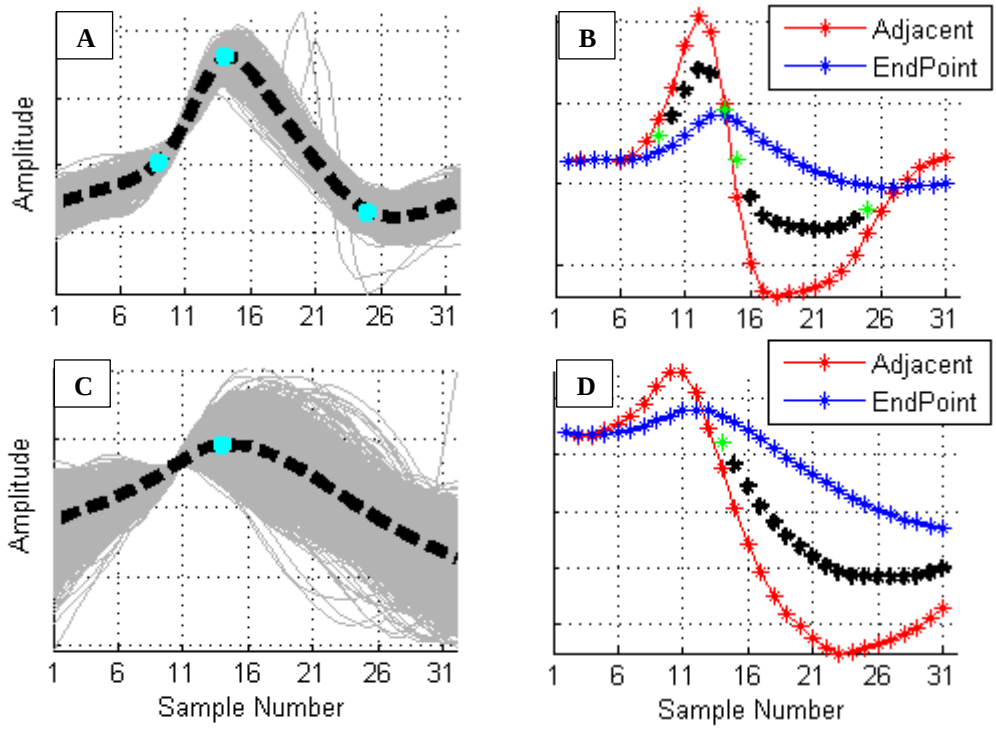

Figure 3.5: Noise Metrics: Single Shoulder

Part A: Shoulders are shown as blue dots on a mean waveform (dotted black line - see *Equation 3.1.1*). **Part B**: These "Shoulders" can be identified by the crossings of the derivative curves. The **Adjacent** derivative (see *Equation 3.1.2*) is slope between sample point n and sample point n+1. The **Endpoint** derivative (see *Equation 3.1.3*) is the slope between sample point 1, and sample point n. Black dots represent σ_{E_Sig} (n) = 1, (*see Equation 3.1.5*). Crossings represent shoulder points. Crossings only count before and after two or more consecutive black dots. If two or more shoulder points are detected, the metric passes. **Part C-D**: An example showing only a single shoulder point detected.

3.3.1.c Single Shoulder

The neural waveforms we are recording are always tri or biphasic. If we identify a unit with a single "bend" or "shoulder" in the mean waveform, it is likely a noise unit, not neural signal (*see Figure 3.5*).

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points } n = 1, \dots, N$$

$$F_j(n) = \frac{1}{M}\sum_{i=1}^{M} f_{ij}(n) \quad \textbf{(1)}$$

$$G_j(n) = F_j(n) - F_j(n+1) \text{ for } n = 1, \dots, N-1 \quad \textbf{(2)}$$

$$H_j(n) = F_j(1) - F_j(n) \quad \text{for } n = 1, \dots, N-1 \quad \textbf{(3)}$$

$$\sigma_E(n) = var(E(n))^{1/2} \text{ where } E(n) = |G_j(n) - H_j(n)| \quad \textbf{(4)}$$

$$\sigma_{E\_sig}(n) = \begin{cases} 1 & \text{iff } \sigma_E(n) > 1 \\ 0 & \text{iff } \sigma_E(n) \leq 1 \end{cases} \quad \textbf{(5)}$$

Equation 3.3: Single Shoulder

The mean waveform of unit j is calculated (Eqn. 1). The derivative of the mean waveform is Eqn. 2. The slope from each point of the mean waveform, with respect to the first sample point of the mean waveform is calculated (Eqn. 3). The standard deviation of the absolute difference between Eqn. 2 and Eqn. 3 is Eqn. 4.

3.3.1.d Inter-Spike Interval Exponential Fit

If we assume the underlying noise is a normally distributed random process, we can detect noise units by identifying characteristics of the ISI histogram of random generated noise. We want to identify any sources of noise in our data (if they exist) and properly isolate them from the rest of the units. If we threshold a normally distributed random signal at an arbitrary threshold, the histogram of times between the points above that threshold can be approximated by an exponential curve. Therefore if a unit's ISI histogram fits an exponential curve with a normalized mean squared error (NMSE) (*see Equation 2.5*) rate of 15% or less, that unit fails this metric.

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points } n = 1, \dots, N$$

$$r_j(i) = recorded\ time\ [ms]\ of\ f_{ij}(n = 1) \qquad R_j = [r_j(1), r_j(2), \dots, r_j(M)] \quad \textbf{(1)}$$

$$hist_j = histogram(R_j) \quad \text{binned at } [0,1,2,3 \dots .100]ms \quad \textbf{(2)}$$

$$expFit_j = exponetial\_fit\big(\ hist_j\ \big) \quad \textbf{(3)}$$

$$Unit\ j's\ Metric\ Score = \begin{cases} 1 \text{ iff } NMSE\big(\ hist_j, expFit_j\big) > 15\% \\ 0 \text{ iff } NMSE\big(\ hist_j, expFit_j\big) \leq 15\% \end{cases} \quad \textbf{(4)}$$

Equation 3.4: ISI Exponential Fit

The Normalize Mean Squared Error (see *Equation 2.5)* is calculated between the exponential fit (Eqn. 3) and the histogram of the distribution of ISI values ranging from 0 to 100 milliseconds (Eqn. 2). If the NMSE is less than 15%, the metric fails (Eqn. 4).

3.3.2 Over-sorting Metrics

These metrics are used to identify two units which contain neural information from the same neuron. We want a single unit to contain all the information from a single neuron, so if a neural information from a single neuron is split between two units; the sorting algorithm has made an error.

3.3.2.a Statistically Similar Peaks

If a single neuron's firing activity has been separated into two different units, we would expect similarity between the mean shapes of those two units. Specifically, if we examine the distribution of minimum and maximum peak waveform amplitudes, we should see a significant

overlap. This metric does a pairwise comparison between units, to find overlapping min/max peak distributions (*see Figure 3.6*). To measure the strength of the overlap, we use the D' statistic (or sensitivity index) (*see Equation 3.5.3*) [15] to compute the distance between the distributions (*see Equation 3.5*). If the D' statistic is less than one, then the peaks are considered overlapping. In order for two units to be considered overlapping, both of their minimum and maximum peak distributions must overlap. If a unit overlaps with any other unit, both units both fail this metric test.

Comparing peak amplitudes gives spatial information about the two units; however, if two peaks are at the same amplitude but occur at vastly different sample points, those waveforms are not similar. Therefore temporal information is captured by finding the average sample point of the minimum and maximum peak distribution. If the peaks fall within 0.125 milliseconds (2-3 sample points) of each other, then the peaks are confirmed to overlap. If the peaks fall outside of that window, the peaks are considered non-overlapping (*see Equation 3.5*).

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points } n = 1, \dots, N$$

$$a_{ij} = \max(f_{ij}), \qquad A_j = [a_{1j}, a_{2j}, \dots, a_{Mj}] \quad \textbf{(1)}$$

$$b_{ij} = \min(f_{ij}), \qquad B_j = [b_{1j}, b_{2j}, \dots, b_{Mj}] \quad \textbf{(2)}$$

$$Unit\ j's\ Metric\ Score = \begin{cases} 1 \text{ iff } d(A_j, A_k)' > 1 \text{ or } d(B_j, B_k)' > 1 \text{ for all } k \neq j \\ 0 \text{ iff } d(A_j, A_k)' < 1 \text{ and } d(B_j, B_k)' < 1 \text{ for any } k \neq j \end{cases} \quad \textbf{(3)}$$

Equation 3.5: Overlapping Peak Distributions

To calculate the metric score of unit j, we look at each waveform snippet in unit j. We form two distributions,: the maximum value of each waveform (Eqn. 1) and the minimum value of each waveform (Eqn. 2). We pairwise compare the d' distance (*see Chapter 2: D' (Sensitivity Index)*) between unit j and all other units. Metric fails if overlap with both minimum and maximum peaks is found with another unit k (Eqn. 3).

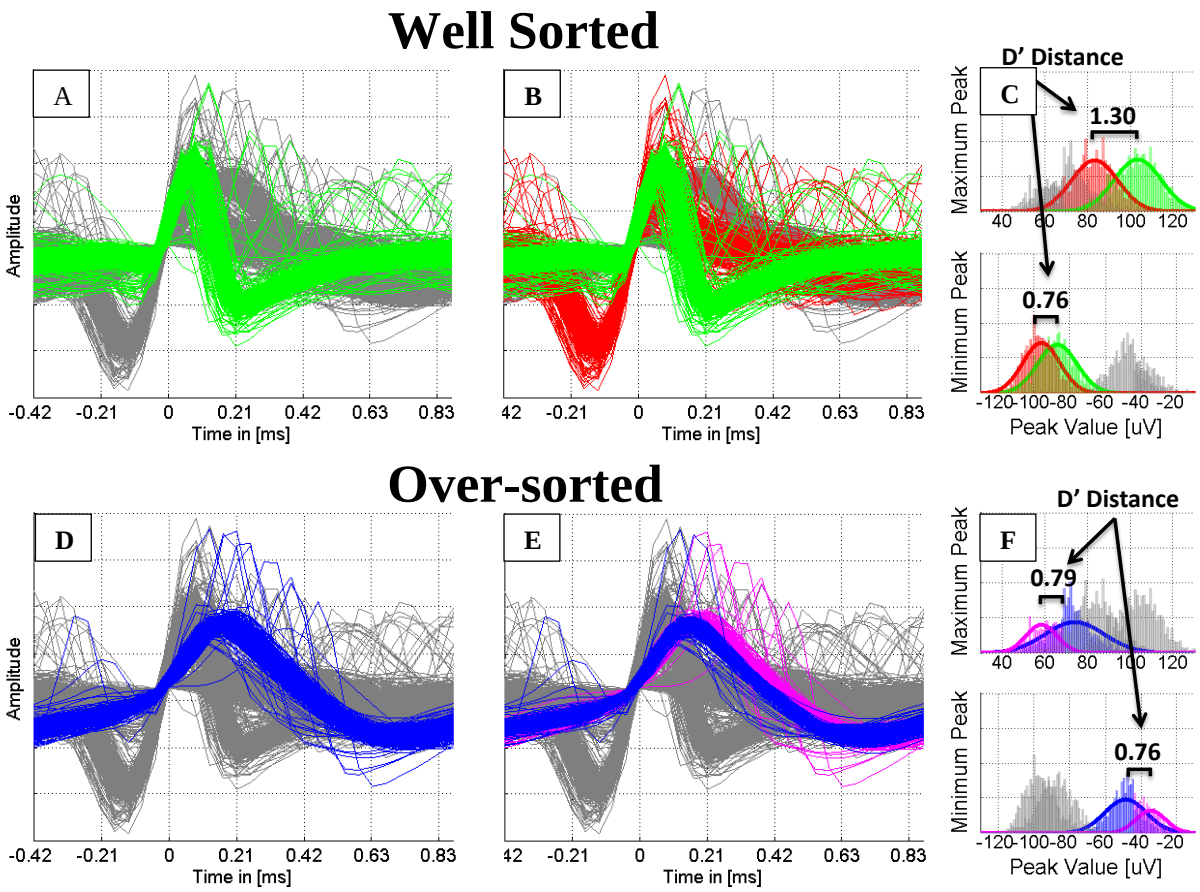

Figure 3.6: Over-Sorting Metrics: Statistically Similar Peaks

Part A: Waveform of Green Unit. Black arrows indicate locations of max/min peaks. **Part B**: Waveform of Green and Red Unit. Black arrows indicate locations of max/min peaks on both units. **Part C**: D' Histogram shows distance between distribution of max/min peaks. We see the while the max peak distribution is similar (D' < 1); however, the min peak distributions are not overlapping. Green and Red Units are not Over-Sorted. **Part D-F**, similar analysis is done on Blue and Magenta Units. However, the D' distance between the both the max and min peak distributions is less than one. Therefore, Blue and Magenta Units are overlapping and over-sorted.

3.3.2.b Mean Waveform SSE

In order to compare waveform shapes as a whole (instead of only its peaks), we pairwise compute the sum of squared errors distribution between each pair of units in our sort. First, we compute the mean waveform for unit A, and calculate the SSE distribution as the sum of squared errors for every waveform in the sort (all waveforms in all units) with the error in relation to the mean waveform of unit A. We organize the SSE distribution by units yielding N histograms (N is the number of units) of SSE values. By combining unit A's histogram pairwise with each other unit's histogram, we can see if two modes are present in each combined distribution of SSE values. Using the bimodality function, we can detect a unimodal distribution or multimodal distribution. If two modes are present, they units are isolated and not over sorted, thus passing the metric. If only one mode is present, the two units are overlapping and thus over-sorted, failing the metric (*see Equation 3.6*).

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points } n = 1, \dots, N$$

$$F_j(n) = \frac{1}{M}\sum_{i=1}^{M} f_{ij}(n) \quad \textbf{(1)}$$

$$sse_{jk}(i) = \sum_{n=1}^{N}\left(F_j(n) - f_{ik}(n)\right)^2 \quad \textbf{(2)}$$

$$SSE_{jk} = [sse_{jk}(1), sse_{jk}(2), \dots., sse_{jk}(M)] \quad \textbf{(3)}$$

$$Unit\ j's\ Metric\ Score = \begin{cases} 1 \text{ iff } d\left(SSE_{jj},\ SSE_{jk}\right)' > 1 \text{ for all } k \neq j \\ 0 \text{ iff } d\left(SSE_{jj},\ SSE_{jk}\right)' < 1 \text{ for any } k \neq j \end{cases} \quad \textbf{(4)}$$

Equation 3.6: Mean Waveform SSE

To calculate the metric score of unit j, we first calculate the mean waveform (Eqn. 1). For each other unit k, we calculate the sum of squared errors (Eqn. 2), between the mean waveform of unit j, and each waveform in unit k. The metric fails, if the SSE distribution (Eqn. 3) of unit j overlaps (*see Chapter 2: D' (Sensitivity Index)*) with any other unit k (Eqn. 4).

3.3.3 Under-sorting Metrics

These metrics are used to identify a unit which contains neural information from the multiple neurons. We want a single unit to contain all the information from a single neuron, so if neural information from many neurons in contained in one unit; the sorting algorithm has made an error.

3.3.3.a Residuals (Template Matching)

To examine the waveform distribution in each unit, snippets are mean subtracted and variance normalized. We examine the concentration about the mean waveform (of each unit) by fitting a Gaussian curve at each sample point, and measuring the percent of waveforms found within one standard deviation of the mean of that Gaussian. If the concentration of waveforms within one standard deviation of the mean is higher than an ideal Gaussian pdf (ie. the variance is lower), we would expect this unit to contain information from a single neuron; conversely, if the concentration is lower than the ideal, (i.e. the variance is higher), then we'd expect this unit to contain information from multiple neurons. At each a sample point with a higher variance than

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points } n = 1, \dots, N$$

$$F_j(n) = [f_{1j}(n), f_{2j}(n), \dots, f_{Mj}(n)] \quad \textbf{(1)}$$

$$Unit\ j's\ Metric\ Score = \begin{cases} 1 \text{ iff } F_j(n) \text{ is unimodal} & \text{for all } n = 1, \dots N \\ 0 \text{ iff } F_j(n) \text{ is multimodal} & \text{for any } n = 1, \dots N \end{cases} \quad \textbf{(2)}$$

Equation 3.7: Residuals

To calculate the metric score of unit j, we first form a distribution of waveform amplitudes at a single sample point (Eqn. 1). Each waveform in the unit, contributions a single value (the amplitude measured at that sample point). The metric fails if at any sample point the distribution is multi-modal (Eqn. 2).

the ideal Gaussian case, we check for bimodality of the histogram slice (across all waveforms of that particular unit). If any of the waveform slices are bimodal (according the function outlined previously), the unit fails this metric (*see Figure 3.7, Equation 3.7*).

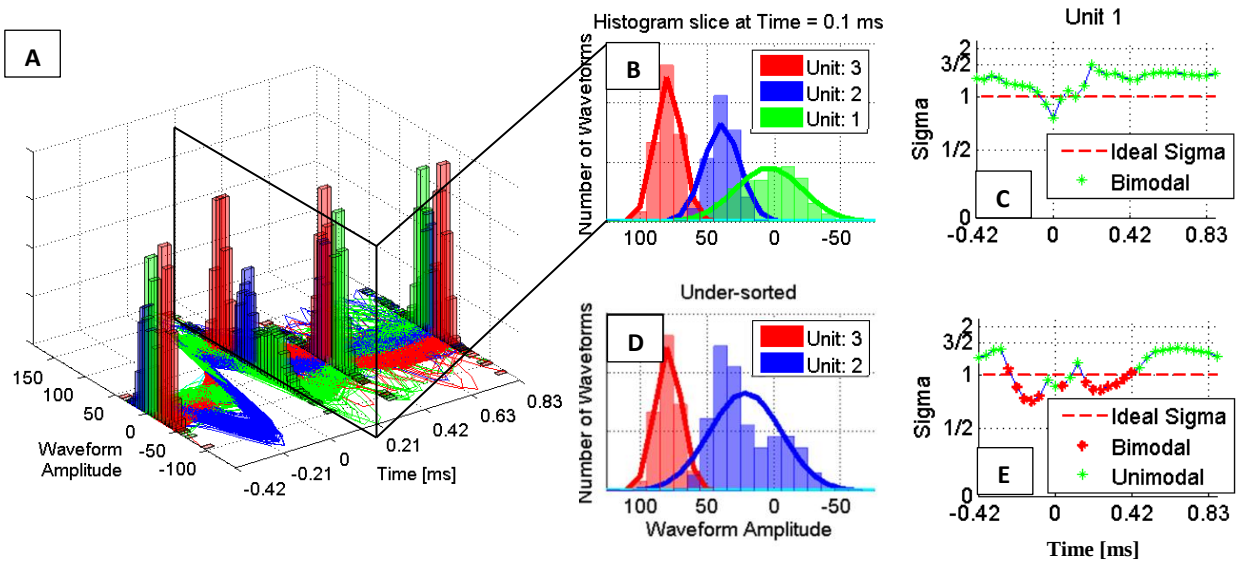

Figure 3.7: Under-Sorting Metrics: Residuals

Illustration of how residuals can be used to determine multi-unit clusters. Simulated data from the WaveClus dataset [13] was used to generate the figures. **Part A**: Histogram slices of distribution of waveform amplitudes (color-coded by Unit ID number) at various time stamps. **Part B**: Histogram of waveform amplitudes (color-coded by Unit ID number) at time 0.1ms after threshold crossing. **Part C**: Measured variance of waveforms at each waveform slice of Unit 1. We can see that this cluster is more concentrated around the mean than an ideal Gaussian pdf at nearly all sample points. This unit has no bimodal slices, and thus passes the metric. **Part D**: Histogram of an under-sorted Unit 1. Unit 1 contains information from both Neuron 1 and Neuron 2 from Part B. We see the Gaussian curve poorly fits the histogram. **Part E**: Percent of waveforms within one standard deviation of the Gaussian mean for Unit 1 at each sample point. For this under-sorted unit, we see that the majority of the samples points are not within one standard deviation. This unit would fail the metric.

3.3.3.b Waveform Similarity (Cross Correlation)

The mean waveform of each unit is computed and cross correlated with all the other waveforms in the unit. The peak value of each correlation becomes a similarity score between the waveform and the mean shape [14]. We generate a histogram of these similarity scores including all scores from within the same unit. If the similarity score histogram is bimodal, the unit fails this metric (*see Figure 3.8, Equation 3.8*).

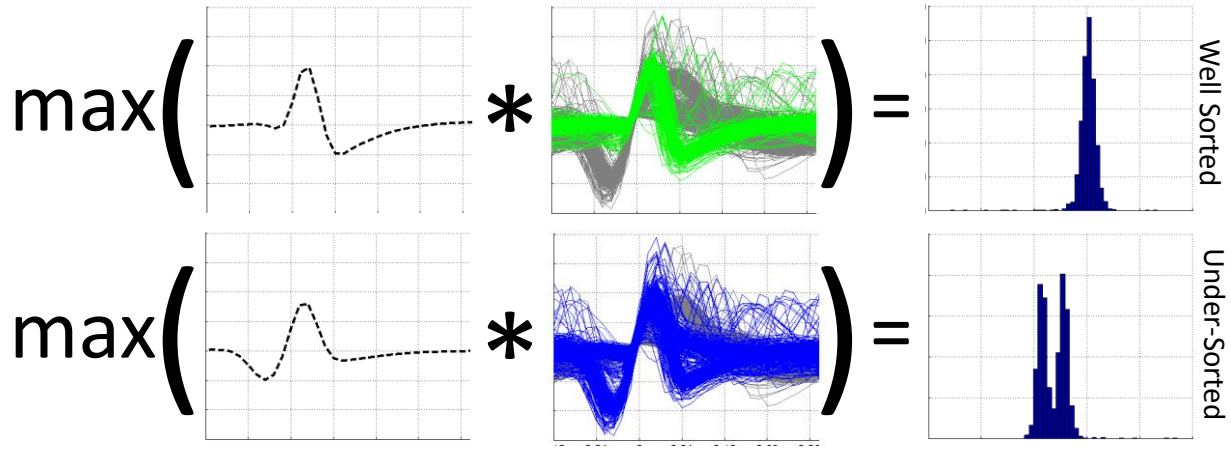

Figure 3.8: Under-Sorting Metrics: Waveform Similarity

We cross correlate (or convolve) the mean waveform with each waveform classified in that unit. We take the maximum value of this correlation (also called the similarity score [11]) and form a histogram of the values. If the histogram is unimodal, it is likely that unit contains information from a single neuron; otherwise, that unit is under-sorted.

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points } n = 1, \dots, N$$

$$F_j(n) = \frac{1}{M}\sum_{i=1}^{M} f_{ij}(n) \quad \textbf{(1)}$$

$$ss_j(i) = \max(\ F_j \otimes f_i\ ), \qquad SS_j = [ss_j(1), ss_j(2), \dots., ss_j(M)] \quad \textbf{(2)}$$

$$Unit\ j's\ Metric\ Score = \begin{cases} 1 \text{ iff } \mathrm{SS_j} \text{ is unimodal} \\ 0 \text{ iff } \mathrm{SS_j} \text{ is multimodal} \end{cases} \quad \textbf{(3)}$$

Equation 3.8: Cross Correlation

To calculate the metric score of unit j, we first calculate the mean waveform (Eqn. 1). For each waveform in unit j, we calculate the similarity score distribution (Eqn. 2), which is the maximum value of the convolution between the mean waveform of unit j, and each waveform in unit j. The metric fails, if the similarity score distribution of unit j is multi-modal (Eqn. 3).

3.3.3.c Threshold Slope

The slope across the threshold of each snippet is calculated. We generate a histogram of these slopes including all slopes from waveforms within the same unit. If the threshold slope histogram is bimodal, the unit fails this metric (*see Figure 3.9, Equation 3.9*).

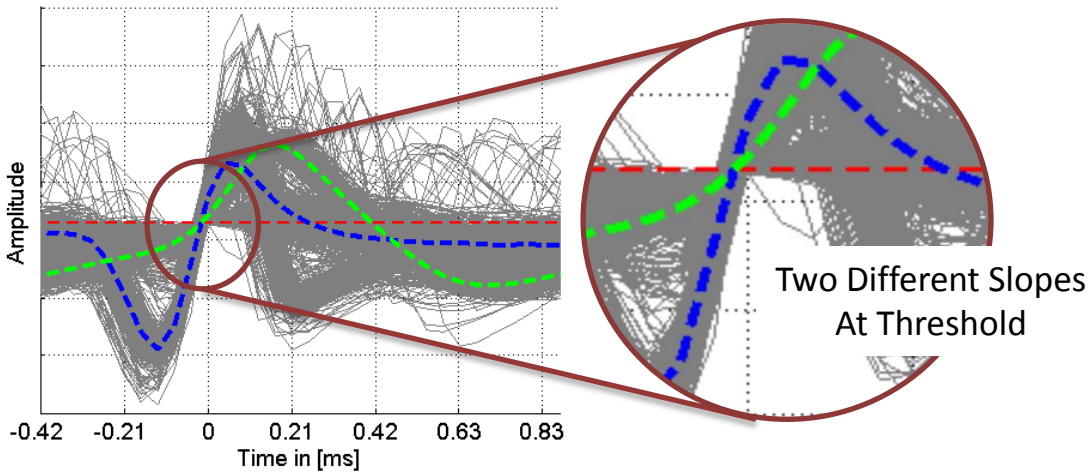

Figure 3.9: Under-Sorting Metrics: Threshold Slope

At the point which the waveforms are thresholded, we can see the rise time-constant contributing to two different slopes in the two different units. The red dotted line is the threshold value. We can look at a histogram for each unit, of the slopes of each waveform in that unit. If the histogram shows more than one modality in the data, we know that unit is under-sorted.

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points } n = 1, \dots, N$$

$$F_j = \left[f_{1j}(n) - f_{1j}(n-1), \dots, f_{Mj}(n) - f_{Mj}(n-1)\right] \quad \text{where } n = time = 0 \qquad \textbf{(1)}$$

$$Unit\ j's\ Metric\ Score = \begin{cases} 1 \text{ iff } F_j \text{ is unimodal} \\ 0 \text{ iff } F_j \text{ is multimodal} \end{cases} \qquad \textbf{(2)}$$

Equation 3.9: Threshold Slope

To calculate the metric score of unit j, we differentiate each waveform in the unit, and form a distribution of the values at the sample point when each waveform crosses the threshold (Eqn. 1). If that distribution is unimodal, the metric passes.

3.3.3.d Peak Waveform Amplitude

The absolute peak value of each snip within a unit is found. We generate a histogram of these peak values from all waveforms within the same unit. If the peak waveform histogram is bimodal, the unit fails this metric (*see Figure 3.10, Equation 3.10*).

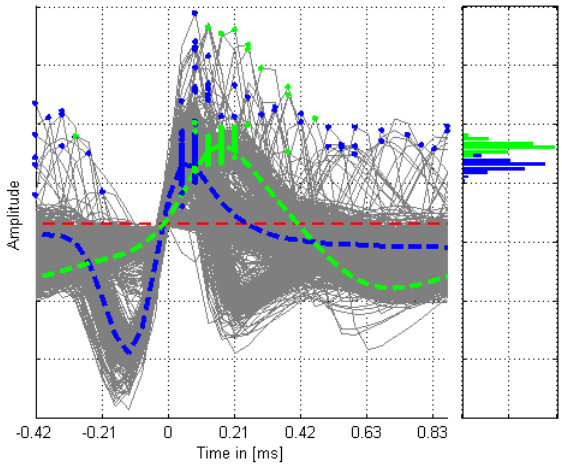

Figure 3.10: Under-Sorting Metrics: Peak Waveform Amplitude

The maximum absolute waveform peak value is shown in a histogram on the right. If these two waveform groups were combined, the histogram would be bimodal and under-sorted. However, it is not. Each colored histogram has a single peak, and therefore both of these units are not under-sorted.

$$f_{ij}(n) = waveform\ snippet\ i, from\ unit\ j, \text{sample points } n = 1, \dots, N$$

$$a_{ij} = \max\left(abs(f_{ij})\right), \qquad A_j = [a_{1j}, a_{2j}, \dots, a_{Mj}] \tag{1}$$

$$Unit\ j's\ Metric\ Score = \begin{cases} 1 \text{ iff } A_j \text{ is unimodal} \\ 0 \text{ iff } A_j \text{ is multimodal} \end{cases} \tag{2}$$

Equation 3.10: Maximum Peak Amplitude

To calculate the metric score of unit j, we differentiate each waveform in the unit, and form a distribution of the values at the sample point when each waveform crosses the threshold (Eqn. 1). If that distribution is unimodal, the metric passes.

4.0 RESULTS

Here we detail the results of using HSST on a generated dataset used with other popular spike sorting algorithms and some real neural data collected from the dorsal root ganglion (DRG) in cats (*see Chapter 1: Motivation for Spike Sorting in Current Research*).

4.1 DATASETS

In order to properly compare the performance of spike sorting algorithms, we need to run the classifiers on a dataset where we know the classification ground truth. When we record from the spinal cord, or brain or peripheral nerves, there is no "known truth". This is why spike sorting is needed! Instead, we can use a generated dataset, which simulates recording electrodes in conductive tissue and electrical properties of action potentials propagating down their axons. In this case, we know, *a priori*, the number of neurons responsible for the spikes in the generated dataset, as well as which neuron is responsible for each spike in the generated waveform.

4.1.1 Generated Dataset: NeuroCube

NeuroCube[20] simulates a point source electrode in the middle of a 1mm cube of conductive tissue. It allows the user to populate the cube with neurons far enough away from the electrode,

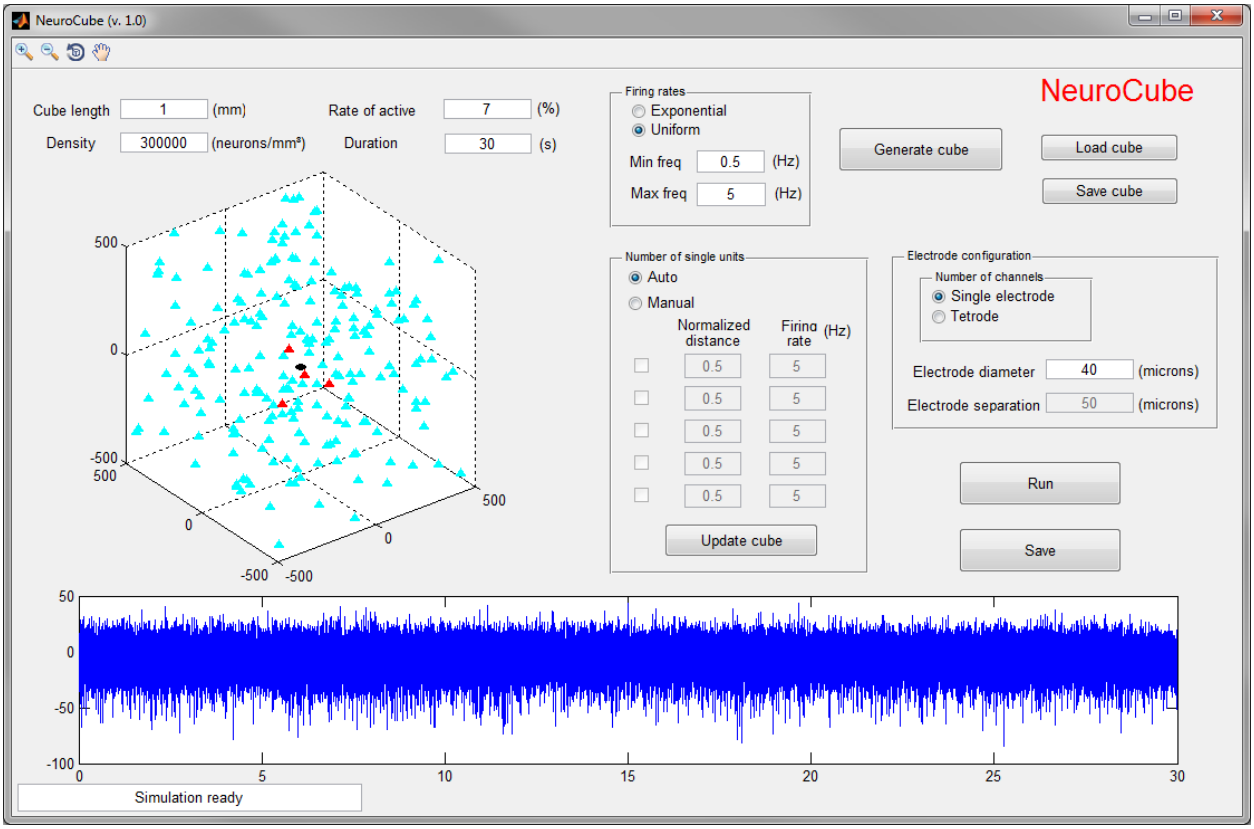

Figure 4.1: NeuroCube GUI

GUI allows user to populate simulated cube of tissue with background neurons (shown in cyan) at a specific density, close neurons (shown in red) at specific distances from the electrode, and adjust the firing rate of each neuron and duration of a simulated recording.

thus having large signal attenuation, that their electrical components recorded by the electrode are at amplitudes low enough that it becomes impossible to isolate the individual signals. These background neurons (colored light blue on *Figure 4.1*) are used to generate the background neural noise often seen on real recordings. The user can specify their density in the cube, generating more or less background noise. NeuroCube also allows the user to specify up to 5 neurons in close proximity to the electrode such that neural activity results in waveform amplitudes large enough to detect and isolate. Equation 4.1 shows the model of the recorded signal which is produced by the simulated electrode.

WaveClus [13], a recent developed spike sorting algorithm, used this program to generate 4 different datasets to test its performance. We used the same datasets to compare performance of HSST to WaveClus and other sorting algorithms. Each of the 4 datasets (*see Figure 4.2)* was also modified such that the close neurons were located at four different distances from the electrode: 50um, 100um, 150um, and 200um.

$$r(t) = \sum_{n=1}^{N}\sum_{k}^{K} a_n f_n(t - t_{n,k}) + \sum_{m=1}^{M}\sum_{k}^{K} a_m f_m(t - t_{m,k})$$

$$unique\_neural\_waveform\_snippet = [x(1), x(2), \dots, x(N)]$$

$$t_{n,k} = Poisson(\lambda_n) = k^{th} \text{ firing time of neuron } n$$

$$f_n(t) = \begin{cases} 0 & \text{at } t < 0 \\ unique\_neural\_waveform\_snippet \text{ amplitude} & \text{at } t = 1, \dots, N \\ 0 & \text{at } t > N \end{cases}$$

$$a_n = \frac{1}{r_n^2} \text{ where } r_n \text{ is the distance from neuron } n \text{ to the simulated electrode}$$

Equation 4.1: NeuroCube Model

r(t) is the recorded voltage of the simulated electrode. 500 different unique neural snippets are available. Each neuron n, is randomly assigned one of the 500 unique waveforms. *N* primary neurons (shown in red in *Figure 4.1*) are specified by the user (N < 6). Each primary neuron is guaranteed to have a different unique waveform. *M* background neurons (shown in cyan in *Figure 4.1*) are auto generated at a distance greater than 250 um with a density specified by the user (M < 10E+7).

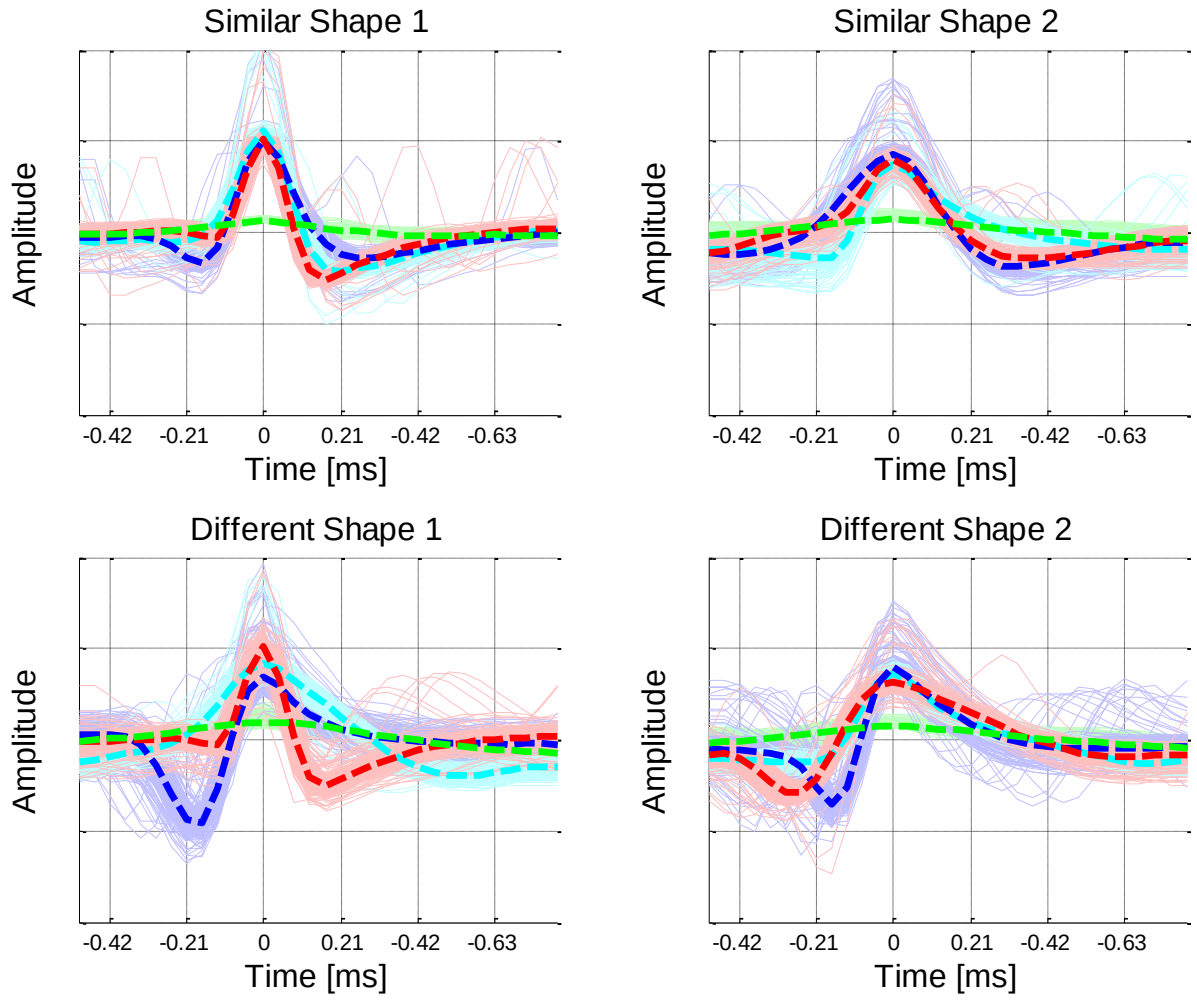

Figure 4.2: Four Generated Datasets from WaveClus

Each dataset used to validate WaveClus [13] is shown while the three close neurons (shown in red, blue and cyan) lay 50um from the simulated recording electrode. Waveforms have been thresholded and extracted from the continuous data. The green cluster represents the background multi-unit activity which has been detected.

4.2 HSST PARAMETER SELECTION ERROR

Before we compare performance between sorting algorithms, we need to look at the performance of HSST. HSST sorts a dataset using a sorting algorithm, and iteratively selects the parameter set

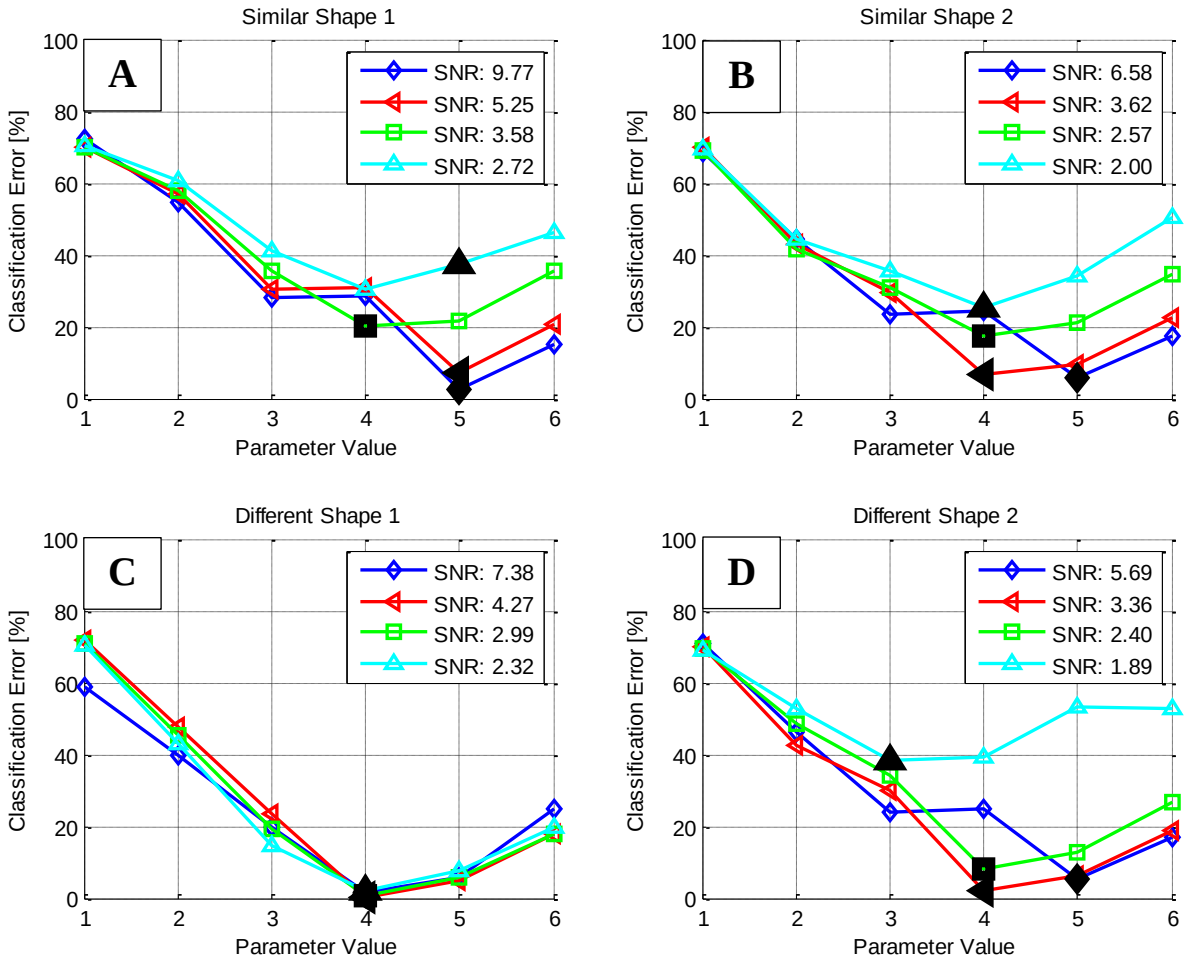

Figure 4.3: Classification Error as a Function of Parameter Settings

On each plot, classification error is shown on the Y axis, and the parameter setting is shown on the X axis. Gaussian Mixture Model (GMM) sorts each data set. The main parameter of the GMM algorithm sets the number of clusters identified. The Parameter Value in the above plot is the number of clusters. The colored lines indicate the distance from recording electrode (lowest SNR corresponding to distance 200um, while highest SNR corresponds to distance of 50um). Each colored line has a black marker indicating the parameter setting chosen by the HSST algorithm. Even though each dataset contains 4 known clusters, due to random initialization of the Expectation Maximization algorithm of GMM, Beta Errors can occur.

(drawn from a range of parameter sets) which yields an optimal result (s*ee Chapter 3: Methods*). What defines optimal? In case of the generated data, optimal means the clustering scheme which has the lowest classification error, defined as the number of waveforms labeled with an incorrect

cluster id. Since we have generated this data, we know exactly which neuron fires a spike at what time. We plot the classification error on a generated dataset by using a particular sorting algorithm's output over a range of parameter sets. In Figure 4.3, we see curves describing the classification error of the data in Figure 4.2. Each colored line represents the four distances between the neurons and the simulated electrode. Due to increasing distances, the SNR decreases as the distances increase. A Gaussian Mixture Model sorted the data with the number of components ranging from one to six, each yielding a different classification error. We can find the parameter which minimizes the classification error, and see which parameter HSST chose, to judge HSST's accuracy of selecting the best sorting parameter. We can make this plot, only because this data is generated. When we record neural activity in animals, the classification is not known; therefore we cannot do any optimization using the classification error on real data.

4.3 ALPHA AND BETA ERRORS

In order to further understand the source of these classification errors, we define two types of error, alpha and beta, to describe failures. Alpha errors (*see Figure 4.4*) occur when HSST selects a parameter set which results in a sorting algorithm producing non-optimal results. Beta Errors (*see Figure 4.5*) occur when HSST chooses a non-optimal parameter set which results in the sorting algorithm producing optimal results.

Alpha Errors occur when HSST selects a parameter which does not minimize the classification error. For example, in Figure 4.3.A, the light blue line shows this dataset sorted when the value of the GMM parameter was set to one through six. Parameter four yielded the

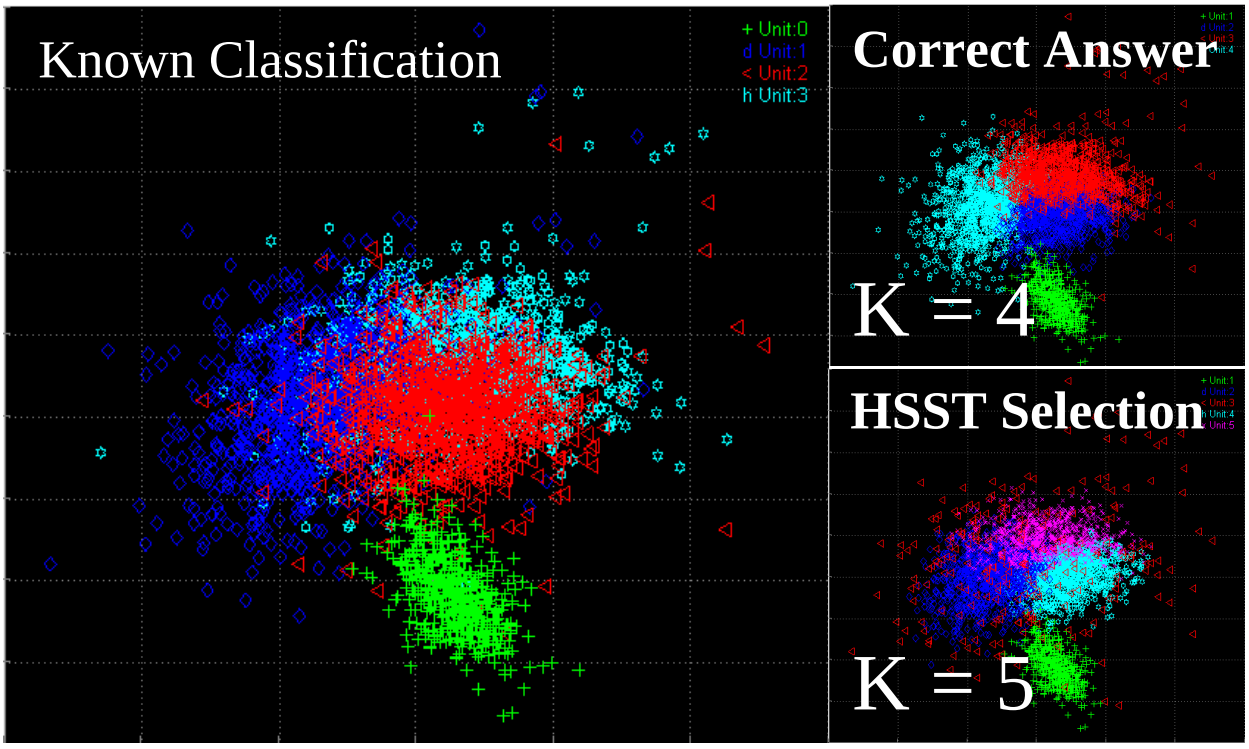

Figure 4.4: Alpha Errors

Waveforms are shown in 2D PCA space. On Figure 4.3.A, the light blue line shows the classification error rate as a function of the parameter selection. We see HSST selects 'K=5' as the optimal parameter value, even though 'K=4', yields the lowest classification error rate. This is an Alpha Error.

lowest classification error; however, HSST chose parameter five, because two units looked too similar when the parameter was set to four. Due to low SNR, the distributions of waveform shape of the two neurons overlapped, making it difficult to distinguish between two sources of neural information. This selection is an error, because the true number of clusters is four, and the parameter four yielded the lowest classification error.

Beta Errors occur when the sorting algorithm performs poorly, leading to a non-optimal result given a parameter which should yield the optimal result. HSST can't do anything about these errors; however, these errors can render a sorting algorithm useless, unless the end user can account for these types of error and predict when they will occur, which HSST cleanly

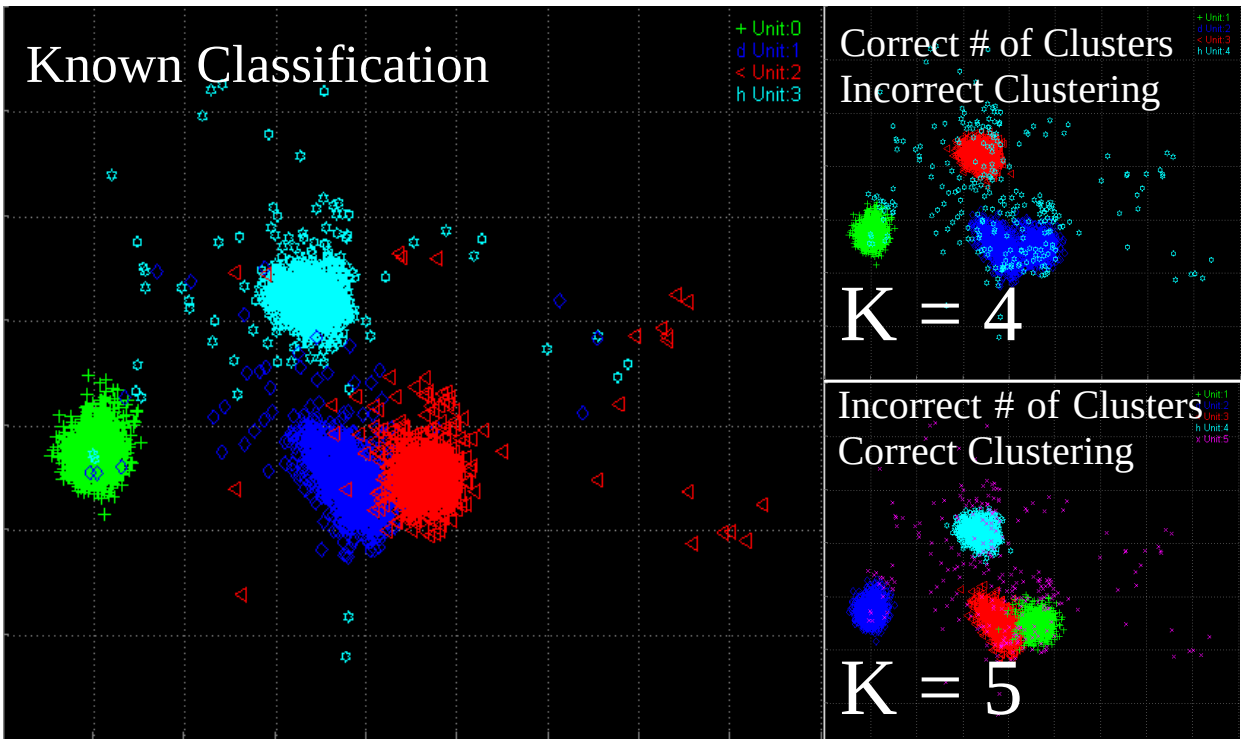

Figure 4.5: Beta Errors

Waveforms are shown in 2D PCA space. We see HSST selects 'K=5' as the optimal parameter value, even though the number of clusters in the known dataset is 4. A parameter value of 5 yields the lowest classification error rate, and HSST selects it appropriately. This is a Beta Error, caused by poor sorting algorithm performance.

is able to do! For example, in Figure 4.3.B, we see on the dark blue line, that 'K=5', yields the lowest classification error. 'K=5' is the "wrong" number of clusters, since we generated this data to include three neurons and background noise (totaling four clusters). However, due to a failure of the sorting algorithm, parameter set 'K=4', yielded a non-optimal result. This illustrates the strength of HSST. Even if the "optimal parameter" is known, in this case four clusters are known to be present; a sorting algorithm may not reach an optimal sort output. Unless a sorting algorithm is guaranteed to reach the global minima (which is very difficult to prove and not true for the vast majority of current sorting algorithms), it still could fall into a local minima, resulting in a poor classification result.

How often do these errors occur? We look at the number of occurances of Alpha Error in Figure 4.6. Now that we are convinced HSST does a good job of selecting the optimal parameter set, we can look at performance of different sorting algorithm (all using HSST to set their free parameters) and compare their performances.

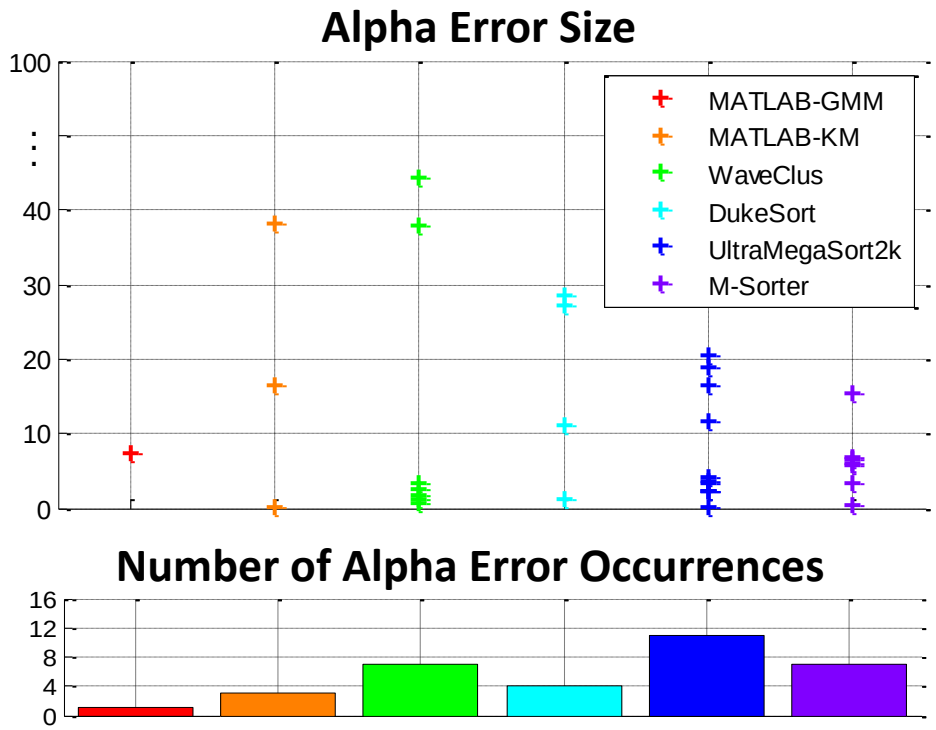

Figure 4.6: Frequency and Size of Alpha Error Occurrences

Illustration of error caused by HSST selecting non-optimal parameter. The X-Axis labels detail the number of markers on the graph in each row (one row for each sorting algorithm). The Y-Axis shows classification error.

We used HSST to select the optimal parameter of six sorting algorithms which sorted the data shown in Figure 4.4, at each distance from the electrode for a total of 16 generated datasets. For each dataset, we found the six parameter sets (one from each sorting algorithm) which produced the minimal classification error. We also found the parameter set which HSST selected. The difference in classification error was plotted above. Only the cases were the difference was greater than 0 (meaning HSST did not select the optimal parameter) are shown.

This means for row 1, GMM, HSST only selected the non-optimal parameter set for one dataset out of 16, which resulted in an increase of 8% classification error of above the absolute minimum error achievable with the optimal parameter set. This data point corresponds directly to the light blue curve in Figure 4.4.A. 8% is the difference between Parameter: 4 (the optimal parameter set) and 5 (the parameter set chosen by HSST. For the vast majority of cases, HSST selects either the optimal parameter, or a parameter within 5% of the optimal parameter.

4.4 SORTING ALGORITHM PERFORMANCE USING HSST

Naturally, we want to see how well HSST performs using many different types of sorting algorithms and compare their performance. We would like to select a sorting algorithm which is accurate, fast and reliable. Two main variables that sorting algorithms are often sensitive to: signal-to-noise ratio (SNR) variability and the number of units in the data.

4.4.1 SNR Variability

In Figure 4.7, we sorted the WaveClus [13] datasets using many different types of sorting algorithms (all using HSST to select the optimal parameter) and compared their misclassification rates. In Figure 4.7, MATLAB-GMM is a Gaussian mixture model with a variable number of components [33]. MATLAB-KM is a K-Means algorithm [6]. WaveClus is the algorithm proposed by Quiroga [13]. DukeSort is the method proposed by Chen, Carlson and Carin [18] and M-Sorter is the method proposed by Yuan et al [19]. UltraMegaSort2k is the method proposed by Hill et al [10]. We would also like to see how well HSST performed when selecting

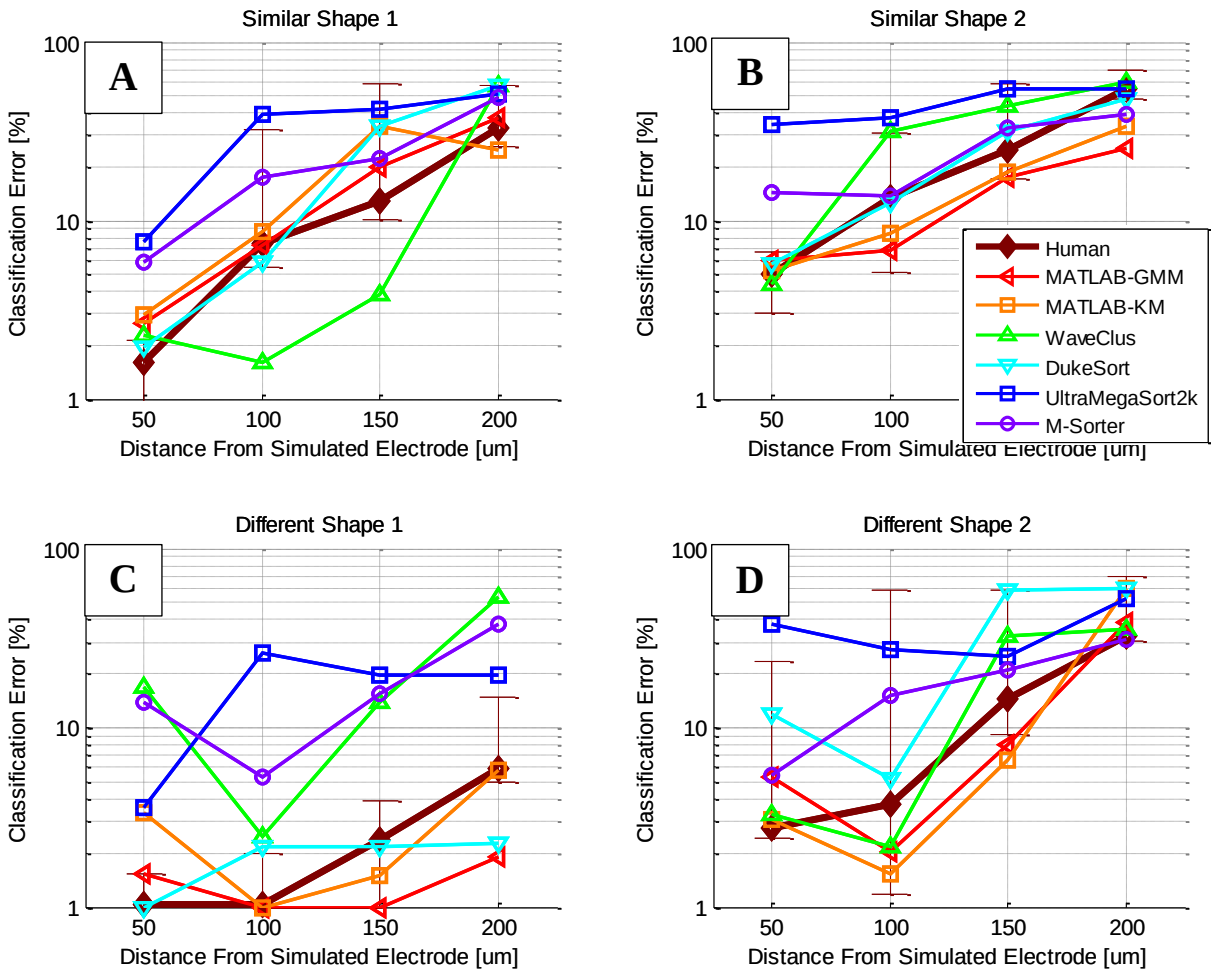

Figure 4.7: Classification Error Comparing Algorithms While Changing SNR

Within each of the four datasets, four different noise levels were present. HSST selected the optimal parameter for each sorting algorithm yielding these classification error results. Each of these algorithms required extensive parameter selection, with high sensitivity in accuracy to selection. Average Human performance is plotted in dark red as a base line with error bars showing max and min values. MATLAB-GMM is a Gaussian mixture model with a variable number of components [33]. WaveClus is the algorithm proposed by Quiroga [13]. MATLAB-KM is a K-Means algorithm [6]. DukeSort is the method proposed by Chen, Carlson and Carin [18] and M-Sorter is the method proposed by Yuan et al [19]. UltraMegaSort2k is the method proposed by Hill et al [10]. Matlab code was gathered from public resources provided by the authors. On each plot, classification error is shown on the Y axis on a log scale, and distance from the neuron to the recording electrode in each instance is on the X axis.

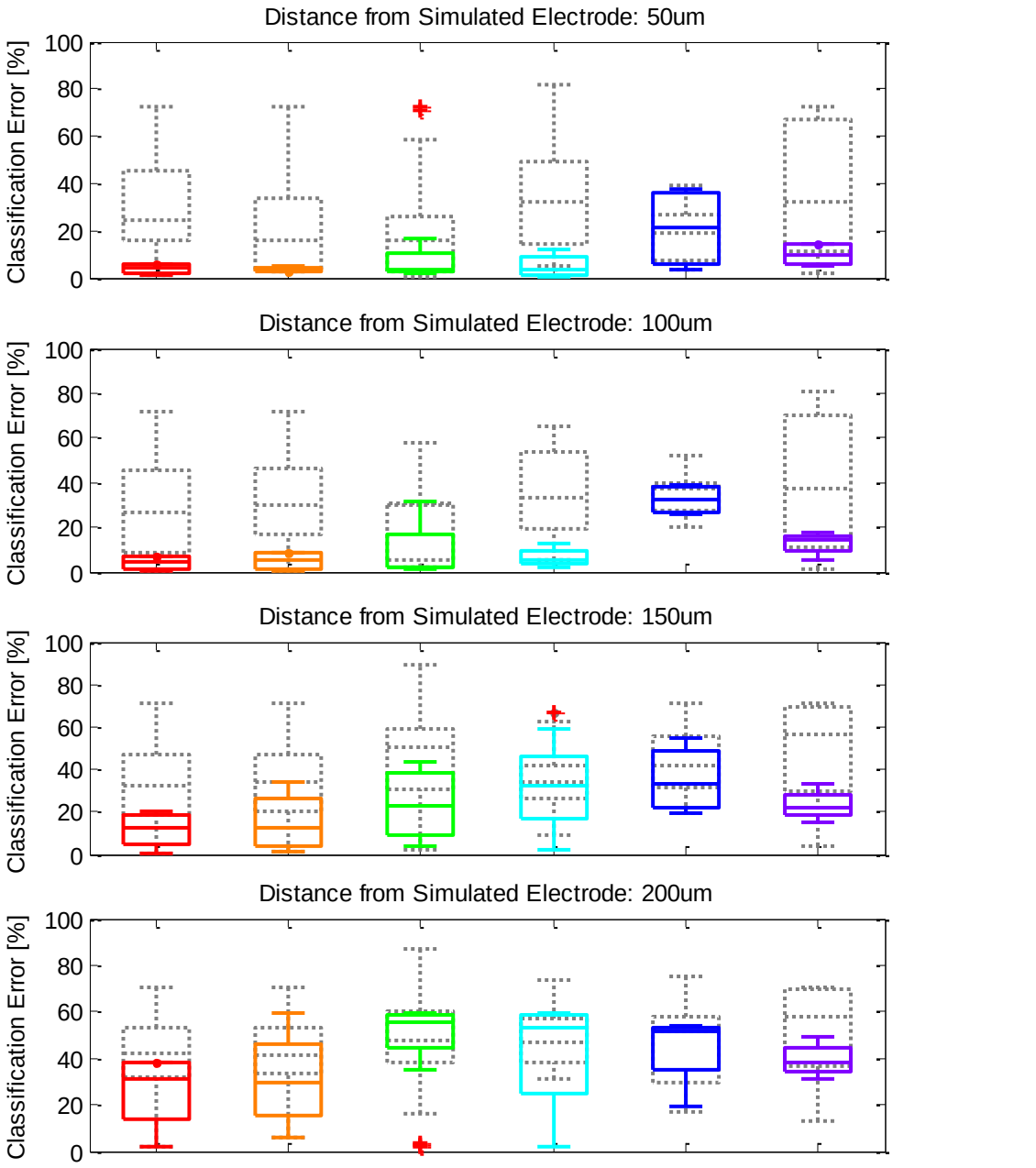

Figure 4.8: Parameter Sensitivity to Changes in SNR

Boxplot showing range of error resulting from parameter selection. Colored boxes show range of error from HSST selection. Grey boxes show range of error from all parameters included in the sweep. Some algorithms have high sensitivity to changes in SNR, other show little change in variability of error.

the optimal parameter. Figure 4.8 shows the range of error produced by the range of parameters given to the algorithms. The grey boxes show the scope of errors possible given the range of input parameters. The colored boxes show the spread of only those "optimal" parameters selected by HSST. This plot shows us two things. First, it shows the sensitivity of these algorithms to changes in SNR (proportional to distance between neuron and recording electrode), and second; it shows the sensitivity of the algorithms to changes in their parameters. We see for the GMM algorithm, the range of input parameters yields a large range of error. It is possible to do a really poor job of classification given some of the parameters; however the range of parameter selected by HSST shows a narrow low range of error. This is indicatory that HSST did a good job of selecting the range of parameters which yield very low error. We can also see for DukeSort[18], that the range of error yielded by all parameters is much tighter. This algorithm is not as sensitive to parameter selection, and therefore doesn't benefit as much from HSST selecting its parameters. The range of error is smaller, but improvement is not as drastic as with GMM. With all algorithms though, we see a decreased performance as SNR decreases (distance to the electrode increases), as we would expect.

4.4.2 Number of Units Variability

In Figure 4.9, we examined each dataset from Figure 4.2 at a single fixed distance, 150um. We discarded the other distances (50,100,200um) to constrain the SNR variable. We generated 8 new datasets by incrementally removing the units present in each dataset, leaving us with 12 datasets, four with three units, four with two units, and four with one unit. We wanted to examine performance as the number of clusters in our dataset changed, while SNR was held constant. We

see the same trends present as when we examined SNR variability. As expected, as the number of clusters decreased we saw the performance increase.

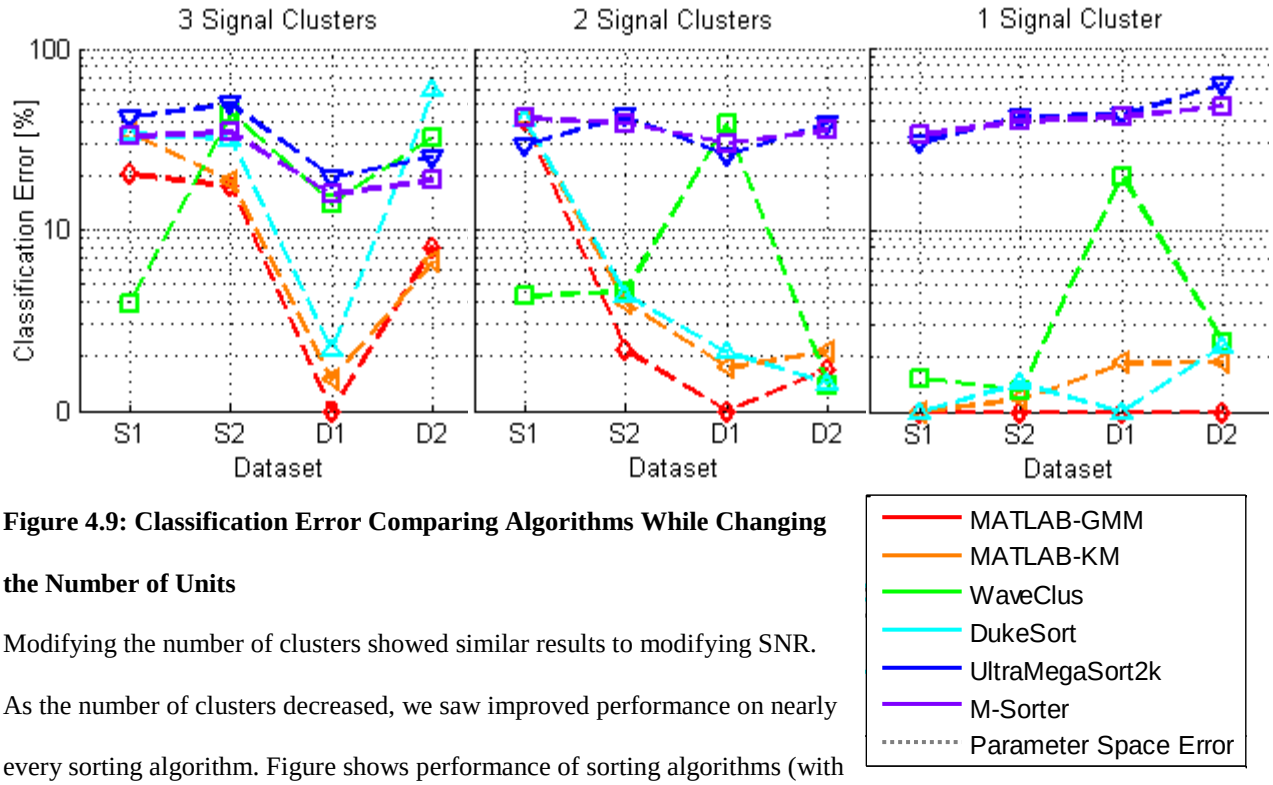

Figure 4.9: Classification Error Comparing Algorithms While Changing the Number of Units

Modifying the number of clusters showed similar results to modifying SNR. As the number of clusters decreased, we saw improved performance on nearly every sorting algorithm. Figure shows performance of sorting algorithms (with HSST parameter selection) across datasets (generated neural data all constrained to 150um from the electrode).

In Figure 4.10, we performed the same sentivity analysis as when looking at changes in SNR of the units. We plotted the parameter range error and the range of error produced by the parameters selected by HSST. The same trends we observed before where noticed in this analysis.

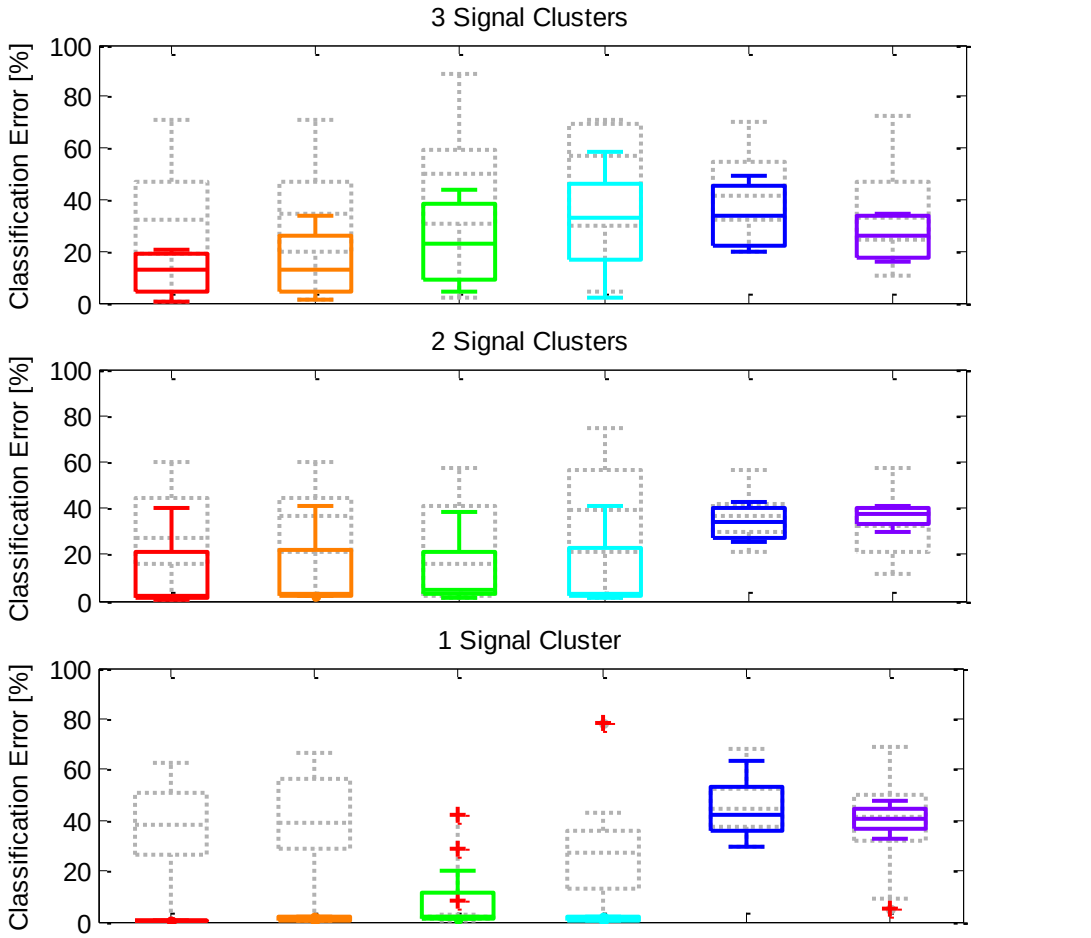

Figure 4.10: Parameter Sensitivity to Changes in Number of Units

Modifying the number of clusters showed similar results to modifying SNR. As the number of clusters decreased, we saw improved performance on nearly every sorting algorithm. Shows boxplots of possible error given the parameters space, with range of error from HSST selected parameters.

4.5 COMPARISON TO OTHER AUTOMATED SYSTEMS

In order to validate performance of the HSST algorithm against fully automated systems, we compared its accuracy against a generated dataset from the WaveClus paper [13], which has the option to select its own "optimal" parameter set. Compared to manual sorting, by an expert human sorter, HSST performance in classifying waveforms was on average at least as good as a human.

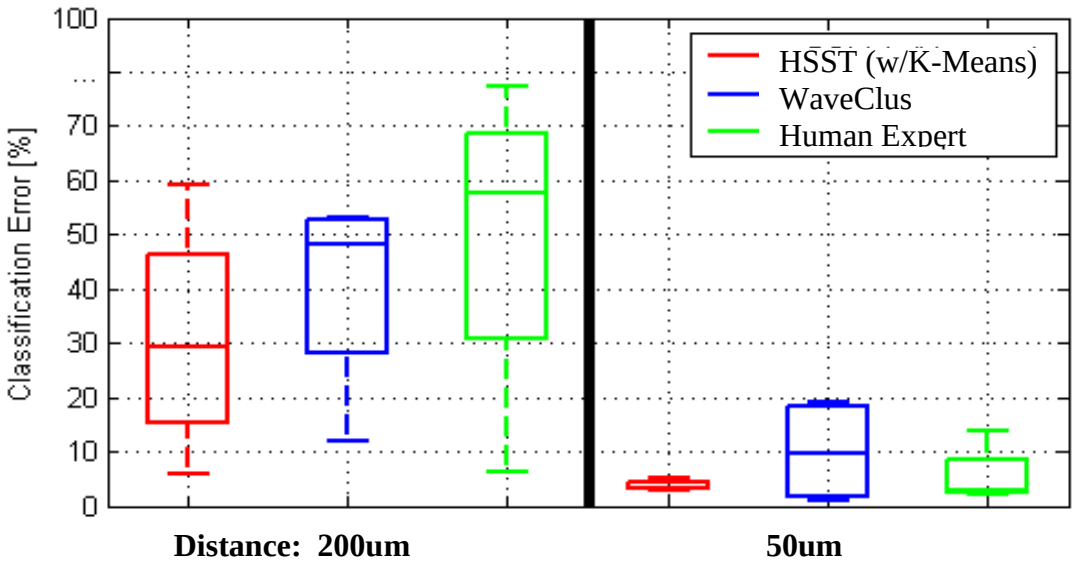

Figure 4.11: Average Classification Error between Unsupervised Algorithms

Results are an average of the four datasets in Figure 4.7 (distances 50um-200um only) compared to average error of four expert human sorters. We see that the HSST (with a simply K-means sorting algorithm) can do at least as well as a human, and does better than the other algorithms. Graph shows a boxplot, in which the middle bar shows the median value of the data, edges of the box show quartiles (25% and 75%) and whiskers show most extreme data points excluding outliners.

4.6 DATASETS: DRG RECORDINGS

To confirm our methods worked on real data, and not just simulated data, we selected 80 recordings from electrodes implanted in the DRG of cats. These recordings all had at least one very high SNR unit (> 5) which had been manually sorted by expert humans. Using the sorting algorithm compared previously, we sorted the data, using HSST to select their parameters.

Humans can do a relatively accurate job on well isolated units, however, once SNR decreases, they become more or less random in their judgments when classifying neural activity [5]. Error was evaluated in the ability of the sorting algorithm to minimally identify the unit in the recording with the highest SNR. Our criterion for developing this algorithm required as good, if not better than human sorting of well isolated units. We ignored other units with lower SNR present in each recording, since the ground truth is not known (only estimated by the human sorters). To compare performance, we calculated accuracy, sensitivity, and specificity of classification of each high SNR unit.

4.7 STATISTICAL MEASURES OF SORTING DRG RECORDINGS

To quantify performance of these sorting algorithms on our manually sorted data, we show a number of statistically metrics to judge quality (*see Figure 4.12*). Binary classification statistical measures were used, with the classifier either selecting a waveform as in the high SNR unit, or not in the high SNR unit. All other units in the data were ignored. Accuracy is a statistical measure, testing the strength of a binary classifier against the known classification. Sensitivity is a statistical measure qualifying the percent of high SNR unit waveforms which

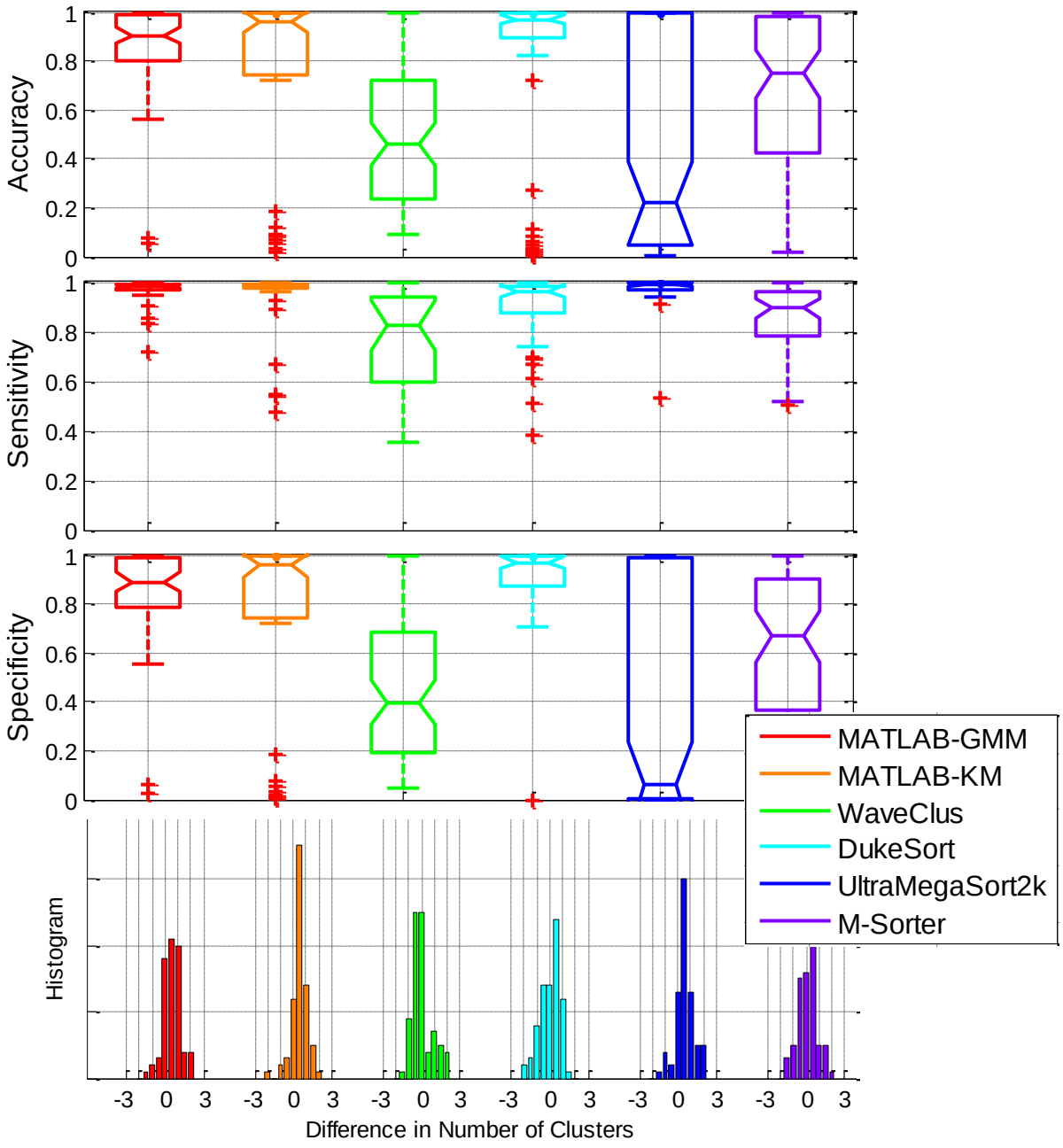

Figure 4.12: DRG Sorting Results - Accuracy, Sensitivity, and Specificity

80 datasets (each with one high SNR unit classified by an expert human sorter.) was sorted using various sort methods (all of whom had HSST selecting their free parameters). These results tell us K-Means, GMM and DukeSort did the best job of reliably selecting these units, while WaveClus and M-Sorter usually included waveforms which didn't belong in the cluster and UltraMegaSort2000 failed to find the high SNR unit. The bottom plot shows the difference between the total number of clusters identified by the expert human sorters and the number of clusters identified by the sorting algorithm. Each of these has zero mean with some standard deviation.

were correctly labeled. A high sensitivity score means the classifier correctly labeled high SNR waveforms as belonging to the high SNR unit. Specificity measures the rejection of waveforms not in the high SNR unit. High specificity scores mean waveforms not belonging to the high SNR unit are not labeled as belonging to the high SNR unit.

4.8 SPEED COMPARISON

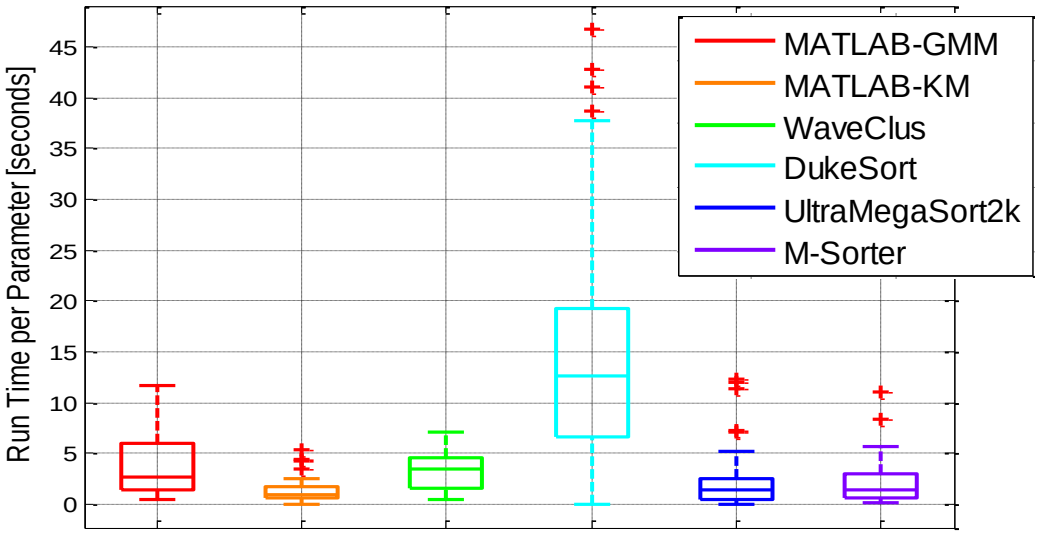

Figure 4.13: Algorithm Speed

Box plot of speed of each algorithm including the overhead of running HSST at the end of each parameter set (as you can see with MATLAB-KM method, the total time averages less than two seconds, so the additional time of scoring a sort with HSST is of negligible consequence). The Y Axis shows seconds to sort and score (using a single set of parameters) a dataset running the given algorithm. The 80 datasets sorted and scored each had an average of 5000 spikes to cluster.

It is also important to see the speed at which these algorithms run. Figure 4.13 shows the relative speed of sorting 5000 waveforms per parameter setting (including the overhead cost of scoring the sort generated by that parameter set with HSST).

5.0 CONCLUSIONS

So far, we have discussed the importance of spike sorting and various challenges it presents, including extremely difficult parameter estimation, inability to test the algorithm on real recorded data, and high variability of types of input data and application (*see Chapter 1, The Problem: Spike Sorting*). All of these make it challenging to build an unsupervised algorithm which can robustly handle many corner-cases and different types of data. We have presented the HSST (Heuristic Spike Sort Tuner) method for combining the statistical power of rigorous mathematical tools with the inherent biological information which is known about the data. HSST is generally applicable to a wide variety of sorting algorithms (*see Chapter 4, Classification Error of HSST Parameter Selection with Different Sorting Algorithms*), it can accurately select an optimal parameter set (*see Chapter 4, HSST Parameter Selection Error*), and it can beat other unsupervised algorithms' performance (*see Chapter 4, Comparison to Other Automated Systems*), even when using a basic sorting algorithm such as K-Means. Since basic sorting algorithms can be utilized with high performance, sorting large datasets can be quickly done (*see Chapter 4, Speed Comparison*). HSST also does a good job of mimicking expert human performance (*see Chapter 4, Accuracy, Sensitivity, and Specificity of Sorting DRG Recordings*) and easily outperforms humans in low SNR cases, while not missing highly isolated units that are often missed by sorting algorithms which, due to over fitting of parameters, can only handle a narrow subset of data. We have shown that HSST can handle both wide changes in

SNR and the number of clusters in a given sort, two of the most challenging parameters to estimate for a sorting algorithm (*see Chapter 4, SNR Variability Cluster Number Variability*).

5.1 DISCUSSION

The most vital aspect of this algorithm is its ability to determine the optimal parameter, given a finite set of parameters, for a specific sorting algorithm for the purpose of generating a reliable neurologically feasible result. Our main designing goal for the HSST was to consistently identity well isolated units in the recorded data. Many other sorting algorithms achieve good results on narrow sets of training data, but don't work well when sorting high volumes of variable neural data without much hand tweaking and human oversight of parameter choices. Our goal was to develop an algorithm which could (given a wide set of input parameters) accurately and consistently sort high volumes of the well isolated data. By seeking to add the biological prior information known about the data we are sorting, we hope to inform powerful statistical tools for extracting the neural signal from the noise to achieve a better result.

Some difficulties of implementing HSST are the strengths of the metrics. While in theory, being able to make judgments about the over-sorted, under-sorted, noise, or good units lead to excellent results. However, if the metrics are unable to identity information from the features they have extracted, the HSST method breaks down. It is easy for a human to look at some one dimensional data and determine bimodality; however, countless papers [51] [52] [53] [54] have tried to address this complex issue for many different types of applications. If bimodality is difficult to detect, the strength of the metrics will be weakened, and HSST will not make a well informed choice as to the parameter selection. Discretizing data always has a cost of

loss of information and inevitable requires some parameter-fixing at some level, whether it be the width of bins (for a histogram) or the number of samples from the waveforms, and for such cases HSST is not immune. This parameter fixing, within some of the metrics (although these have been fixed for every dataset analyzed in this study) can further contribute to some form of tuning for a particular dataset. While acknowledging that we cannot avoid this issue, we believe we have chosen values of those parameters which are appropriate for a wide set of data. Furthermore, it is possible to for a user to adjust these parameters if they are deemed inappropriate for a specific dataset. These values have fixed for the entirety of the analysis of this study. For instance, the SNR metric discards low SNR units which are potentially undesirable in our application. However, if the user does not wish to discard low SNR units, he/she can simply set the SNR rejection threshold to 0. The bimodality function has several fixed parameters, including the valley depth to determine bimodality. These values where determined empirically, however, are easily translatable to an accuracy score or some other measure of overlap between two one-dimensional Gaussian distributions. The user can set these parameters accordingly if the results are non-optimal.

While designing the method to calculate the final score, many solutions were proposed. Currently, it is an average of the three category scores, over-sorting, under-sorting, and noise. Some issue could be raised about the decision of how to incorporate each of the different metrics into those three scores. Certainly each of the three categories could be strengthened by the addition of other metrics; however, there is little data/evidence to suggest a right or wrong way to classify each of these conditions other than evaluating the end result of parameter selection. One could also imagine determining a weighting vector to attach to each of the metrics, depending on the strength of each metric. Some machine learning algorithm could easily decide

the weights if the metrics judged known units. Such analysis has not been done, and could be the topic of future study concerning this research.

One could also imagine designing an iterative algorithm to identity isolated units, remove them from the sort, and continue with the over and under sorted data until some end condition has been reached. Since information about each unit from each parameter set is known, one could build a sorting algorithm which iteratively changes its parameters in response to each unit score. These ideas would require some critical thinking regarding the best way to incorporate the detailed information provided about each unit. The results of knowing such detailed information about each unit could be very useful.

The HSST framework, allowing for anyone to add their own scoring methods, sort methods, feature extraction methods, has been made publicly available for use with Matlab 2013a (see Appendix A). The code for all the algorithms discussed in this thesis, along with all the data used to generate the figures in this paper, are also available from the sources listed.

5.2 FUTURE WORK

As mentioned earlier, additional sorting algorithms, additional metrics for judging the unit score could be introduced. Most of the metrics discussed here rely on some feature in the amplitude domain of the waveforms. Some metrics exploring features in the time domain, or frequency domain could prove to be useful. Future work includes development of a sorting algorithm which incorporates the information provided by the unit score, during the sorting process, by adding a feedback loop such that the sorting algorithm could adaptively change its parameters in response to the unit score feedback.

APPENDIX A

APPENDIX A: INSTRUCTIONS FOR DOWNLOADING, INSTALLING AND RUNNING HSST

As follows are the instructions for running, installing and operating HSST as a piece of software, distributed under the terms of the GNU General Public License. Any user has unrestricted access to change, modify, add, or remove the source code in whole or in part. The user has full permission to redistribute their version of the code, provided said redistribution refers to this thesis. Under no circumstances, can a redistribution of this software (either in whole or any part) be sold for profit, bundled with closed-source software or any software distributed with another license other than the GPL license. Other software bundled in the distribution was acquired through public access, or through express permission of the creator. Public links will be disclosed for any further interest. Every possible attempt at modifying the bundled software as little as possible has been made, and bundled software was only modified such that HSST could accept the inputs and outputs of the internal algorithm. That said, the author of this thesis makes no guarantees that any algorithm here has not been modified in some way which deviates from the original author's intent or is in fact identical to the publicly available copy.

A.1 REQUIRMENTS

This software was designed to operate on a Microsoft Windows 7 (x64 bit) operating system, with 8 GB of RAM, running MATLAB 2013a, 64-bit edition. No other hardware versions or software version are supported by the author.

A.2 INSTALLATION

After MATLAB 2013a has been installed, the HSST algorithm may be downloaded as a zip folder or as a cloned folder (using the GIT software), from the GIT hub website at the following url: https://github.com/davidbjanes/hsst. This folder (and all subfolders) must be added to the MATLAB path.

A.3 RUNNING THE SOFTWARE

A readme file is included in the top directory of the download zip file. Instructions inside show how to run the software, implement it in an existing neural signal processing data stream, and documentation on the code framework are all inside. Further questions may be directed to public@davidbjanes.com.

BIBLIOGRAPHY

[1] Abeles, M., & Goldstein, M. H. (1977). Multispike train analysis. *Proceedings of the IEEE*, *65*(5), 762–773.

[2] Lewicki, M. S. (1999). A review of methods for spike sorting : the detection and classification of neural action potentials, *9*(January).

[3] Harris, K. D., Henze, D. a, Csicsvari, J., Hirase, H., & Buzsáki, G. (2000). Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of neurophysiology*, *84*(1), 401–14.

[4] Harris, K. D., Hirase, H., Leinekugel, X., Henze, D. a, & Buzsáki, G. (2001). Temporal interaction between single spikes and complex spike bursts in hippocampal pyramidal cells. *Neuron*, *32*(1), 141–9.

[5] Wood, F., Black, M. J., Vargas-Irwin, C., Fellows, M., & Donoghue, J. P. (2004). On the variability of manual spike sorting. *IEEE transactions on bio-medical engineering*, *51*(6), 912–8.

[6] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on Mathematical Statistics and Probability*, *233*(233), 281–297.

[7] Fee, M. S., Mitra, P. P., & Kleinfeld, D. (1996). Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-Gaussian variability, 175–188.

[8] Pouzat, C., Mazor, O., & Laurent, G. (2002). Using noise signature to optimize spike-sorting and to assess neuronal classification quality. *Journal of Neuroscience Methods*, *122*(1), 43–57.

[9] Schmitzer-Torbert, N., Jackson, J., Henze, D., Harris, K., & Redish, a D. (2005). Quantitative measures of cluster quality for use in extracellular recordings. *Neuroscience*, *131*(1), 1–11.

[10] Hill, D. N., Mehta, S. B., & Kleinfeld, D. (2011). Quality Metrics to Accompany Spike Sorting of Extracellular Signals, *31*(24), 8699–8705.

[11] Fraser, G. W., & Schwartz, A. B. (2013). Recording from the same neurons chronically in motor cortex Recording from the same neurons chronically in motor cortex, (December 2011), 1970–1978.

[12] Gibson, S., Judy, J. W., & Markovic, D. (2008). Comparison of spike-sorting algorithms for future hardware implementation. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, *2008*, 5015–20.

[13] Quiroga, R. Q., Nadasdy, Z., & Ben-Shaul, Y. (2004). Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, *16*(8), 1661–87.

[14] Jackson, A., & Fetz, E. E. (2007). Compact movable microwire array for long-term chronic unit recording in cerebral cortex of primates. *Journal of* neurophysiology, 98(5), 3109–18. doi:10.1152/jn.00569.2007

[15] Green, D.M., Swets J.A. (1966) Signal Detection Theory and Psychophysics. New York: Wiley. (ISBN 0-471-32420-5)

[16] Mosteller. F. and Tukey, J.W. (1977) Data Analysis and Regression, Addison-Wesley. Reading, MA

[17] Cocatre-Zilgien, J. H., & Delcomyn, F. (1992). Identification of bursts in spike trains. Journal of neuroscience methods, 41(1), 19–30.

[18] Chen, B., Carlson, D., & Carin, L. (2011). On the Analysis of Multi-Channel Neural Spike Data. NIPS, (Ml), 1–9.

[19] Yuan, Y., Yang, C., & Si, J. (2012). The M-Sorter: an automatic and robust spike detection and classification system. Journal of neuroscience methods, 210(2), 281–90.

[20] Camuñas-Mesa, L. a, & Quiroga, R. Q. (2013). A detailed and fast model of extracellular recordings. Neural computation, 25(5), 1191–212.

[21] Image: http://www.factrange.com/interesting-facts-information-human-brain/

[22] Image: http://newton.umsl.edu/tsytsarev_files/Lecture02.htm

[23] Image: http://hyperphysics.phy-astr.gsu.edu/hbase/biology/actpot.html

[24] Image: http://www.urbanchildinstitute.org/why-0-3/baby-and-brain

[25] Herculano-Houzel S (2009). "The human brain in numbers: a linearly scaled-up primate brain". Frontiers in Human Neuroscience 3: 31.

[26] http://neurosurgery.stanford.edu/research/NPTL/research2.html

[27] http://www.neurofisiologia.net/?p=694

[28] http://www.massgeneral.org/psychiatry/research/neuroimaging_equipment.aspx

[29] http://www.brain-tuning.de/2ndlevel/bci/compo3_b.htm

[30] Joshua, M., Elias, S., Levine, O., & Bergman, H. (2007). Quantifying the isolation quality of extracellularly recorded action potentials. Journal of neuroscience methods, 163(2), 267–82.

[31] Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". Journal of the Royal Statistical Society, Series B 39 (1): 1–38.

[32] Hubel, D. H.; Wiesel, T. N. (1959). "Receptive fields of single neurones in the cat's striate cortex". The Journal of physiology 148 (3): 574–591

[33] Neal, Radford; Hinton, Geoffrey (1999). "A view of the EM algorithm that justifies incremental, sparse, and other variants". In Michael I. Jordan. Learning in Graphical Models (Cambridge, MA: MIT Press): 355–368.

[34] MacMillan N, Creelman C (2005) Detection Theory: A User's Guide. Lawrence Erlbaum Associates. (p.7)

[35] Gold C, Henze DA, Koch C, Buzsaki G (2006) On the Origin of the Extracellular Action Potential Waveform: A Modeling Study. J Neurophysiol 95:3113-3128

[36] Gerstein GL, Clark WA (1964) Simultaneous Studies of Firing Patterns in Several Neurons. Science 143:1325-1327

[37] Buzsaki G (2004) Large-scale recording of neuronal ensembles. Nature Neuroscience 7:446-451.

[38] Brown EN, Kass RE, Mitra PP (2004) Multiple neural spike train data analysis: state-of-the-art and future challenges. Nature Neuroscience 7:456-461.

[39] Boulton, A. A. (1990). Neurophysiological techniques: applications to neural systems. Clifton, New Jersey: Humana Press.

[40] Milstein JN and Koch C. (2008) Dynamic moment analysis of the extracellular electric field of a biologically realistic spiking neuron. Neural Comput. 20: 2070-2084

[41] Rousche, P. J. and R. A. Normann (1998). "Chronic recording capability of the Utah Intracortical Electrode Array in cat sensory cortex." Journal of Neuroscience Methods 82: 1-15.

[42] Thompson, R. F. (1973). Bioelectric Recording Techniques: Part A Cellular Processes and Brain Potentials. New York: Academic Press.

[43] Hubel, D. H. (1957). Tungsten Microelectrode for Recording from Single Units. Science 125, 549550.

[44] Holt, G. R. and Koch, C. (1999). Electrical interactions via the extracellular potential near cell bodies. J Comput Neurosci 6, 169184.

[45] Pettersen, K. H., Hagen, E., and Einevoll, G. T. (2008). Estimation of population firing rates and current source densities from laminar electrode recordings. J Comput Neurosci 24, 291313.

[46] Rall W, Shepherd GM (1968) Theoretical reconstruction of field potentials and dendro-dendritic synaptic interactions in olfactory bulb. J Neurophysiol 31:884–915.

[47] Pettersen, K. H. and Einevoll, G. T. (2008). Amplitude Variability and Extracellular Low-Pass Filtering of Neuronal Spike Biophys J 94, 784802

[48] Aoyagi, Y., Stein, R. B., Branner, A., Pearson, K. G., & Normann, R. a. (2003). Capabilities of a penetrating microelectrode array for recording single units in dorsal root ganglia of the cat. Journal of Neuroscience Methods, 128(1-2), 9–20.

[49] Image: http://blog.csdn.net/jwh_bupt/article/details/7663885

[50] Harris KD (2005) Neural signatures of cell assembly organization. Nat Rev Neurosci 6:399-407

[51] Hartigan JA, Hartigan PM (1985) The dip test of unimodality. Ann Statist 13 (1) 70-84

[52] Rozál GPM Hartigan JA (1994) The MAP test for multimodality. J Classification 11 (1) 5-36

[53] Larkin RP (1979) An algorithm for assessing bimodality vs. unimodality in a univariate distribution. Behavior Research Methods 11 (4) 467-468

[54] Holzmann H, Vollmer S (2008) A likelihood ratio test for bimodality in two-component mixtures – with application to regional income distribution in the EU.AStA 2(1)57-69

[55] The variable X-ray spectrum of Markarian 766 - I. Principal components analysis. L. Miller, T. J. Turner, J. N. Reeves, I. M. George, S. B. Kraemer and B. Wingert. A&A 463 (1) 131-143 (2007)

[56] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" (PDF). Philosophical Magazine 2 (11): 559–572.

[57] http://cidia.ucsd.edu/68.html

[58] http://people.brandeis.edu/~sekuler/eegERP.html

[59] Niedermeyer E. and da Silva F.L. (2004). Electroencephalography: Basic Principles, Clinical Applications, and Related Fields. Lippincot Williams & Wilkins. ISBN 0-7817-5126-8.

[60] Thomas CA, Springer PA, Loeb GE, Berwald-Netter Y, Okun LM. 1972. A miniature microelectrode array to monitor the bioelectric activity of cultured cells. Exp Cell Res. 74: 61-66.

[61] Huettel, S. A.; Song, A. W.; McCarthy, G. (2009), Functional Magnetic Resonance Imaging (2 ed.), Massachusetts: Sinauer, ISBN 978-0-87893-286-3

[62] Cohen D. "Magnetoencephalography: evidence of magnetic fields produced by alpha rhythm currents."Science 1968;161:784-6

[63] Penfield, W. and Boldrey, E. (1937). "Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation". Brain 60 (4): 389–443

[64] Kandel ER, Schwartz JH, Jessell TM 2000. Principles of Neural Science, 4th ed. McGraw-Hill, New York. ISBN 0-8385-7701-6

[65] Eckart, C.; Young, G. (1936). "The approximation of one matrix by another of lower rank". Psychometrika 1(3): 211–8

[66] Loève, M. (1978). Probability theory. Vol. II, 4th ed. Graduate Texts in Mathematics 46. Springer-Verlag. ISBN 0-387-90262-7