2015

Analyzing and clustering neural data

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Thesis

ANALYZING AND CLUSTERING NEURAL DATA

by

AMIT SINHA

B.S., Rutgers University, 2011
M.S., Boston University, 2015

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science

2015

Approved by

First Reader

David Castanon, PhD
Professor of Electrical and Computer Engineering
Professor of Systems Engineering

Second Reader

Peter (Sang) Chin, PhD
Chief Scientist-Decision Systems, Charles Stark Draper Laboratory
Research Professor of Computer Science

Facilis descensus Averni;
Noctes atque dies patet atri janua Ditis;
Sed revocare gradum, superasque evadere ad auras,
Hoc opus, hic labor est. Virgil

Acknowledgments

Thank you to my Advisors, professor David Castanon and Peter (Sang) Chin. Thank you to the Charles Stark Draper Laboratory for the oppurtunity to contribute to the TRANSFORM-DBS project. Thank you to all the professors that I interacted with at Boston University. Thank you to my Mom, Dad, and sister.

Amit Sinha
Student
ECE Department

ANALYZING AND CLUSTERING NEURAL DATA

AMIT SINHA

ABSTRACT

This thesis aims to analyze neural data in an overall effort by the Charles Stark Draper Laboratory to determine an underlying pattern in brain activity in healthy individuals versus patients with a brain degenerative disorder. The neural data comes from ECoG (electrocorticography) applied to either humans or primates. Each ECoG array has electrodes that measure voltage variations which neuroscientists claim correlates to neurons transmitting signals to one another. ECoG differs from the less invasive technique of EEG (electroencephalography) in that EEG electrodes are placed above a patients scalp while ECoG involves drilling small holes in the skull to allow electrodes to be closer to the brain. Because of this ECoG boasts an exceptionally high signal-to-noise ratio and less susceptibility to artifacts than EEG [6]. While wearing the ECoG caps, the patients are asked to perform a range of different tasks. The tasks performed by patients are partitioned into different levels of mental stress i.e. how much concentration is presumably required. The specific dataset used in this thesis is derived from cognitive behavior experiments performed on primates at MGH (Massachusetts General Hospital).

The content of this thesis can be thought of as a pipelined process. First the data is collected from the ECoG electrodes, then the data is pre-processed via signal processing techniques and finally the data is clustered via unsupervised learning techniques. For both the pre-processing and the clustering steps, different techniques are applied and then compared against one another. The focus of this thesis is to evaluate clustering techniques when applied to neural data.

For the pre-processing step, two types of bandpass filters, a Butterworth Filter and a Chebyshev Filter were applied. For the clustering step three techniques were applied to the data, K-means Clustering, Spectral Clustering and Self-Tuning Spectral Clustering. We conclude that for pre-processing the results from both filters are very similar and thus either filter is sufficient. For clustering we conclude that K-means has the lowest amount of overlap between clusters. K-means is also the most time-efficient of the three techniques and is thus the ideal choice for this application.

Contents

List of Tables

List of Figures

List of Abbreviations

D	. . . .	Diagonal Degree matrix
e	. . . .	matrix of eigenvectors
G	. . . .	gain in a filter
k	. . . .	expected number of clusters a priori
K	. . . .	Kth Nearest Neighbor
S	. . . .	Similarity Matrix
U	. . . .	Probability Distribution Function
x	. . . .	data sample

Chapter 1

Introduction

1.1 Introduction

Understanding how dynamic changes in brain activity correspond to learning is a major challenge in cognitive neuroscience. A topic of interest is finding a correlation in neural spiking data and an observable stimuli the subject is experiencing. The application of signal processing techniques to raw analog signals, along with various machine learning techniques to neural spiking data, will be investigated along with an analysis of trade offs between chosen parameters and underlying distributions.

In order to contribute to the Transdiagnostic Restoration of Affective Networks by System Identification and Function Oriented Real- Modeling and Deep Brain Stimulation (TRANSFORM-DBS) project at the Charles Stark Draper Laboratory, an approach will be developed for determining the salient features of neural data when a subject is experiencing specific stimuli. The goal of this thesis is help determine an underlying pattern in the neural data via clustering. Once this is done, other efforts within the overall project can develop ways to correlate the neural data with external stimuli applied to the patient.

The overall goal of the TRANSFORM-DBS Project is to develop a brain-implantable chip that monitor signals across multiple brain structures in real time to alleviate symptoms related to neuropsychiatric disorders such as PTSD (Post Traumatic Stress Disorder), severe depression, drug addiction, and TBI (Traumatic Brain Injury). Because the overall approach is expected to run on the implanted CPU as well as in real time, any effort to reduce the complexity of the approach is worthwhile. Therefore approaches that reveal underlying structure in the data (via unsupervised learning techniques) that are also computationally efficient are desirable.

This will be accomplished by first pre-processing the data via bandpass filters to remove noise. The resulting waveforms are then grouped together via amplitude thresholding. Finally clustering techniques will be applied to the resulting waveforms to see if an underlying pattern exists in the data.

1.2 Review of Prior Work

There are other studies that aim to characterize neural data by clustering neural spikes. Most of these studies treat the pre-processing step in a similar fashion, where the raw ECoG/EEG data is bandpass filtered and amplitude thresholding is applied to extract prominent spikes. From here, each the studies aims to analyze the resulting waveforms through some form of clustering. The specific goals of each study and thus their approach to clustering differs.

Carin et al. (1) propose a methodology for clustering EEG data that claims to accurately capture spikes that occur, disappear and then reappear in a dataset that spans over multiple days. Their approach first uses dictionary learning to learn elements of a dictionary that would allow the original signal to be represented as a sparse vector. Those dictionary elements are then clustered using what (1) refers to as a focused mixture model, which is a mixture model that focuses on a small number of clusters, those with a high probability of being observed from a set of specified priors. There are many priors and parameters involved in the focused mixture model approach described in (1) and the rationale for choosing each of these distributions isnt clear.

Calabrese and Planksi (2) proposed a mixture of Kalman filters (MoK) model to explicitly deal with slow changes in the shape of the waveform. This approach also models the spike rate and potentially the refractory period (the period where its believed a single neuron must wait until re-firing). This approach is novel in the sense that it claims to deal with nonstationarity (i.e. phase lag between otherwise similar waveforms) in the data. This approach however relies on Expectation Maximization (EM) to estimate the model parameters in the mixture of Kalman Filters. Depending on the termination criteria, the time complexity of the overall approach could be significantly worse than a simpler method.

Another approach known as Superparamagnetic Clustering is described in (3) where after the initial pre-processing, the coefficients of the waveforms wavelet decomposition are treated as features. This approach then uses a generalized version of the Ising model, known as the Potts model, to assign random states to each sample in the data. In the Ising model, each sample of data can be thought of as a point in a lattice. The Ising model allows each point in a lattice one of two different spin values but the Potts model can allow q spin values. After random states have been assigned, the approach performs Monte Carlo iterations until the probability of any given point changing from its neighbors has converged to a local minimum, and thus clusters are formed. While this approach claims to infer the ideal amount of clusters instead of choosing a value a priori (as in k-means and spectral clustering) this approach has other parameters in the form of the number of possible states q per point as well as the temperature T. Its not obvious what these parameters should be set to for any given clustering application. Furthermore the run time of a Monte Carlo based approach may prove to be less efficient than a simpler clustering method.

While there is a number of existing studies that propose solutions to clustering neural data, the above approaches are complex. In this thesis, we consider the use of simpler machine learning techniques that can effectively cluster the data to reveal the underlying structure of the waveforms. The resulting techniques will be computationally efficient, and lead to accurate clustering into meaningful classes that correlate well with varying external stimuli. The goal of the project is the make the overall approach both accurate and computationally efficient.

Chapter 2

Approach

2.1 Problem formulation and Solution

The overall goal of the TRANSFORM-DBS project is to find a correlation to neural activity and observable phenomenon in the patient such as mood swings, anxiety attacks, stress, etc... While we can attempt to induce different behaviors in a patient and then treat each set of recordings as a state (or label in a machine learning context) we dont know the underlying mechanisms involved in a patients cognitive behavior. For example, in an experiment where we induce behavior in a patient there may be some internal states that lead to the final observable phenomenon. A deeper understanding of those internal states may allow us to more accurately diagnosis patients, develop more effective treatments and predict future developments in patients.

Because we do not know the internal mechanisms of cognitive behavior we have no ground truth i.e. no baseline to compare empirical datasets to. Thus we treat this as an unsupervised learning application. Figure 2-1 captures the step-by-step process taken to analyze the data. First, before analyzing the neural data we first must consider how to remove noise and electrode artifacts from the data. Finally clustering algorithms need to be applied to the chosen features.

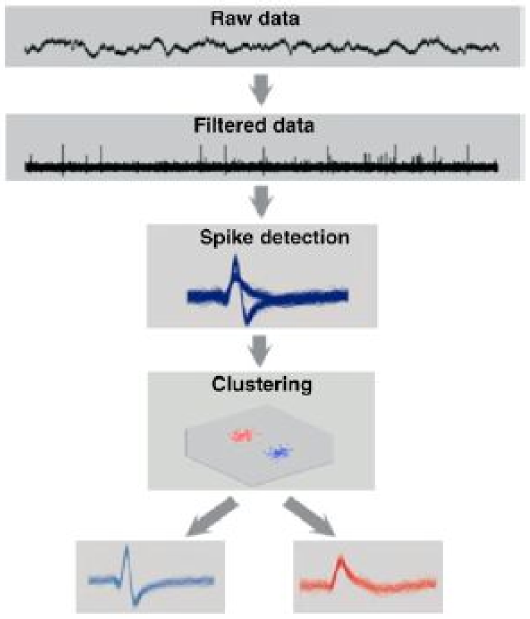


Figure 2·1: A pipelined approach to analyzing the neural data.

2.2 Filtering

The first step in our analysis is the filter the raw data gathered from the various electrodes in the EEG cap. The first type of filter considered is the butterworth filter, first described in 1930 by Stephen Butterworth in (4). The butterworth filter aims to have a maximally flat frequency response in the passband. The gain of a Butterworth low pass filter can be seen in Figure 2-2. Most of the Spike sorting literature suggests a bandpass filter for the pre-processing step so we can modify our lowpass filter to behave like a bandpass filter by applying the transformation in Figure 2-3 from (11) to the gain in Figure 2-2.

$$G_n(w) = |H_n(jw)| = G_o/\sqrt{1 + (w/w_c)^{2n}}$$

Figure 2·2: Gain for a lowpass Butterworth Filter

$$w/w_c => (w_o/(w_2 - w_1)) * (w/w_o - w_o/w)$$

Figure 2·3: Transformation from lowpass to bandpass. The original quantity is the normalized frequency. w0 is the center frequency while w1 and w2 are the low cutoff and high cutoff for the basspand.

The second type of filter considered is the Chebyshev filter, named after Pafnuty Chebyshev due to the filter being derived from Chebyshev polynomials. There are two kinds of Chebyshev filters, referred to in most texts as Type I and Type II. A Type I filter allows equiripple in the passband and a maximally flat response in the stopband, while Type II is the opposite with a maximally flat response in the passband and allowing equiripple in the stopband. The Chebyshev polynomials can be derived from the recursion as seen in Figure 2-4. The respective gain of a lowpass Chebyshev filter can be expressed in terms of this recursion as seen in Figure 2-5. We modify Figure 2-5 by replacing the normalized frequency with Figure 2-3 as in the Butterworth filter case.

$$T_{n+1} = 2 * x * T_n - T_{n-1}$$

Figure 2·4: The recursion for deriving a Chebyshev polynomial

$$G_n(w) = |H_n(jw)| = 1/(\sqrt{1 + (c^2 T_n^2(w/w_0))^e})$$

Figure 2·5: The gain of a Chebyshev filter. Here e=1 if the filter is Type I and e= -1 if the filter is Type II

In Figure 2-6 we can see an example of some raw ECoG data to be filtered. In Figure 2-7 we see the result of a 3rd order Butterworth filter applied to it with a DC gain of 0.45. In Figure 2-8 the same raw data from Figure 2-6 filtered by a 2nd order Type I Chebyshev filter with 1 dB of passband ripple. For both filters the passband was 300Hz - 3000Hz. Comparing Figures 2-7 and 2-8, we observe that the results of both filters are very similar.

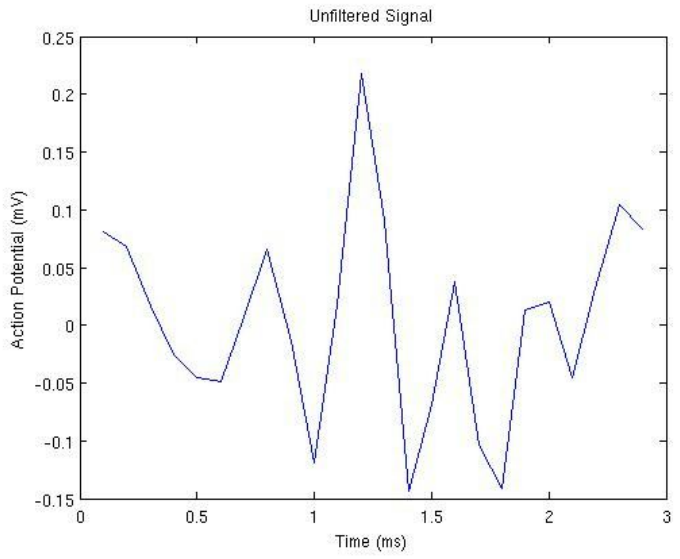


Figure 2·6: A sub-section of raw EEG data to be filtered

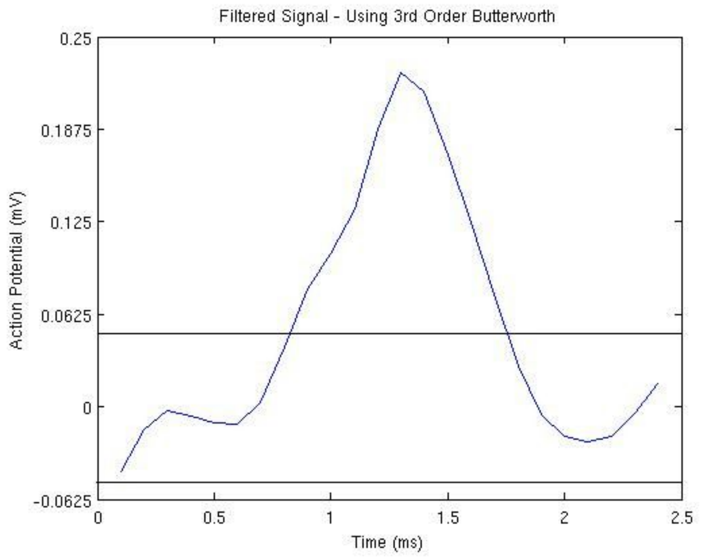


Figure 2·7: The data from Figure 2-6 filtered by a 3rd order Butterworth Filter. The black line is a minimum for the peak threshold. The maximum of the threshold in this case is larger than the local maxima

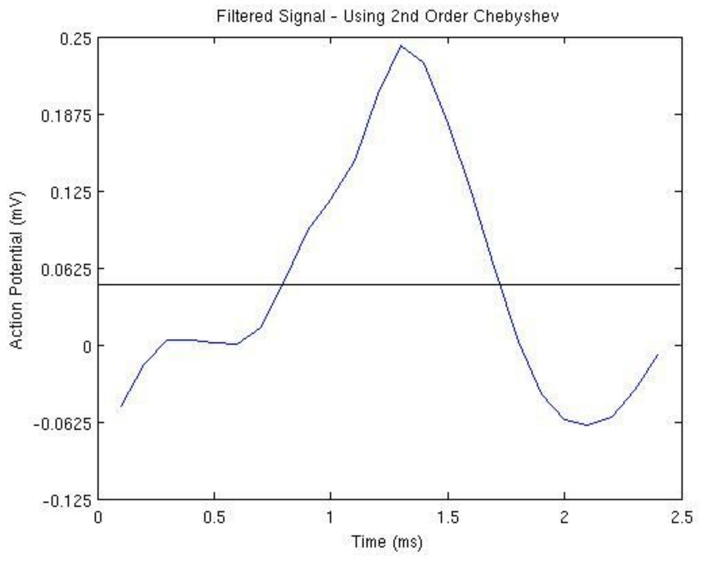


Figure 2·8: The data from Figure 2-6 filtered by a 2nd order Chebyshev Filter. The black lines are minimums for the peak threshold. The maximum of the threshold in this case is larger than the local maxima

2.3 Spike Detection

After the raw data has been filtered, the next step is to perform amplitude thresholding along the filtered data to extract relevant waveforms that correspond to neural spikes. Amplitude thresholding involves picking a minimum value that the sample must be equal to or greater than in order to be considered a neural spike for the rest of the algorithm. To minimize the amount of electrode artifacts i.e. any abrupt changes in the signal that are believed to not correlate to neural activity (an electrode being touched, moved, etc...) a maximum value is also imposed in some studies (8). In our implementation, a minimum of 0.05 mV and a maximum of 1mV are used. From there, we threshold the signal to detect subsets of the signal that fit within the threshold. Then the largest value within the subset is treated as the peak. Each detected peak and a small time segment surrounding will be treated as a waveform throughout the rest of the approach. The window we use is 3ms. The size of this window is arbitrary, and along with threshold values, isnt specified in most of the existing literature. We take all of these waveforms, superimpose them and treat that as the dataset in the clustering step later on. An example of the output of amplitude thresholding can be seen in Figure 2-9.

The dataset used in Figure 2-9 is known as the Pittsburgh dataset, a synthetic dataset derived from sine waves for the purpose of testing the focused mixture model approach in (1). The Pittsburgh dataset has 3310 samples of data and 33 features. Each sample is a single waveform and each feature is a segment of time along the x-axis in figure 6. The dataset we are mainly interested in is known as the Clarissa dataset. This dataset comes from an experiment performed on primates at the Massachusetts General Hospital (MGH) in 2013. Each primate has two electrodes implanted into it. During the recording session the primates are shown an object and then immediately after asked to to select that object among a lineup of objects. After applying amplitude thresholding to our Clarissa dataset we have 1781 samples of data and 32 features. From here we are ready to consider applying different clustering techniques.

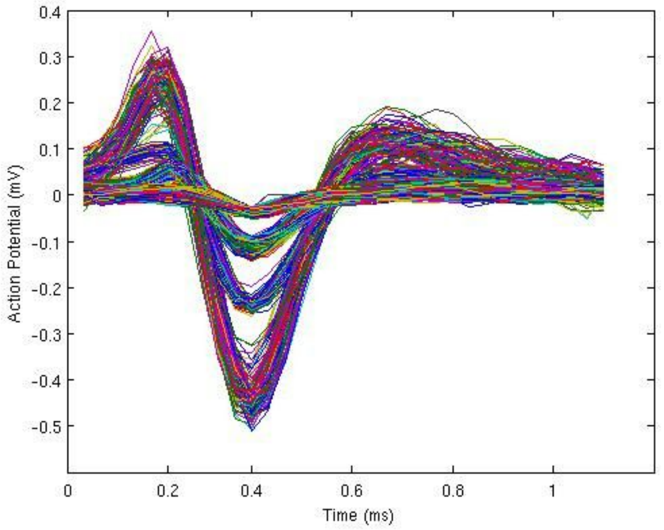


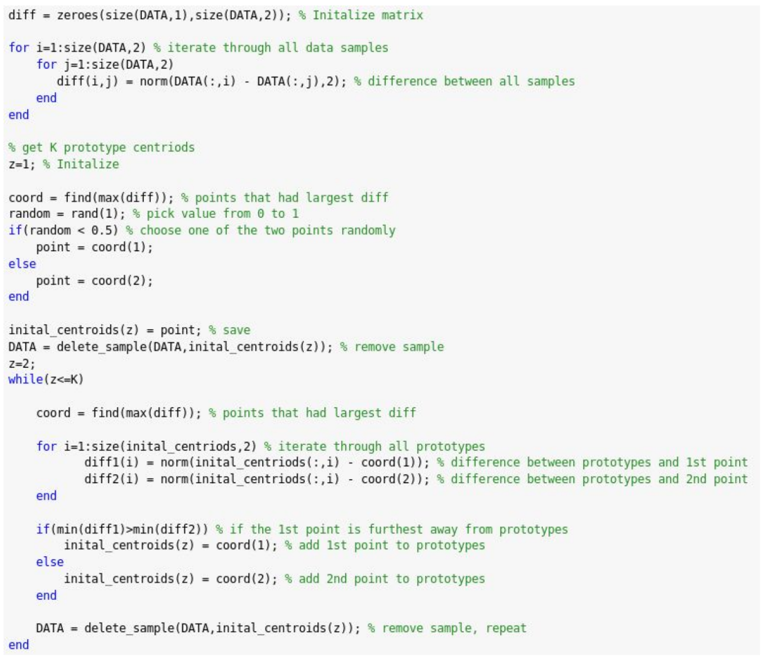
Figure 2·9: Filtered EEG Data ready to be clustered. This dataset has clear gaps between each group of spikes.

2.4 Clustering

Now that we have successfully pre-processed that data, the next and final stage is to use unsupervised learning in order to see some underlying structure. In this section we will discuss three clustering algorithms and then apply them to the data; k-means Clustering, Spectral Clustering, and Self-Tuning Spectral Clustering.

The first clustering algorithm is known as k-means Clustering (7). To perform k-means Clustering, we first pick a scalar value for k which represents the expected number of clusters. We place these initial clusters as to maximize the distance between each cluster. In our implementation, this is done by keeping track of the highest residual difference between samples and then pick a sample as to maximize the Euclidian difference between the initial cluster locations.

Here in Figure 2-10 we can see some example code. First we keep track of the difference between every data sample as seen in the first nested for loop. Next we randomly choose one the of the two candidate points that formed the maximum residual, this will be the first prototype location for the clusters. Next we exclude the chosen sample from the dataset. For choosing the remaining prototype locations, we find the maximum residual within the new dataset, but we want the point that is furthest away from the existing prototypes as well. We keep track of all the distances between the set of prototypes and the two candidate points as seen in the final for loop. We take the point from the residual that is furthest away from the prototypes as seen in the final if-else statement by comparing the min of each set of differences. We repeat this process until we have the required amount of prototype locations, as seen in the while loop.



```
diff = zeroes(size(DATA,1),size(DATA,2)); % Initalize matrix

for i=1:size(DATA,2) % iterate through all data samples
    for j=1:size(DATA,2)
       diff(i,j) = norm(DATA(:,i) - DATA(:,j),2); % difference between all samples
    end
end

% get K prototype centriods
z=1; % Initalize

coord = find(max(diff)); % points that had largest diff
random = rand(1); % pick value from 0 to 1
if(random < 0.5) % choose one of the two points randomly
    point = coord(1);
else
    point = coord(2);
end

inital_centroids(z) = point; % save
DATA = delete_sample(DATA,inital_centroids(z)); % remove sample
z=2;
while(z<=K)

    coord = find(max(diff)); % points that had largest diff

    for i=1:size(inital_centriods,2) % iterate through all prototypes
           diff1(i) = norm(inital_centriods(:,i) - coord(1)); % difference between prototypes and 1st point
           diff2(i) = norm(inital_centriods(:,i) - coord(2)); % difference between prototypes and 2nd point
    end

    if(min(diff1)>min(diff2)) % if the 1st point is furthest away from prototypes
        inital_centroids(z) = coord(1); % add 1st point to prototypes
    else
        inital_centroids(z) = coord(2); % add 2nd point to prototypes
    end

    DATA = delete_sample(DATA,inital_centroids(z)); % remove sample, repeat
end
```

Figure 2·10: Choosing Prototype locations in the k-means clustering algorithm

We then attribute each point (in our case a single sample from the waveforms) to the cluster containing the closest centroid. Next we recalculate the cluster centroid which is defined as the mean of all points in that cluster. In this case because our data resides in N-dimensional space where N=32 time segments, we take the average of all the samples in each dimension and treat the resulting point as the running mean. We then repeat the previous two steps (attribute and re-calculate) until convergence.

In Figures 2-11, 2-12, and 2-13 we see the results of applying k-means Clustering to the Clarissa dataset. In Figures 2-14, 2-15, and 2-16 we see the same clusters represented in two dimensions by applying Principal Component Analysis (PCA) to the waveforms and then plotting the first two principal components on each axis. The chosen values of k are 2 ,3, and 4. The Clarissa dataset is ECoG data provided by MGH from a recording session of primates performing a range of tasks.

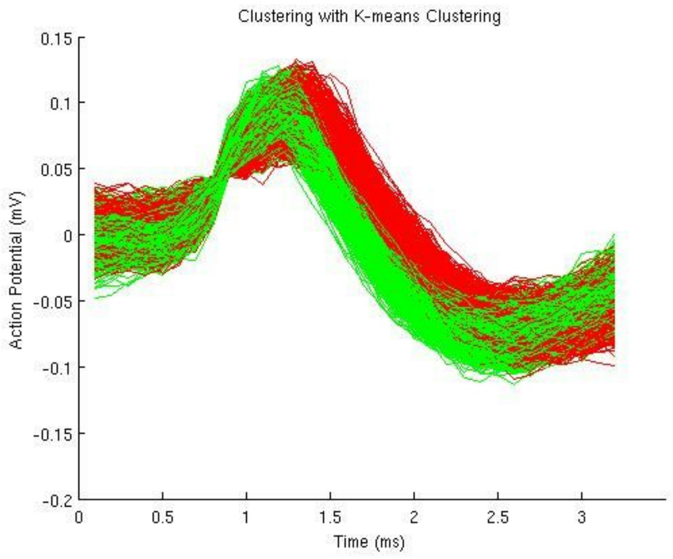


Figure 2·11: K-means clustering applied to Clarissa dataset. K=2

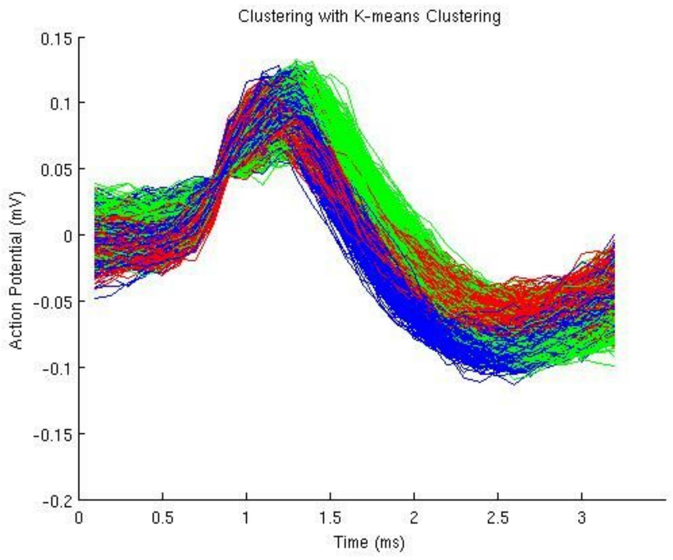


Figure 2·12: K-means clustering applied to Clarissa dataset. K=3

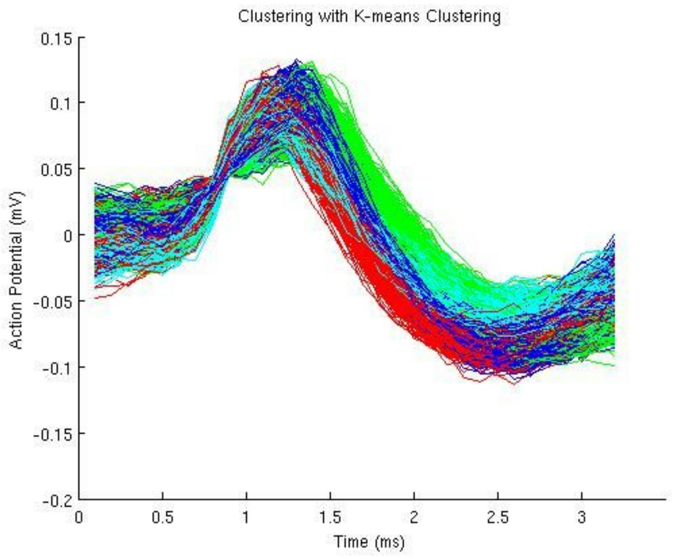


Figure 2·13: K-means clustering applied to Clarissa dataset. K=4

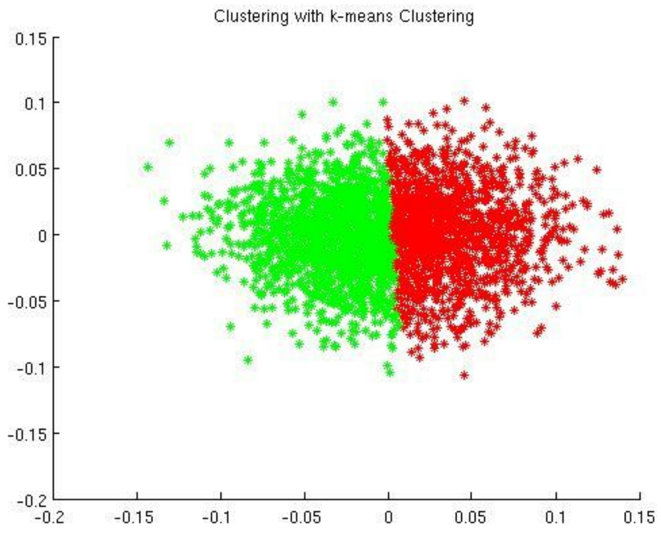


Figure 2·14: The clusters in Figure 2-11 visualized through PCA. Each axis is the 1st and 2nd Principal Component respectively.

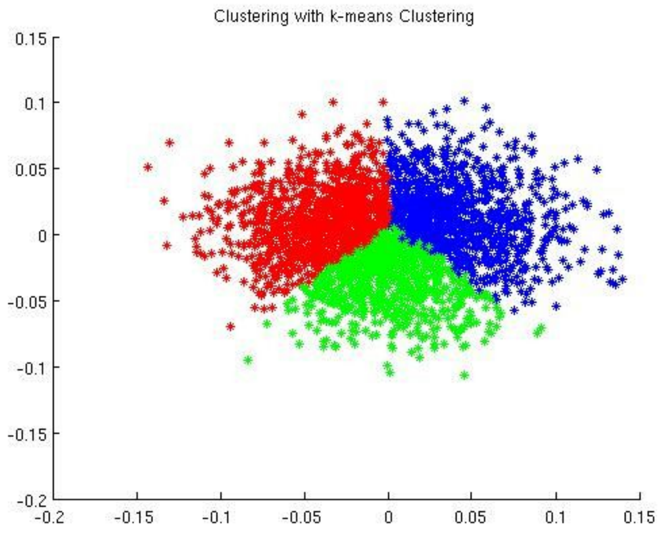


Figure 2·15: The clusters in Figure 2-12 visualized through PCA. Each axis is the 1st and 2nd Principal Component respectively.

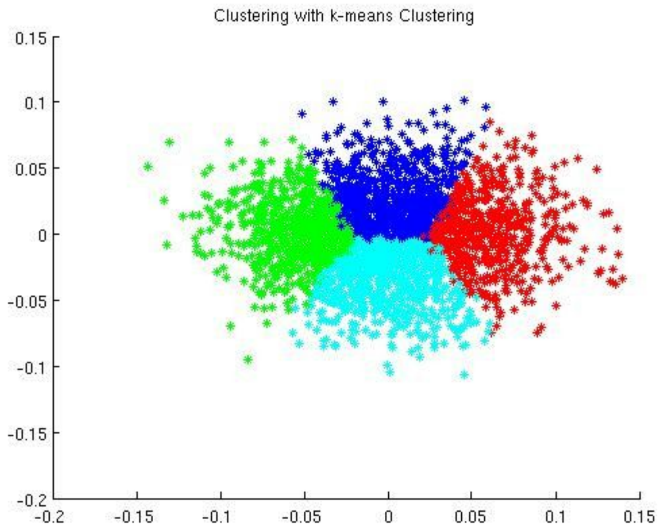


Figure 2·16: The clusters in Figure 2-13 visualized through PCA. Each axis is the 1st and 2nd Principal Component respectively.

The next clustering algorithm applied to the data is known as Spectral Clustering. Spectral Clustering involves first building an Affinity matrix S to gauge the pairwise interactions of all data samples. Each entry of the Affinity matrix is determined by a predefined similarity metric. A common similarity metric is described in equation 4 from (5). In Figure 2-17 σ is known as a decay parameter (set to 0.5 in our simulation) and the d(x,y) of two samples in this context is the Euclidean distance between x_i and x_j. The next step is to build a diagonal degree matrix D, described in Equation 2-18, and a normalized Affinity matrix Lnorm as seen in Equation 2-19.

$$s(i,j) = e^{-d(x_i,x_j)^2/2\sigma^2}$$

Figure 2·17: Similarity metric described in (5). σ is known as a decay parameter

$$D_{ii} = \sum_{j=1}^{n} S_{ij}$$

Figure 2·18: Diagonal Degree Matrix as described in (10).

$$L_{norm} = D^{-1/2} S D^{-1/2}$$

Figure 2·19: Normalized Affinity matrix L. D is the Diagonal Degree matrix of the data and S is the similarity matrix described by Figure 2-17.

The next step is to find the k eigenvectors of Lnorm associated with the k highest eigenvalues. Then an n x k matrix is defined (denoted by E) where each column is an eigenvector, in order of decreasing eigenvalue and each row corresponds to a scalar within the eigenvector. This E matrix is then re-normalized by the formula in Figure 2-20 and at this point becomes the matrix U. Finally each row in the Matrix U is treated as a data sample and clustered using the k-means algorithm.

$$u_{ij} = (e_{i,j}/\sqrt{\sum_k e_{ik}^2})$$

Figure 2·20: The matrix U after the rows are normalized.

In Figure 2-21, Figure 2-22, and Figure 2-23, Spectral Clustering is applied to the Clarissa dataset. Here the values of k are 2,3, and 4. The base code comes from a tutorial on Spectral Clustering (10), which was modified as needed for clustering the neural data. In Figures 2-24, Figure 2-25, and Figure 2-26, the same clusters can be seen in a two-dimensional representation via PCA, similar to Figures 2-21, 2-22, and 2-23.

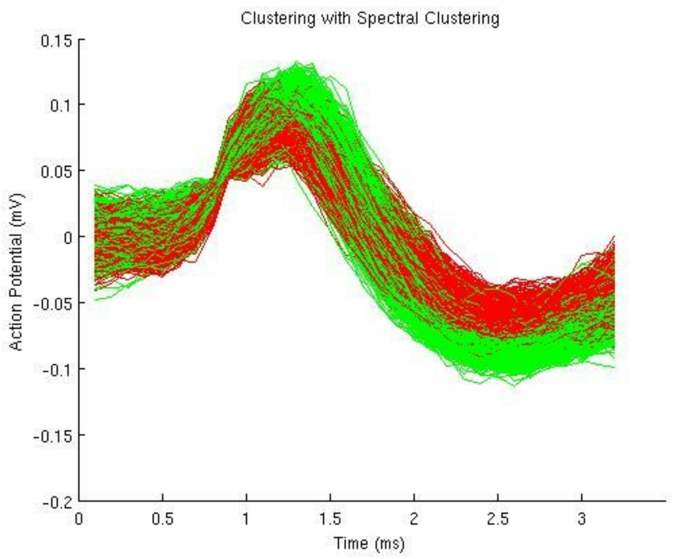


Figure 2·21: Spectral Clustering applied to Clarissa dataset. K=2

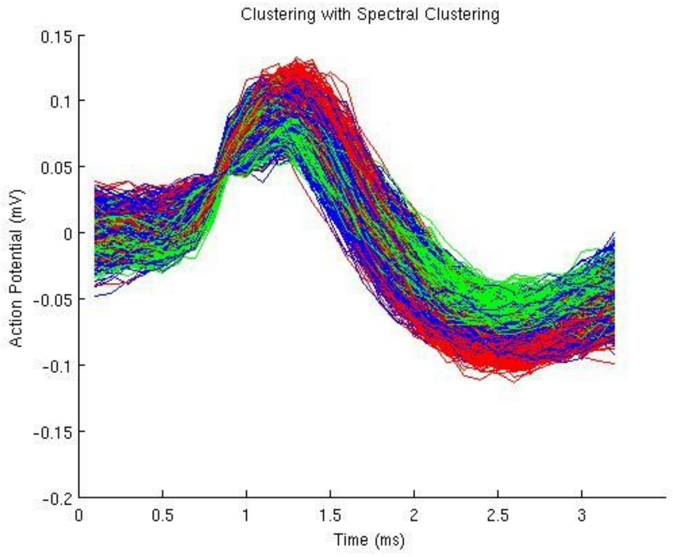


Figure 2·22: Spectral Clustering applied to Clarissa dataset. K=3

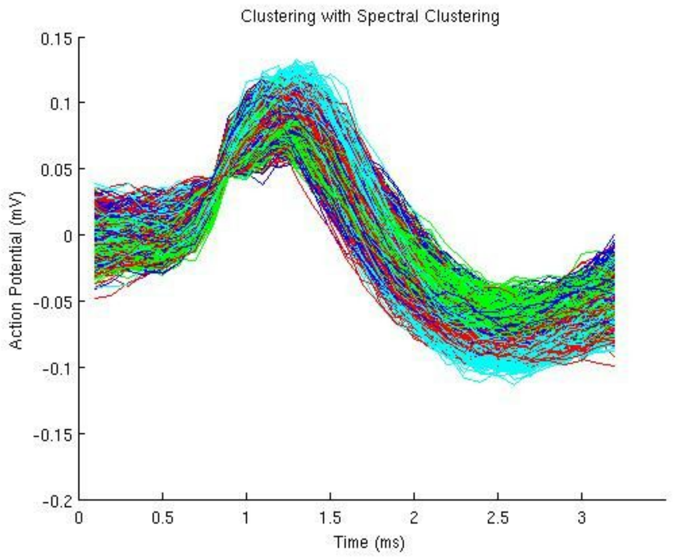


Figure 2·23: Spectral Clustering applied to Clarissa dataset. K=4

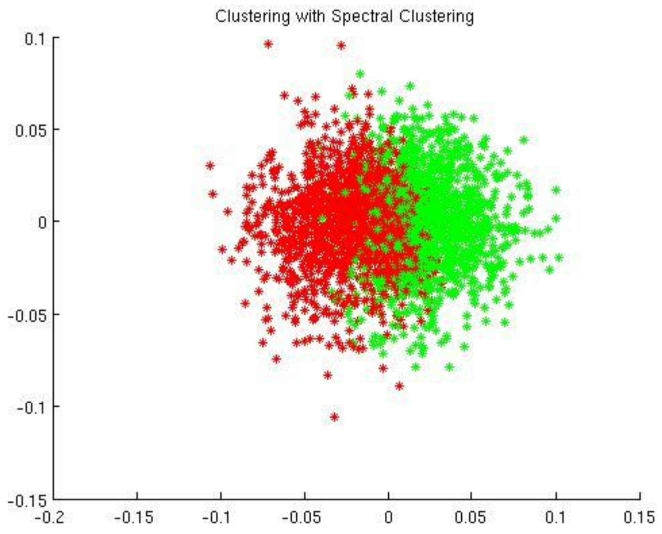


Figure 2·24: The clusters in Figure 2-21 visualized through PCA. Each axis is the 1st and 2nd Principal Component respectively

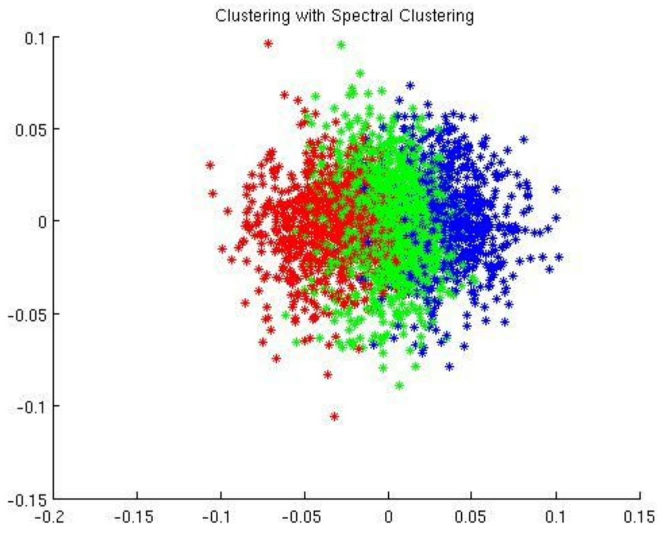


Figure 2·25: The clusters in Figure 2-22 visualized through PCA. Each axis is the 1st and 2nd Principal Component respectively

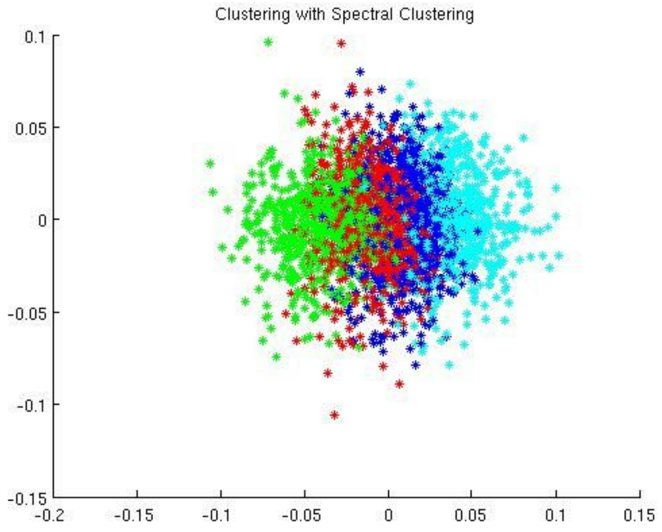


Figure 2·26: The clusters in Figure 2-23 visualized through PCA. Each axis is the 1st and 2nd Principal Component respectively

Comparing Figures 2-11, 2-12, and 2-13 to Figures 2-21, 2-22, and 2-23 we notice much more overlap in clusters generated in Spectral Clustering then we do with clusters generated from K-means clustering. This behavior occurs throughout all three values of k, while in k-means the amount of overlap seems to be minimal. Also it is important to note that because Spectral Clustering applies k-means clustering as its final step, its run time will most likely be longer. The difference in cluster results motivates us to try a different variant of Spectral Clustering, namely Self-Tuning Spectral Clustering (10).

The third clustering approach we will try is very similar to Spectral Clustering as weve previously defined it. The key difference is in the decay parameter . Previously the decay parameter was a scalar that remained constant throughout the whole affinity matrix, but in Self-Tuning Spectral Clustering the concept of local scaling is introduced (10) where the value of changes for each data sample s_i. The distance between two data samples s_i and s_j as seen by s_i is the euclidean distance of the two divided by i, a scaling parameter associated with s_i as seen in Figure 2-27. The distance between two data samples s_i and s_j as seen by s_j is the converse of Figure 2-27 as seen in Figure 2-28. The squared distance between two samples can be generalized to resemble Figure 2-29 and thus the affinity between two samples is described in Figure 2-30.

$$d(s_i, s_j)/\sigma_i$$

Figure 2·27: The distance between two data samples s_i and s_j as seen by s_i

$$d(s_j, s_i)/\sigma_j$$

Figure 2·28: The distance between two data samples s_j and s_j as seen by s_j

$$d(s_j, s_i) * d(s_j, s_i) = d^2(s_i, s_j)/\sigma_i\sigma_j$$

Figure 2·29: The squared distance between two samples s_i and s_j

$$A_{ij} = exp(-d^2(s_i, s_j)/\sigma_i\sigma_j)$$

Figure 2·30: The affinity between two samples s_i and s_j

Local scaling involves using a specific σ for each pair of samples to account for the local statistics of the neighborhoods surrounding points si and sj (10). One way of performing local scaling is seen in Figure 2-31 where d(x,y) is the Euclidean distance between s_i and s_k and s_k is the K-th nearest neighbor of s_i.

$$\sigma_i = d(s_i, s_k)$$

Figure 2·31: Local scaling described in (10). A value for is calculated for each value in the Affinity matrix S.

The results of applying Self-Tuning Spectral Clustering can be seen in Figure 2-32, Figure 2-33, and Figure 2-34. Figure 2-35, Figure 2-26, and Figure 2-37 depict the same clusters in a two-dimensional representation via PCA, similar to Figures 2-14, 2-15, and 2-16.

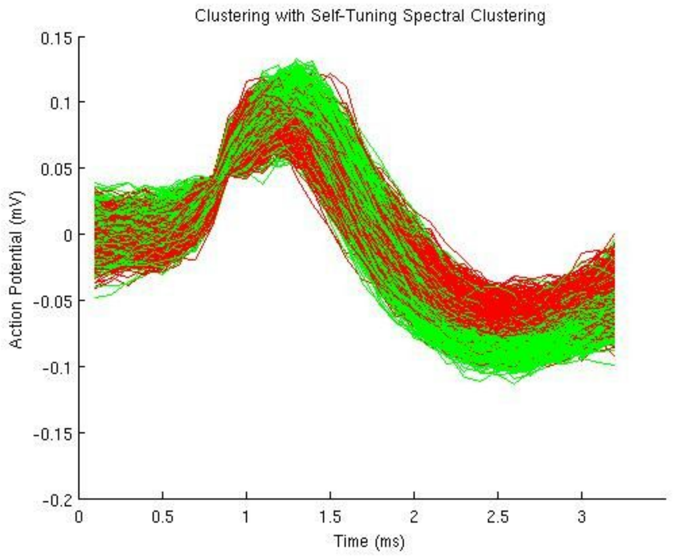


Figure 2·32: Self-Tuning Spectral Clustering applied to the Clarissa dataset. K=2

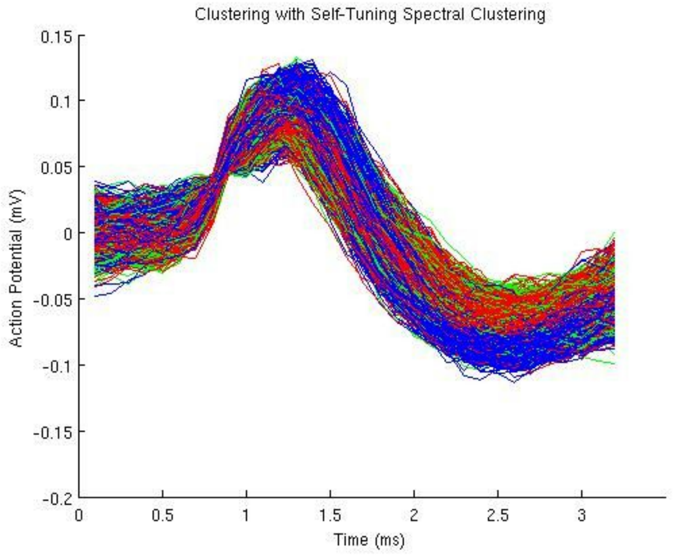


Figure 2·33: Self-Tuning Spectral Clustering applied to the Clarissa dataset. K=3

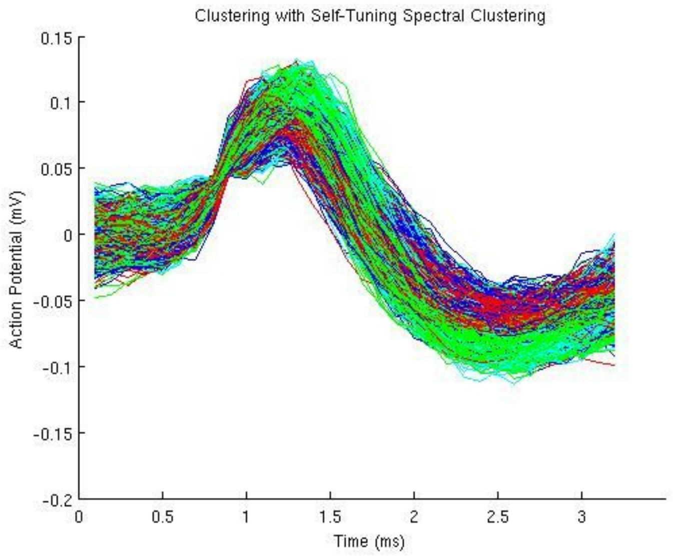


Figure 2·34: Self-Tuning Spectral Clustering applied to the Clarissa dataset. K=4

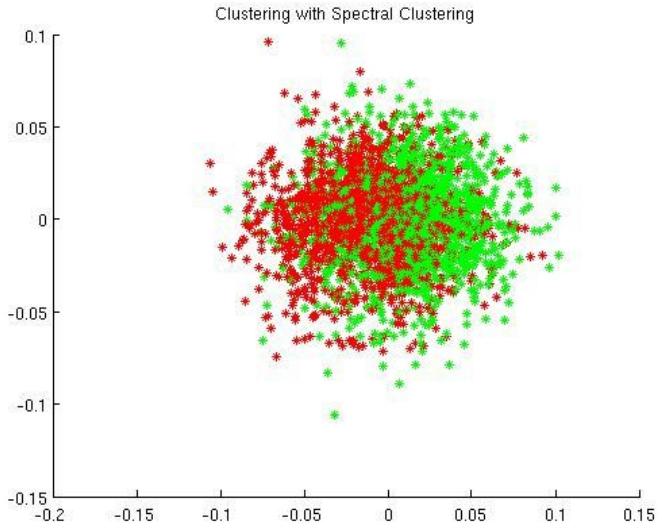


Figure 2·35: The clusters in Figure 2-32 visualized through PCA. Each axis is the 1st and 2nd Principal Component respectively.

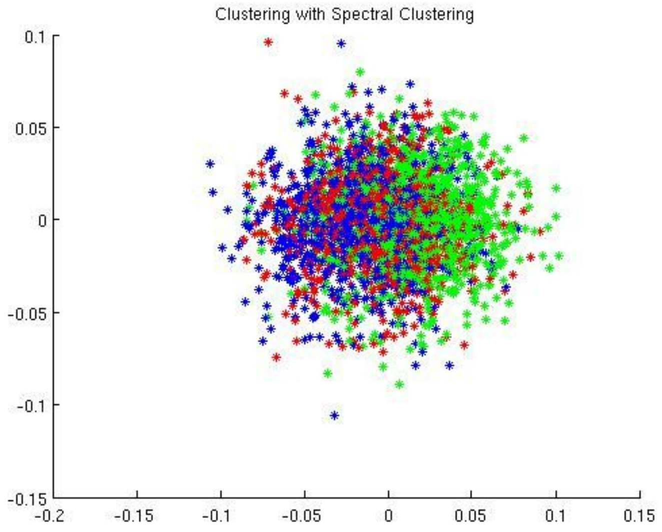


Figure 2·36: The clusters in Figure 2-33 visualized through PCA. Each axis is the 1st and 2nd Principal Component respectively.

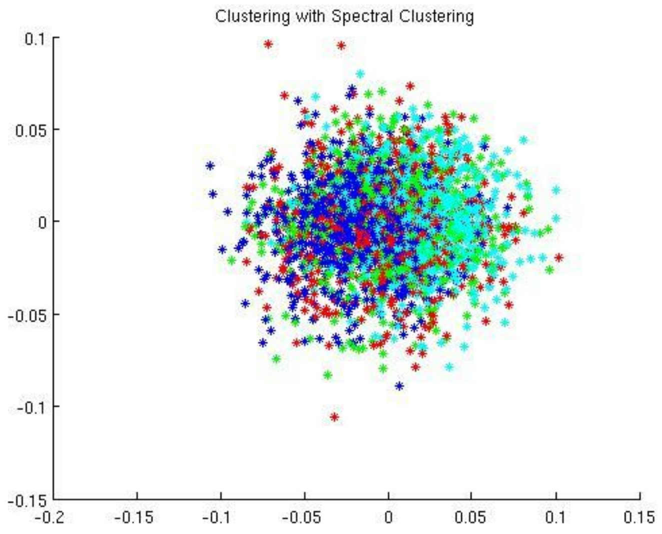


Figure 2·37: The clusters in Figure 2-34 visualized through PCA. Each axis is the 1st and 2nd Principal Component respectively.

When comparing the clusters of two-dimensional form to the original version of spectral clustering, the overlap in clusters does not seem to have decreased, in fact the overlap seems to have increased. Its important to note that the clusters analyzed in (10) typically had noticeable gaps between them while in the case of the Clarissa dataset there seems to be an inherent overlap in the clusters. Even in the k-means case, which initializes its centroids to be maximally far away, results in clusters that touch at their borders as seen in Figures 10-12. Because this variant of Spectral Clustering requires a value for the decay parameter to be calculated for every sample, (which involves calculating the k-th nearest neighbor of every data sample) it is likely that Self-Tuning Spectral Clustering will have a larger time complexity than its counterpart.

Table 2.1 lists the average distance from a data sample to its respective cluster center as well as the distance between clusters. The distance between clusters seems to be the largest for k-means. For cluster tightness i.e. how close the points are to the center, one of the clusters in Spectral Clustering seems to be much larger than the clusters in k-means and Self-Tuning Spectral Clustering. Interestingly, the tightness of clusters in self-tuning spectral clustering is comparable to k-means.

Table 2.1: Cluster Metrics for k-means, Spectral, and Self-Tuning Spectral Clustering when k=2.

	k-means Clustering	Spectral Clustering	Spectral Clustering (Self Tuning)
Average distance for Cluster 1	0.0582	0.612	0.0692
Average distance for Cluster 2	0.0615	0.0683	0.0652
Distance between Cluster 1 and 2	0.0688	0.0412	0.0206

Table 2.2 lists the average distance within a cluster as well as distance between clusters for when k=3. In this case the average distance within a cluster seems to be similar for all three clustering algorithms. In terms of distance between clusters, its important to note that the distance is much less for Spectral and for Self-Tuning Spectral vs k-means Clustering.

Table 2.2: Cluster Metrics for k-means, Spectral, and Self-Tuning Spectral Clustering when k=3.

	k-means Clustering	Spectral Clustering	Self-Tuning Spectral Clustering
Average distance for Cluster 1	0.0563	0.0670	0.0654
Average distance for Cluster 2	0.0678	0.0633	0.0701
Average distance for Cluster 3	0.0627	0.0677	0.0661
Distance between Cluster 1 and 2	0.0495	0.0144	0.0142
Distance between Cluster 2 and 3	0.0387	0.0162	0.0131
Distance between Cluster 1 and 3	0.0319	0.0294	0.0028

Table 2.3 lists the average distance within a cluster as well as distance between

clusters for when k=4. In this case the average distance within a cluster seems to be similar for all three clustering algorithms. In terms of distance between clusters, its important to note that the distance for k-means Clustering is noticeably larger than in Spectral Clustering and Self-Tuning Spectral Clustering. The distances between cluster centers seems to be slightly lower in Self-Tuning when compared to Spectral Clustering.

Table 2.3: My caption

	k-means Clustering	Spectral Clustering	Self-Tuning Spectral Clustering
Average distance for Cluster 1	0.0563	0.0680	0.0666
Average distance for Cluster 2	0.0678	0.0640	0.0689
Average distance for Cluster 3	0.0643	0.0617	0.0682
Average distance for Cluster 4	0.0643	0.0637	0.0663
Distance between Cluster 1 and 2	0.0495	0.0113	0.0117
Distance between Cluster 1 and 3	0.0378	0.0221	0.0116
Distance between Cluster 1 and 4	0.0378	0.0342	0.0025
Distance between Cluster 2 and 3	0.0328	0.0109	0.0007
Distance between Cluster 2 and 4	0.0321	0.0235	0.0136
Distance between Cluster 3 and 4	0.0302	0.0133	0.0135

Chapter 3

Conclusion

3.1 Conclusion

In this MS thesis we analyze ECoG data using signal processing and machine learning techniques in order to contribute to the TRANSFORM-DBS project at the Charles Stark Draper Laboratory. We assert to treat the problem with unsupervised learning techniques because we do not have ground truth to the supposed hidden mechanisms involved in cognitive behavior. In the Clarissa dataset it appears that both a Butterworth and a Chebyshev filter provide similar filtering results. In terms of clustering the Clarissa dataset, when only looking at the clusters as colored waveforms, in both k-means and Spectral, there is a noticeable pattern of two averaged peaks when k=2. When k=3 or 4, the results of k-means clustering continue to yield clusters with minimal overlap while the spectral clustering results are inconclusive. For all three cases of Self-Tuning Spectral Clustering the resulting clusters seem to overlap. When observing the clusters in a two-dimensional representation, there is noticeable overlap between clusters in Spectral Clustering and Self-Tuning Spectral Clustering as opposed to K-means.

Future work in clustering this data could involve using different clustering algorithms as well trying different approaches for feature extraction. Future work in the overall DBS project would be to find a way to correlate the resulting clusters to recorded patient activity. The clustering efforts of this Thesis focused on the shape of waveforms as opposed to when and in what sequence they occur (i.e. the stochastic aspect of the neural data). Approaches that account for changes over time may prove to give more insight on the data, but the overall project needs to strike a balance between a comprehensive characterization of the data and an approach that can be energy/time efficient.

References

1. Carlson, David E., et al. "Multichannel Electrophysiological Spike Sorting via Joint Dictionary Learning and Mixture Modeling."
arXiv preprint arXiv:1304.0542(2013).

2. Calabrese, Ana, and Liam Paninski. "Kalman filter mixture model for spike sorting of non-stationary data."
Journal of neuroscience methods 196.1 (2011): 159-169.

3. Quiroga, Rodrigo, Zoltan Nadasdy, and Yoram Ben-Shaul. "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering."
Neural computation 16.8 (2004): 1661-1687.

4. Butterworth, Stephen. "On the theory of filter amplifiers."
Wireless Engineer 7.6 (1930): 536-541.

5. Tademir, Kadim. "Vector quantization based approximate spectral clustering of large datasets."
Pattern Recognition 45.8 (2012): 3034-3044.

6. Hill, N. Jeremy, et al. "Recording human electrocorticographic (ECoG) signals for neuroscientific research and real-time functional cortical mapping."
Journal of visualized experiments: JoVE 64 (2012).

7. Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm."
Applied statistics (1979): 100-108.

8. Gwin, Joseph T., et al. "Removal of movement artifact from high-density EEG recorded during walking and running."
Journal of neurophysiology 103.6 (2010): 3526-3534.

9. Von Luxburg, Ulrike. "A tutorial on spectral clustering."
Statistics and computing 17.4 (2007): 395-416.

10. Zelnik-Manor, Lihi, and Pietro Perona. "Self-tuning spectral clustering."
Advances in neural information processing systems. 2004.

11. Stiles, Jim. Filter Transformations University of Kansas, Dept of EECS. 19 Jun. 2015. PDF file.

CURRICULUM VITAE

[redacted]