Iowa Research Online
The University of Iowa's Institutional Repository

University of Iowa
Iowa Research Online

Theses and Dissertations

Summer 2011

A feature-based algorithm for spike sorting involving intelligent feature-weighting mechanism

Kaustubh Anil Patwardhan
University of Iowa

Copyright 2011 Kaustubh Anil Patwardhan

This thesis is available at Iowa Research Online: http://ir.uiowa.edu/etd/1253

Recommended Citation
Patwardhan, Kaustubh Anil. "A feature-based algorithm for spike sorting involving intelligent feature-weighting mechanism." MS (Master of Science) thesis, University of Iowa, 2011.
http://ir.uiowa.edu/etd/1253.

Follow this and additional works at: http://ir.uiowa.edu/etd


Part of the Electrical and Computer Engineering Commons

A FEATURE-BASED ALGORITHM FOR SPIKE SORTING INVOLVING INTELLIGENT FEATURE-WEIGHTING MECHANISM

by

Kaustubh Anil Patwardhan

A thesis submitted in partial fulfillment
of the requirements for the Master of
Science degree in Electrical and Computer
Engineering in the Graduate College of
The University of Iowa

July 2011

Thesis Supervisor: Professor Er-Wei Bai

Copyright by

KAUSTUBH ANIL PATWARDHAN

2011

All Rights Reserved

Graduate College
The University of Iowa
Iowa city, Iowa

CERTIFICATE OF APPROVAL

MASTER'S THESIS

This is to certify that the Master's thesis of

Kaustubh Anil Patwardhan

has been approved by the Examining Committee for the thesis requirement for the Masters of Science degree in Electrical and Computer Engineering at the July 2011 graduation.

Thesis Committee: _____
Er-Wei Bai, Thesis Supervisor

_____
Steven Stasheff

_____
Punam Saha

ACKNOWLEDGEMENT

It gives me great pleasure to present my Master's thesis titled 'A feature-based algorithm for spike sorting involving intelligent feature-weighting mechanism' in partial fulfillment of my MS degree in Electrical and Computer Engineering.

First and foremost, I would like to express profound gratitude to my advisor, Professor ErWei-Bai, for his invaluable support, encouragement, supervision, patience and useful suggestions throughout the research work. His moral support and continuous guidance enabled me to complete my work successfully.

Special thanks to Dr. Steven Stasheff for providing all the data and all his time and efforts. I would like to thank Zhizun Cai for all his help and guidance throughout the project. This work would not have been possible without Zhijun's help.

I would like to thank the members of my committee, Dr. Stasheff and Professor Punam Saha for taking time off from their busy schedule.

Last but not the least I would like to thank my family and all my friends for their love and support during the project and throughout my life.

TABLE OF CONTENTS

APPENDIX

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

BACKGROUND AND MOTIVATION

This chapter provides the background, motivation and the organization of the thesis.

1.1 Retinitis Pigmentosa and the Retina

Retinitis Pigmentosa (RP) is a group of inherited retinal degenerations that lead to chronic blindness [1]. The retina is a complex light sensitive tissue that lines the inner surface of the eye. It consists of several layers of neurons. The photoreceptor cells are the set of neurons that are directly sensitive to light. They are mainly of two types: the rods and cones. Rods provide black-and-white vision and function mainly in dim light while cones help us perceive colors and support daytime vision. Retinal ganglion cells are a recently discovered type of photoreceptor cells located near the inner surface of the retina that provide responses to global bright daylight. Light striking the retina produces neural signals from the rods and cones which are further processed by other retinal neurons [2]. This leads to action potentials in the retinal ganglion cells.

Action potential refers to the rapid rise and fall of the electric membrane potential of a cell. The cell's plasma membrane consists of voltage gated ion channels which generate these action potentials [3]. These channels are shut when the membrane potential is near the resting potential of the cell, but they rapidly begin to open if the membrane potential increases to a precisely defined threshold value. When the channels open, they allow an inward flow of sodium ions, which changes the electrochemical gradient, which in turn produces a further rise in the membrane potential. This then causes more channels to open, producing a greater electric current. The process proceeds explosively until all of the available ion channels are open, resulting in a large upswing in the membrane potential. The rapid influx of sodium ions causes the polarity of the plasma membrane to reverse, and the ion channels then rapidly inactivate. As

the sodium channels close, sodium ions can no longer enter the neuron, and they are actively transported out of the plasma membrane. Potassium channels are then activated, and there is an outward current of potassium ions, returning the electrochemical gradient to the resting state. Observing this activity of different neurons can help us understand any changes occurring in the morphology of the retina.

People with RP usually suffer from night blindness and tunnel vision caused by photoreceptor degeneration. Abnormalities of the photoreceptors or the retinal pigment epithelium (RPE) of the retina lead to progressive visual loss. Its progression varies in each individual [4]. This retinal degeneration causes changes in the anatomy of outer retinal pathways as well as the morphology of the inner retina. These changes in the inner retinal physiology are associated with increased spontaneous activity and burst firing of the ganglion cells. Blindness occurs due to the loss of photoreceptors and other inner retinal cells in some cases but the ganglion cells still remain viable for months despite this activity [5].

1.2 Treatment for RP

Based on the different stages of the disease, four major approaches to the treatment of patients with R.P have been described [6]. The first method corrects the biochemical abnormalities that cause vision loss while some photoreceptors remain structurally intact. An example is correction of an abnormality of the visual (retinoid) cycle in the form of retinitis pigmentosa known as Leber's congenital amaurosis — specifically, the subtype caused by mutations in the gene encoding RPE65. Subretinal gene therapy, in which a normal RPE65 gene was delivered under the retina by intraocular injection, has successfully and safely corrected dysfunction of the visual cycle in patients [7] [8]. The mechanism underlying Leber's congenital amaurosis is complex, and there also is a component of progressive retinal degeneration. It is not yet known if the correction of the visual-cycle defect halts or slows this degeneration.

A second approach to the treatment of R.P. focuses on slowing the progressive degeneration of photoreceptors. It involves using neurotrophic factors, nutritional supplements and other pharmaceutical agents that improve the viability of neurons by inhibiting proapoptotic pathways, activating antiapoptotic signaling, reducing the production of retinotoxic molecules, and limiting oxidative damage.

Two other therapeutic approaches have been tried on animal subjects with advanced stages of RP where there are few or no functional photoreceptors remaining. One approach focuses on regenerating lost photoreceptors by means of transplantation or genetic manipulation of nonphotoreceptor retinal cell types, such as glia. The other approach involves creating electrical signals in the visual pathway that substitute for the usual input from photoreceptors.

The success of most of these treatment approaches largely depends on the preservation of a morphologically and functionally intact inner retina [5]. In order to decide a treatment plan, one needs to investigate the activity of these neurons to assess photoreceptor functionality and how well the inner retina has been preserved.

A study on retinal degeneration (rd) mice showed that even as the animal turns blind, the retinal ganglion cells maintain a high level of activity for many weeks [5]. The activity of these neurons can be recorded using multiple electrodes. However, each electrode records waveforms from different neurons and we need to distinguish spikes produced by different neurons to examine the activity of each neuron.

1.3 Spike sorting

Spike sorting refers to the process of assigning these action potentials to different neurons. Complex brain processes are reflected by the activity of large neural populations and that the study of single-cells in isolation gives only a very limited view of the whole picture [9] [10]. Further developments in this area rely to a large extent on the ability to record

simultaneously from large populations of cells. This problem is difficult because, a) the waveforms being classified are often noisy, b) waveforms coming from the same cell can vary in both shape and amplitude, depending on the previous activity of the cell and c) waveforms can overlap in time, resulting in even more complex waveforms. The implementation of optimal spike sorting algorithms is a critical step forward since it can allow the analysis of the activity of a few close-by neurons from each recording electrode. Distinguishing among the spikes produced by different neurons is critical in observing the activity of each neuron and detect any changes in it.

1.4 Previous work

Some of the previously reported approaches include manual sorting [11], k-means clustering [12], principal component analysis (PCA) [13] and support vector machines (SVM) [14]. However, some of these approaches depend heavily on inputs from human experts while others do not consider the inherent features of the neural data. This thesis presents a completely different approach by considering the various features of neural data and introducing an intelligent weighting mechanism to cluster the spikes without any human intervention. This new method also yields improved performance compared to the above mentioned approaches.

1.5 Thesis contribution:

This thesis aims to contribute the following:

- Calculate various features of the neural data based on the geometric properties of the waveforms
- Develop an intelligent weighting technique to weight the features
- Incorporate the features and weighting techniques into the K-means clustering algorithm

CHAPTER 2

EXISTING METHODS

Some of the previously reported methods to determine the activity of each neuron include manual sorting, K-means clustering and PCA. Following is a brief overview of these existing methods.

2.1 Manual Sorting

Manual sorting refers to a common and labor-intensive approach which involves manually identifying clusters by visual inspection. It involves grouping spikes with similar features into clusters, corresponding to the different neurons. This approach has the advantage that the human expert can apply information specific to the cells that are being studied. The waveforms or firing patterns of neurons can be compared with their known characteristics. This knowledge can also be used to decide criteria for including a particular spike to a cluster and evaluate the results of the overall clustering procedure. Human sorters use commercial software and various techniques such as PCA and manual cluster cutting [15] to label the spikes. These softwares provide the users with tools to sort all waveforms from a particular recording one channel at a time. This is usually achieved by manual cluster selection and refinement in a graphical display constructed by projecting the waveforms onto their first two principal components or other features.

There are, however, several disadvantages of this procedure because (1) you cannot differentiate between waveforms visually (2) the rigorous process scales poorly to experiments where large number of electrodes are used, (3) the results of this procedure are difficult to reproduce due to human biases, (4) it is difficult to design quality metrics to assess this subjective approach and (5) it is a very time-consuming task.

Significant variability among human sorters has previously been shown for recordings from tetrodes and single electrodes [16].Expert human spike sorters have shown widely varying performance on both real and synthetic neural datasets. On real data, subjects differed not only in what constituted a spike versus noise but even in the number of units present in the data. These results point to the need for objective spike-sorting algorithms that provide consistency across experiments. Reduced human intervention can definitely improve the time taken for clustering.

2.2 K-means Clustering

K-means algorithm is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. K-means defines a prototype in terms of a centroid, which is usually the mean of a group of points [17].

It is an algorithm to classify or to group objects based on attributes/features into k number of groups. K is positive integer number. The grouping is done by minimizing the sum of the squares of distances between data and the corresponding cluster centroid.

The basic step of k-means clustering involves determining the number of cluster K and assuming the centroid or center of these clusters. Initial centroids can be taken randomly from the data. Then the K means algorithm iteratively determines the centroid coordinate, the distance of each observation from the centroid and groups the observations based on minimum distance.

K-Means starts with a single cluster with its center as the mean of the data. This cluster is split into two and the means of the new clusters are iteratively obtained. These two clusters are again split and the process continues until the specified number of clusters is obtained. To assign a point to the closest centroid, we need a proximity measure to quantify ‘closest’ for the neural data. In other words, a clustering criterion has to be adopted. Euclidean distance is the most commonly used proximity measure. Here is summary of clustering process:

Algorithm: K-means

Given a dataset X with N observations and M variables, the algorithm searches for a seaparation of X into K clusters that minimizes the sum of within cluster distance of all variables.

Step 1 Determine number of clusters K

Step 2 Randomly select K distinct objects as the initial cluster centers.

Step 3 For each observation in X, calculate the distances between the observation and each cluster center based on a proximity measure such as Euclidean distance and assign the observation to the cluster with the shortest distance.

Step 4 Repeat Step 3 until all observations are assigned to clusters. For each cluster, compute a new cluster center as the mean (average) of the observations of that cluster.

Step 5 Compare the new cluster centers to the previous centers. Stop the process if the centers are the same; otherwise go back to Step 3.

The main weaknesses of K-means clustering are (1) the number of clusters k must be determined beforehand and (2) it is unknown as to which variable of the data contributes more to the clustering process since it is assumed that each variable has the same weight [17]. In order to overcome these difficulties, we need to develop an intelligent mechanism to determine the number of clusters and to weight the features based on their contribution to the clustering process.

2.3 Principal Component Analysis (PCA):

PCA is a very useful mathematical procedure in analyzing data. It is a method to identify various patterns in the data and express the data in such a way as to highlight the variability in the data [18]. It is also helpful in compressing the data by reducing the number of dimensions without much loss of information.

PCA converts a set of observations of possibly correlated variables into a set of values of uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has as high a variance as possible and each succeeding component in turn has the highest variance among the remaining components. The last component has the lowest variance. The dimensionality of the data can then be reduced by selecting only the first few principal components. Following is a summary of the process:

Algorithm: PCA

Given: A dataset X with N observations and M variables

Step 1 Organize the data into a matrix of M rows and N columns

Step 2 Calculate the mean of the data along each dimension M and subtract it from each column of the data matrix X.

Step 3 Calculate the MxM covariance matrix C of the mean-subtracted data.

Step 4 Determine the MxM eigenvector matrix V and the MxM diagonal matrix D of eigenvalues of the covariance matrix C.

Step 5 Sort the columns of the eigenvector matrix V and eigenvalue matrix D in order of decreasing eigenvalue.

Step 6 To reduce the dimensions of the original data, choose a subset of the eigenvectors that is to select the first L columns of the matrix V since the remaining components can be discarded due to their lower variance.

Step 7 Derive the transformed data by multiplying the transpose of the sorted matrix V (MxM) with the original mean subtracted dataset X (MxN).

$$Transformed\ data = V' \times X$$

Step 8 Follow K-means procedure to cluster the data

The main weakness of PCA is that it is very sensitive to outliers which are difficult to identify in some cases. The concepts of proximity and distance become less meaningful with increasing dimensionality and the process of subspace determination in PCA can be misled in the presence of noise and outliers. Assigning different weights to the observations of the data based on their relevancy can increase robustness of this method [19].

2.4 Support Vector Machines (SVM)

SVMs are a set of supervised learning methods used to analyze data and identify patterns to classify it. It takes a set of input data and predicts the class to which each observation belongs. It is a binary classifier which when given a training set, builds a model and assigns the new observations into one cluster or the other. The SVM model is a representation of the observations as points in space, mapped so that the observations of different clusters are clearly separated. New observations are then mapped into that same space and predicted to belong to a particular

cluster based on which side of the separation they fall on. Support vector machine constructs a hyperplane in a high dimensional space which can be used for classification. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any cluster. But in many cases, the points are separated by a nonlinear region. Rather than fitting nonlinear curves to the data, SVM handles this by using a kernel function to map the data into a different space where a hyperplane can be used to do the separation.

The use of kernels along with the absence of local minima is a major advantage of using SVMs. But the choice of the kernel function can also be a drawback. The most serious problem with SVMs is the high algorithmic complexity and extensive memory requirements [20].

CHAPTER 3

FEATURE BASED APPROACH

This chapter explains the proposed method and its working.

3.1 Data used

The data was obtained from simultaneous multiple electrode recordings of spontaneous and light-evoked extra- cellular action potentials from 30 to 90 retinal ganglion cells of rd1 mice [5]. The size of the data was 32xN where N is the number of observations and 32 is the number of time instants at which the activity was recorded. An observation refers to a complete waveform consisting of 32 data points. Nineteen such data sets were used with varying number of observations and number of clusters.

3.2 Features calculated

The original input data obtained from rd1 mice is 32 dimensional and it takes a lot of time to process it in some cases. Working with this huge data increases the cost of processing and also does not take into consideration, the inherent information present in the data. Each neuron tends to fire spikes of a particular shape. If the shape can be characterized then that information can be used to classify each spike. The spikes can be characterized by measuring different features. In general, the more features we have, the better it could be to distinguish among different spikes [21]. The proposed method calculates various features of the neural data obtained based on the geometric properties of the waveforms. These features reduce the dimensions of the data. They also capture the inherent information of the data and this reduced representation can then be used in the clustering procedure. All feature values are normalized. Following is a brief description of the features calculated from the data.

1. Magnitude in frequency domain (magF): This refers to the magnitude of each observation in the frequency domain and is obtained by taking the maximum of the absolute value of the Fast Fourier Transform of each observation.

$$magF(x) = \max(abs(fft(x))$$

where *x* is the observation.

2. Magnitude in time domain: This value is obtained by taking the difference of the maximum and minimum values of each observation. It is also called the peak-to-peak amplitude [22].

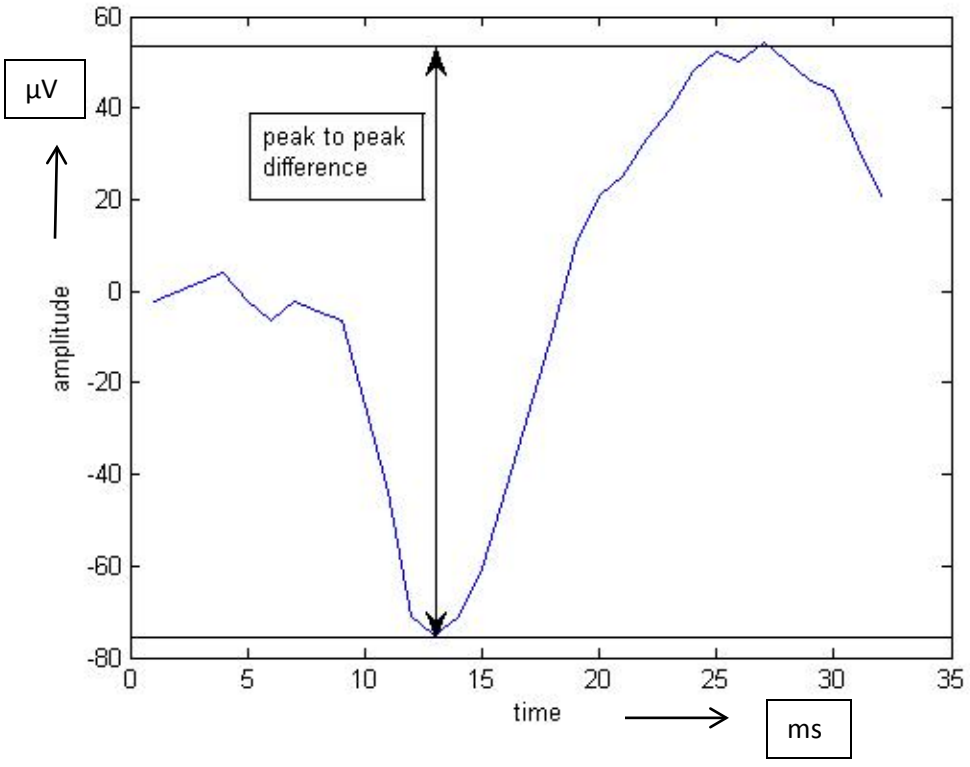

Figure 3.1: Peak to peak difference

3. Spike variance: This refers to the variation of the values of observations from their mean and is obtained by taking the summation of the square of the difference between each observation and its mean and dividing it by the total number of data points.

$$Variance(j) = \sum_{i=1}^{32} [x(i) - mean(j)]^2/32$$

where *x*(i) are data points of each observation *x*, 32 is the number of variables of each observation and j= 1,…, N is the number of observations.

4. Difference between right and left peak: Each observation is divided into two halves based on the 32 time instants at which the data is recorded. First half is from 1 to 16 and the second half is from 17 to 32 and the difference between the maximum values in each half is calculated. At this point, we have neglected any time shifting of the waveform within the window during the recording process.

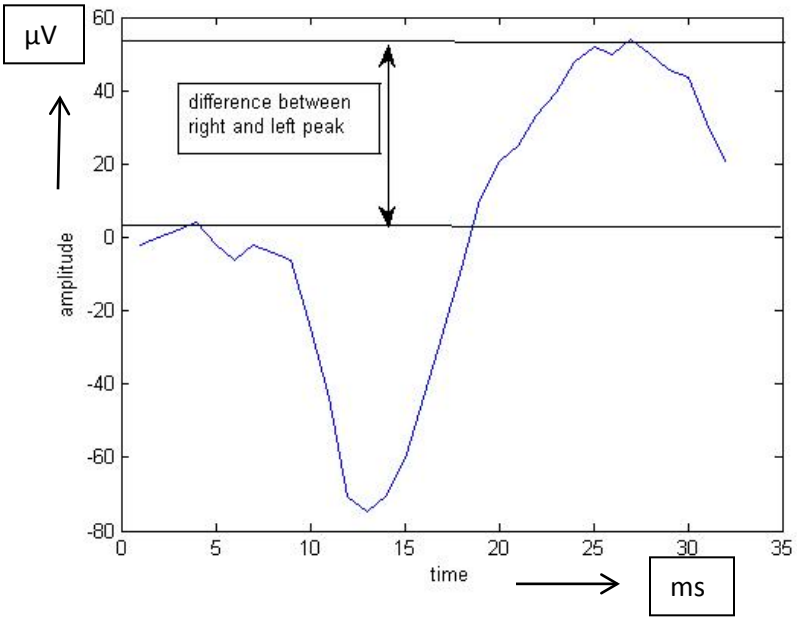

Figure 3.2: Difference between right and left peak

5. Slope of the right peak: This is the slope of each waveform as it rises from the valley to the peak.

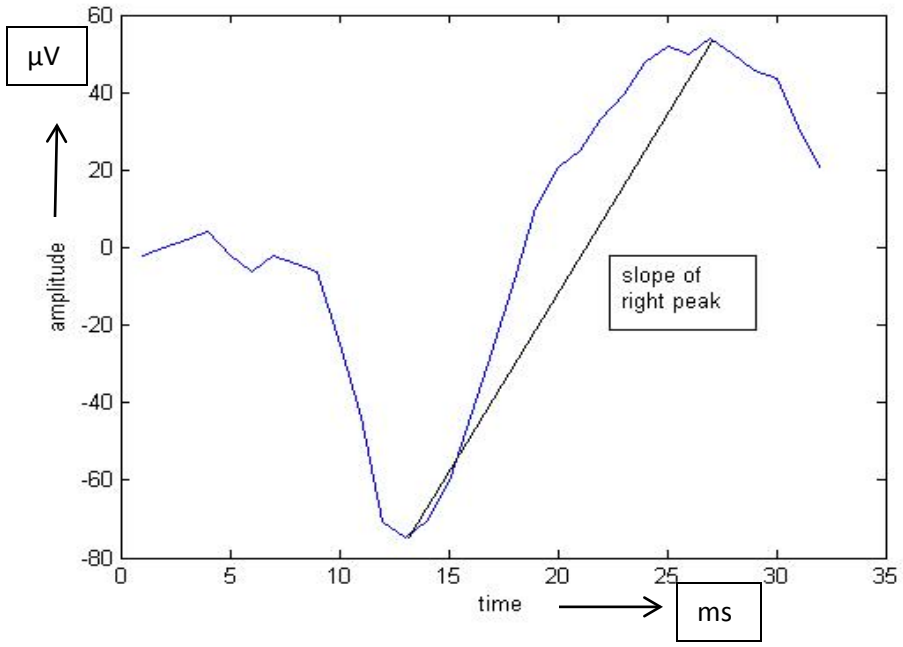

Figure 3.3: Slope of right peak

6. Polynomial fit of peak-to-peak: This feature provides the coefficients of a second order polynomial fit for the part of each spike from the peak in the left half to the peak in the right half of the spike. It considers only the second order coefficient (leading coefficient) of the polynomial for each spike as it provides an insight about how deep the valley/ minimum of the spike is.

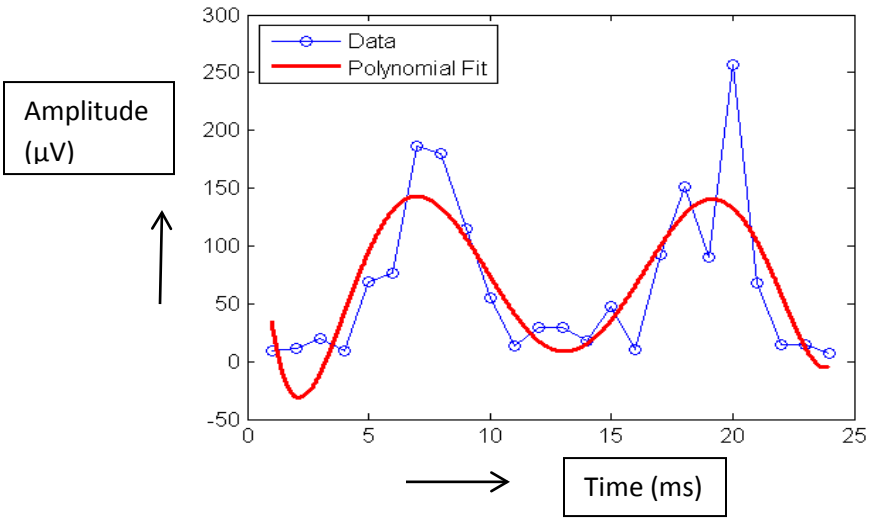

Figure 3.4: Polynomial fit

The above plot just shows what a polynomial fit does. The waveform has not been taken from the data that we worked with.

7. Polynomial fit of valley-to-peak: This feature provides the coefficients of a second order polynomial fit for the part of each spike from minima of the spike to the peak of the spike in the right half. Again, it considers only the second order coefficient (leading coefficient) of the polynomial for each spike

8. Total absolute area under both the positive and negative going peaks [23]: It is the area under the peaks of each observation and is calculated by taking the summation of the absolute value of the difference between each observation and its mean divided by the number of sampling instants.

$$Total\ absolute\ area(j) = \left\{ \sum_{i=1}^{32} |x(i) - mean(j)| \right\} / 32$$

9. Root mean square (RMS) distance [23]: It is obtained by taking the summation of the square of the difference between each observation and its mean, taking the square root of this quantity and dividing it by the total number of data points. It would have been ideal to know the amplitude of the waveform relative to the resting voltage in between spikes but for simplicity purposes, the mean voltage has been considered.

$$RMS\ distance(j) = \left\{\sum_{i=1}^{32} [x(i) - mean(j)]^2\right\}^{1/2} /32$$

These nine features form the variables for each observation of the input data. These feature vectors are put together as rows in a matrix to get the reduced representation of the original data. The new dimensions of the data are now 9xN. Each feature provides some information about the original data itself based on its geometric properties and can be used to cluster the data and distinguish between the activity of different neurons.

3.3 Weighting Techniques:

Although each feature contributes some information about the original data, the contribution is not the same. Existing methods used to distinguish between the activity of retinal ganglion cells do not take this into consideration. The more useful a feature is, the more important role it should play in the clustering process. Hence, there is a need to assess the importance of these features for every data set and put a bias on the more important features in order to improve the clustering results. Here, we propose two such intelligent measures to determine the dominance of features and weight the features accordingly before clustering the data.

3.3.1 Entropy-based feature weighting:

Entropy is defined as the measure of uncertainty associated with any random variable Each feature provides different amount of information about the original data which can be used to improve the clustering. Entropy quantifies the expected value of this information. The higher

the entropy value, the higher is the information content associated with the feature. In this approach, the information content of each feature is calculated and these entropy values are then used to weight the features.

The values of each feature are initially divided into a number of bins using a histogram. The number of bins is decided beforehand. Although entropy values increase with increasing number of bins, no improvement is observed in the clustering results. Hence we have chosen number of bins = 100 in our case. Once the data is divided into bins, their probability mass function is calculated. The entropy of any feature 'i' is then defined as

$$H(i) = -\sum_{j=1}^{B+1} p(i,j) * \log[p(i,j)]$$

where B is the number of bins and p(i, j) is the probability mass function [24]. The feature vector is then weighted with its own entropy value. This process is followed for all the nine features. Following is a summary of the weighting process:

Entropy calculation and weighting:

Given: A dataset X with 9 variables and N observations (9xN)

Step 1: Select number of bins B used in the histogram.

Step 2: Divide the data values of each row vector into these bins.

Step 3: Based on the number of data points in each bin of the histogram, calculate the probability mass function (p.m.f) of that feature row vector.

$$p(i,j) = (count\ in\ bin\ j)/N$$

where i=1, …, 9 (features) and j= 1, …, (B+1). All values equal to 1 are recorded by the $(B+1)^{th}$ column.

Step 4: Calculate the entropy of each feature (each row of X) using the entropy formula.

Step 5: Multiply each feature vector (row of X) with its corresponding entropy value.

$$X(i, all\ columns) = entropy(i) * X(i, all\ columns)$$

where i is the feature and goes from one to nine.

Step 6: Run K-means algorithm on this entropy weighted data.

3.3.2 Eigen-value based feature weighting:

This approach applies the concept of PCA in finding the relative relevance of each feature. PCA is performed on the reduced 9xN representation of the original 32xN data. The eigenvectors obtained during this process represent the projections of each feature on a lower dimensional space. The significance of each eigenvector is associated with its eigenvalue as the eigenvalue corresponds to the variance correlated with its eigenvector. The eigenvector with the highest eigenvalue is the principle component of the data set and has as high a variance as possible. These eigenvalues are used to weight the features [25].

The data is initially organized. Considering the N observations having nine variables each as N column vectors, these N column vectors are placed into a matrix X of dimensions 9xN. The mean value is calculated along each of the nine dimensions corresponding to nine features and is subtracted from each column (observation) of the matrix X. We then calculate the 9x9 dimensional covariance matrix of X. The eigenvector matrix V and eigenvalues of this covariance matrix are determined. The eigenvectors form the basis of the data. These eigenvectors are then sorted based on their eigenvalues, highest to lowest. The eigenvalues and eigenvectors are ordered and paired. The i^{th} eigenvalue corresponds to the i^{th} eigenvector. We then obtain the 9xN transformed data matrix X' by multiplying the transpose of the eigenvector matrix V with the data matrix X. X' represents the original data solely in terms of the eigenvectors. Each row 'i' of X' corresponds to the projections of the entire data on the i^{th} eigenvector. Each row 'i' is then weighted with the i^{th} eigenvalue.

Following is a summary of the weighting process:

Eigenvalue based weighting:

Given: A data set X having M=9 variables and N observations (9xN)

Step 1: Organize the data into a matrix of M rows and N columns

Step 2 Calculate the mean of the data along each dimension (feature) M and subtract it from each column of the data matrix X.

$$mean(m) = \frac{1}{N}\sum_{n=1}^{N} X(m,n)$$

where m = 1, …, 9

$$mean\ subtracted\ data\ X = X - mean * h$$

where h is a 1xN row vector of all ones. h[n] = 1, n=1, …, N to match the matrix dimensions for subtraction.

Step 3 Calculate the MxM covariance matrix C of the mean-subtracted data using outer product.

$$C = \frac{1}{N-1}[X * X^T]$$

where X is mean subtracted data and X^T is its transpose.

Step 4 Determine the MxM eigenvector matrix V and the MxM diagonal matrix D of eigenvalues of the covariance matrix C such that CV = VD.

Step 5: Sort the columns of V and D in order of decreasing eigenvalue to get the 9x1 eigenvalue matrix having the nine eigenvalues in descending order. Normalize the eigenvalues.

$$D'(i) = D(i)/D_{max}$$

where i = 1, …., 9 and D_{max} is the highest eigenvalue.

Step 6: Obtain the transformed data X'.

$$X' = V^T * X$$

Step 7: Weight each feature row vector of transformed data X' with the corresponding eigenvalue.

$$X'(i, all\ columns) = X'(i, all\ columns) * D'(i)$$

where i = 1, …., 9

Step 8: Run K-means to cluster eigenvalue weighted data.

3.4 K-means Clustering Procedure

After the features have been calculated, the dimensions of the data have been reduced and the features have been weighted, the reduced representation is then clustered using the standard K-means clustering algorithm. It provides the labels for each observation of the data describing which cluster it belongs to and the cluster centers. These labels can then be used to evaluate the clustering procedure if we already have the labels for the original data (gold standard).

3.5 Cluster assignment

There is a non-trivial problem of cluster assignment when you are given the original and observed labels for the clusters. This is because the labels in each set of clustering results do not correspond in a one-to-one manner. For instance, label 1 in original clustering need not correspond to label 1 in the observed labels; it may correspond to label 3 in the observed labels

and this cannot be always inferred visually. Hence there is a need to determine which observed cluster labels correspond to which original cluster labels. To solve this problem, we use the standard Hungarian Algorithm [26]. It is a combinatorial optimization algorithm which solves the assignment problem in polynomial time.

3.6 Evaluation measures

Here I will briefly introduce some of the measures used to evaluate the clustering performance. The manually sorted results are used as gold standard. For all the measures, we first calculate the following quantities:

True positives (tp): the number of spikes correctly labeled as belonging to the correct cluster

True negatives (tn): spikes which were correctly labeled as not belonging to the cluster

False positives (fp): spikes incorrectly labeled as belonging to the cluster

False negatives (fn): spikes which were not labeled as belonging to the correct cluster but should have been.

Following are the evaluation measures:

Precision [27]: It is defined as the number of relevant spikes retrieved divided by the total number of spikes retrieved.

$$precision = \frac{tp}{tp + fp}$$

Recall [27]: It is the number of relevant spikes retrieved divided by the total number of spikes that should have been ideally retrieved.

$$recall = \frac{tp}{tp + fn}$$

These two measures have a range of 0 to 1 and reach their best value at 1.

Missed classifications: These are the number of spikes missed in the observed cluster divided by the number of spikes in the original cluster.

$$missed(i) = \frac{fn}{actual\ number\ of\ spikes\ in\ cluster\ i}$$

False classification: This refers to the number of spikes falsely classified in the observed cluster divided by the number of spikes in the original cluster.

$$false\ classified(i) = \frac{fp}{actual\ number\ of\ spikes\ in\ cluster\ i}$$

CHAPTER 4

RESULTS

All the relevant results of this project are described in this chapter. To determine the efficacy of the proposed algorithm, we performed experiments on nineteen real data sets with known cluster labels obtained from manual sorting (gold standard).

Following is a description of these data sets and the results obtained.

4.1 Real data sets

Nineteen real data sets were obtained from rd mice. The results of the existing methods (K-means, PCA) for these nineteen, thirty two dimensional data sets were obtained. These results were used as the baseline (32xN K-means). Some statistics are shown in Table 1.

Table 4.1: Real data sets and some statistics

Data set	**Total Number of Observations / Spikes**	**Number of clusters**
sorted052307channel_54	13163	2
sorted050207channel_31	16764	3
sorted050207channel_34	5163	2
sorted050207channel_12	6512	3
sorted052307channel_23	34681	2
sorted050207channel_13	4486	2
sorted050207channel_16	10605	2
sorted050207channel_46	36469	4
sorted050207channel_57	14333	3
sorted050207channel_67	2474	2
sorted050207channel_78	9184	3
sorted052307channel_16	2937	7
sorted052307channel_32	9751	3
sorted052307channel_37	3812	6
sorted052307channel_46	9451	7
sorted052307channel_47	5472	3
sorted052307channel_63	10524	5
sorted052307channel_64	8882	5
sorted052307channel_71	1939	2

4.2 Manual weighting

As mentioned before in section 3.4, all the nine features do not contribute equally towards the clustering process. The more important a feature is the more important role it should play in the clustering process. Unfortunately, we do not know what the best criterion to decide the importance of all the features is or how to select the weights for all the features. To overcome this problem, we first selected weights manually for all the features and obtained the results of the clustering process. Following is a description of the results for cluster number two of the data set: sorted050207channel_31.

Data set: sorted050207channel_31

Number of clusters: 3

Cluster under consideration: cluster # 2

Number of spikes in data set: 16764

Number of spikes in cluster # 2: 15637

Weights: Manual weights set 1

Table 4.2: Results of manual weighting (set 1)

Evaluation measure	**Cluster**	**32xN K-means**	**9xN K-means with Manual weights set 1**
Precision	C2	0.9975	0.9994
Recall	C2	0.6302	0.689
Missed	C2	0.3698	0.311
False classified	C2	0.0016	0.0004

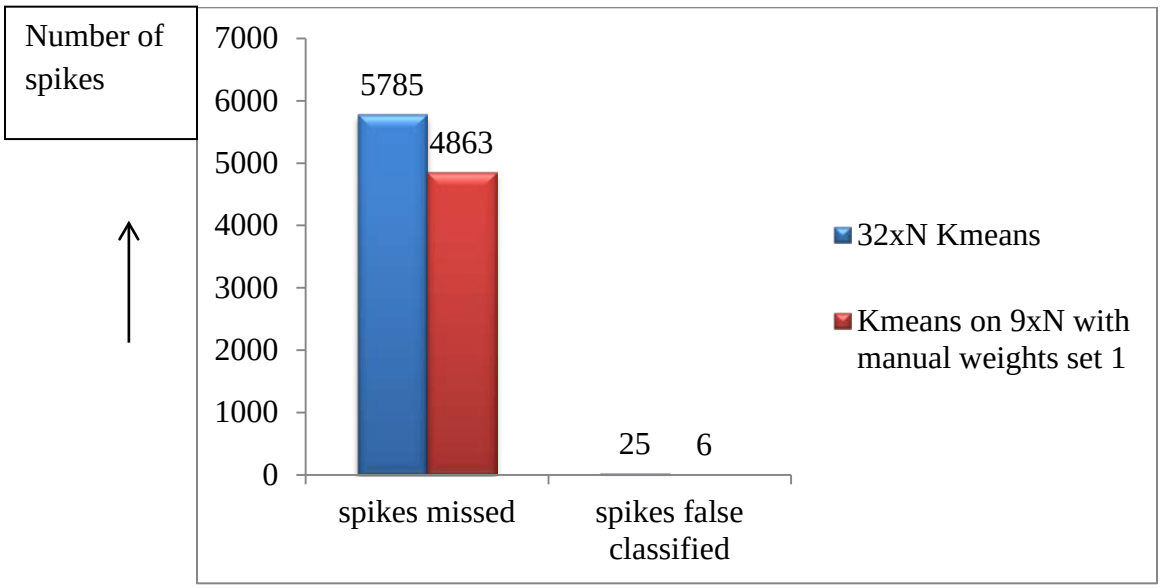

Figure 4.1: Results of manual weighting (set 1)

Based on the above results, the existing method falsely classified 25 spikes as belonging to cluster 2 and missed 5785 out of 15637 spikes which should have been classified as cluster 2. The manually weighted data showed a marked improvement over the baseline results with only 6 spikes false classified and missed 4863 out of 15637 spikes which is almost 1000 spikes less than the baseline result. Following is a description of results for the same cluster of data set: sorted050207channel_31 with manual weights set 2.

Data set: sorted050207channel_31

Number of clusters: 3

Cluster under consideration: cluster # 2

Number of spikes in data set: 16764

Number of spikes in cluster # 2: 15637

Weights: Manual weights set 2

Table 4.3: Results of manual weighting (set 2)

Evaluation measure	**Cluster**	**32xN K-means**	**9xN K-means with Manual weights set 2**
Precision	C2	0.9975	0.9886
Recall	C2	0.6302	0.4816
Missed	C2	0.3698	0.5184
False classified	C2	0.0016	0.0056

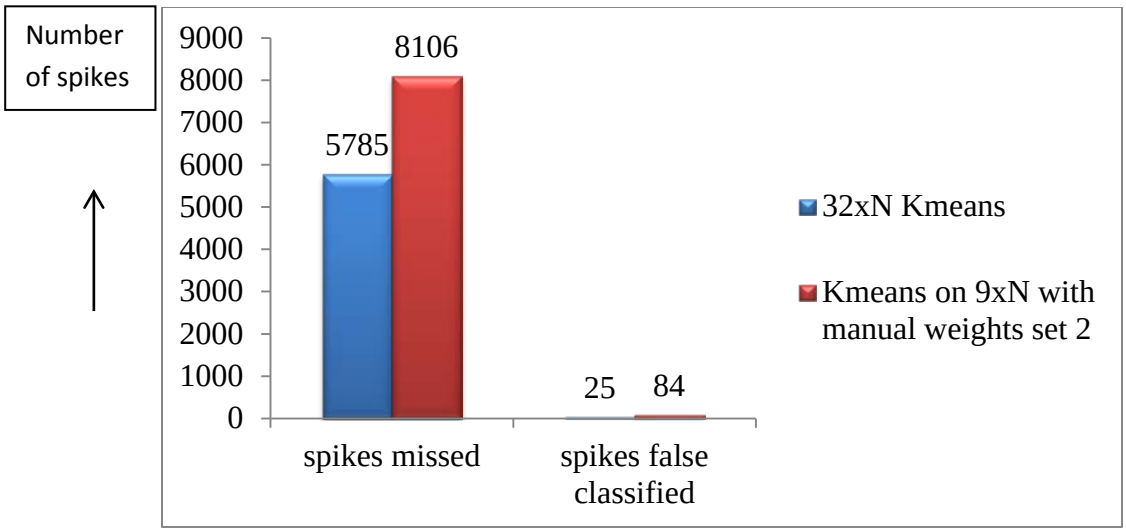

Figure 4.2: Results of manual weighting (set 2)

Based on the above results, the manually weighted data set (set 2) performed very poorly compared to the baseline results. It falsely classified 84 spikes compared to 25 of the baseline and missed 8106 out of 15637 spikes compared to 5785 for the baseline.

The above results show that we need to select the weights very carefully. If we select set 1, we get much improved results over the baseline. If we select set 2, we get very poor results. It also establishes the fact that there exist sets of weights which can provide an improved performance over the baseline results.

Following sections provide a description of the results for the two proposed weighting methods based on entropy and eigenvalues. For each of the following tables, column 1 refers to the data set under consideration, column 2 refers to the evaluation measure, column 3 corresponds to the particular cluster of the data set under consideration. Columns 4 and 5 refer to the two clustering approaches that are being compared. The evaluation measures of precision and recall reach their best value at 1 and worst at 0. So any number close to 1 is desirable. On the

other hand, the measures of missed and false classification should have a value of 0 ideally. So the lower the value, the better the result is for these two measures. In some cases, the missed or false classification value may be more than 1 because the original cluster sizes are very small and dividing any number greater than that gives a value which is more than 1.

4.3 Comparison between results of K-means on 32xN data and Entropy weighted 9xN data

This section provides a comparison between the results of K-means on the original 32xN data set and the results of K-means on feature-based, entropy weighted 9xN data set.

Table 4.4: Results of K-means on 32xN data and Entropy weighted 9xN data

Data set	**Metric**	**Cluster**	**32xN Kmeans**	**9xN Entropy weighted**
sorted050207channel_12	precision	C1	0.9821	0.9539
		C2	0.8964	0.771
		C3	0.9163	0.9177
	recall	C1	0.9948	0.9969
		C2	0.8715	0.8584
		C3	0.9268	0.8075
	missed	C1	0.0052	0.0031
		C2	0.1285	0.1416
		C3	0.0732	0.1925
	false	C1	0.0181	0.0482
		C2	0.1008	0.2549
		C3	0.0847	0.0724
sorted050207channel_13	precision	C1	0.9695	0.8232
		C2	0.9307	0.8975
	recall	C1	0.9014	0.8668
		C2	0.9791	0.8623
	missed	C1	0.0986	0.1332
		C2	0.0209	0.1377
	false	C1	0.0283	0.1862
		C2	0.0729	0.0985
sorted050207channel_16	Precision	C1	0.1274	0.1551

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
		C2	0.9961	0.9995
	Recall	C1	0.966	0.9954
		C2	0.5694	0.6472
	Missed	C1	0.034	0.0046
		C2	0.4306	0.3528
	False	C1	6.6157	5.4213
		C2	0.0022	0.0003
sorted050207channel_31	Precision	C1	0.9564	0.8268
		C2	0.9975	0.9978
		C3	0	0.0005
	Recall	C1	0.9937	0.9946
		C2	0.6302	0.7382
		C3	0	0.087
	Missed	C1	0.0063	0.0054
		C2	0.3698	0.2618
		C3	0	0.913
	False	C1	0.0453	0.2083
		C2	0.0016	0.0016
		C3	0	168.0435
sorted050207channel_34	Precision	C1	0.0513	0.0606
		C2	1	1
	Recall	C1	1	1

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
		C2	0.522	0.5998
	Missed	C1	0	0
		C2	0.478	0.4002
	False	C1	18.5077	15.4923
		C2	0	0
sorted050207channel_46	Precision	C1	0.0006	0
		C2	0.6775	0.6339
		C3	0.0028	0.0012
		C4	0.9988	0.9997
	Recall	C1	0.0166	0
		C2	0.9967	1
		C3	0.7442	0.3488
		C4	0.3444	0.4237
	Missed	C1	0.9834	0
		C2	0.0033	0
		C3	0.2558	0.6512
		C4	0.6556	0.5763
	False	C1	27.4679	0
		C2	0.4745	0.5776
		C3	265.4419	291.1628
		C4	0.0004	0.0001
sorted050207channel_57	Precision	C1	0.9886	0.957

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
		C2	0.962	0.9934
		C3	0.0033	0.0013
	Recall	C1	0.8995	0.5221
		C2	0.324	0.3731
		C3	1	0.5
	Missed	C1	0.1005	0.4779
		C2	0.676	0.6269
		C3	0	0.5
	False	C1	0.0103	0.0235
		C2	0.0128	0.0025
		C3	304	390
sorted050207channel_67	Precision	C1	0.9797	0.8701
		C2	0.9784	0.9024
	Recall	C1	0.9725	0.8773
		C2	0.9841	0.8965
	Missed	C1	0.0275	0.1227
		C2	0.0159	0.1035
	False	C1	0.0201	0.131
		C2	0.0217	0.097
sorted050207channel_78	Precision	C1	0.3414	0.117
		C2	0.9527	0.6949
		C3	0.9749	0.9881

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
	Recall	C1	0.918	0.2882
		C2	0.9117	0.8961
		C3	0.6832	0.6038
	Missed	C1	0.082	0.7118
		C2	0.0883	0.1039
		C3	0.3168	0.3962
	False	C1	1.7707	2.175
		C2	0.0452	0.3935
		C3	0.0176	0.0072
sorted052307channel_16	Precision	C1	0.9888	0.9697
		C2	0.9782	0.3403
		C3	0.9962	0.9824
		C4	0.9812	0.7897
		C5	0.0046	0
		C6	0	0
		C7	0.003	0.0056
	Recall	C1	0.5473	0.9877
		C2	0.534	0.3333
		C3	0.7689	0.3835
		C4	0.7009	0.6007
		C5	0.5	0
		C6	0	0
		C7	0.1667	0.3333

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
	Missed	C1	0.4527	0.0123
		C2	0.466	0.6667
		C3	0.2311	0.6165
		C4	0.2991	0.3993
		C5	0.5	0
		C6	0	0
		C7	0.8333	0.6667
	False	C1	0.0062	0.0309
		C2	0.0119	0.6463
		C3	0.0029	0.0069
		C4	0.0134	0.16
		C5	108.5	0
		C6	0	0
		C7	56.1667	58.8333
sorted052307channel_23	Precision	C1	0.9996	0.998
		C2	0.8632	0.892
	Recall	C1	0.9765	0.9822
		C2	0.9972	0.987
	Missed	C1	0.0235	0.0178
		C2	0.0028	0.013
	False	C1	0.0004	0.0019
		C2	0.1581	0.1195
sorted052307channel_32	Precision	C1	0.001	0.8932

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
		C2	1	0.9877
		C3	0.0021	0
	Recall	C1	0.0022	0.9734
		C2	0.4661	0.5148
		C3	1	0
	Missed	C1	0.9978	0.0266
		C2	0.5339	0.4852
		C3	0	0
	False	C1	2.1983	0.1164
		C2	0	0.0064
		C3	473.6667	0
sorted052307channel_37	Precision	C1	0.9989	1
		C2	0.0122	0.1141
		C3	0.9242	0.8522
		C4	0.8735	0.7034
		C5	0.0066	0.0101
		C6	0	0
	Recall	C1	0.9103	0.497
		C2	0.2	0.95
		C3	0.3245	0.4798
		C4	0.5413	0.5953
		C5	0.4286	0.7143
		C6	0	0

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
	Missed	C1	0.0897	0.503
		C2	0.8	0.05
		C3	0.6755	0.5202
		C4	0.4587	0.4047
		C5	0.5714	0.2857
		C6	0	0
	False	C1	0.001	0
		C2	16.225	7.375
		C3	0.0266	0.0832
		C4	0.0784	0.2511
		C5	65	70.2857
		C6	0	0
sorted052307channel_47	Precision	C1	0	0.0043
		C2	0.632	0.0988
		C3	1	0.9995
	Recall	C1	0	0.32
		C2	1	0.8075
		C3	0.5315	0.3569
	Missed	C1	0	0.68
		C2	0	0.1925
		C3	0.4685	0.6431
	False	C1	0	74.16
		C2	0.5822	7.3662

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
		C3	0	0.0002
			32xN	Entropy
sorted052307channel_54	Precision	C1	0.9882	0.9897
		C2	0.9618	0.9882
	Recall	C1	0.9731	0.9919
		C2	0.9831	0.985
	Missed	C1	0.0269	0.0081
		C2	0.0169	0.015
	False	C1	0.0117	0.0103
		C2	0.0391	0.0117
sorted052307channel_63	Precision	C1	0.9995	0.9833
		C2	0.9124	0.9762
		C3	0	0.0204
		C4	0.0317	0.0336
		C5	0	0
	Recall	C1	0.2811	0.4455
		C2	0.9967	0.6051
		C3	0	0.3871
		C4	0.2435	0.3391
		C5	0	0
	Missed	C1	0.7189	0.5545
		C2	0.0033	0.3949
		C3	0	0.6129

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
		C4	0.7565	0.6609
		C5	0	0
	False	C1	0.0001	0.0076
		C2	0.0957	0.0148
		C3	0	18.6129
		C4	7.4261	9.7522
		C5	0	0
sorted052307channel_64	Precision	C1	0.9754	0.9723
		C2	0.0213	0.906
		C3	0.9193	0.9822
		C4	0.0091	0.0177
		C5	0	0.0061
	Recall	C1	0.3611	0.5382
		C2	0.0075	0.5543
		C3	0.9804	0.7969
		C4	0.9615	1
		C5	0	0.4762
	Missed	C1	0.6389	0.4618
		C2	0.9925	0.4457
		C3	0.0196	0.2031
		C4	0.0385	0
		C5	0	0.5238
	False	C1	0.0091	0.0154

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
		C2	0.3454	0.0575
		C3	0.086	0.0144
		C4	104.6923	55.5
		C5	0	77.9524
sorted052307channel_71	Precision	C1	0.0529	0.0593
		C2	1	1
	Recall	C1	1	1
		C2	0.5744	0.623
	Missed	C1	0	0
		C2	0.4256	0.377
	False	C1	17.9111	15.8667
		C2	0	0
sorted052307channel_46	Precision	C1	0.703	0.7414
		C2	0.0014	0
		C3	0.0163	0.0498
		C4	0.1639	0.0007
		C5	0.9973	0.989
		C6	0.0721	0.8435
		C7	0.0009	0
	Recall	C1	0.8838	0.9876
		C2	0.0187	0
		C3	0.4444	1

Data set	Metric	Cluster	32xN Kmeans	9xN Entropy weighted
		C4	0.9602	0.008
		C5	0.2845	0.3051
		C6	0.123	0.5061
		C7	0.5	0
	Missed	C1	0.1162	0.0124
		C2	0.9813	0
		C3	0.5556	0
		C4	0.0398	0.992
		C5	0.7155	0.6949
		C6	0.877	0.4939
		C7	0.5	0
	False	C1	0.3734	0.3444
		C2	13.514	0
		C3	26.75	19.0833
		C4	4.8964	11.2908
		C5	0.0008	0.0034
		C6	1.584	0.0939
		C7	584	0

It is difficult to analyze the performance for an entire data set as a whole since a particular algorithm performs better for some clusters and poorly for others. Instead, if each

cluster is considered as a separate entity, then it is easier to quantify the performance. We had 264 such clusters in total. Based on this analysis of the experimental results, if we take a vote by comparing the values as means for evaluation then the results of K-means on the entropy weighted 9xN data are better than the baseline results of K-means on the original 32xN data in more than 54% of the cases. The baseline results have a better value in the remaining 46% cases. The following plot depicts this comparison.

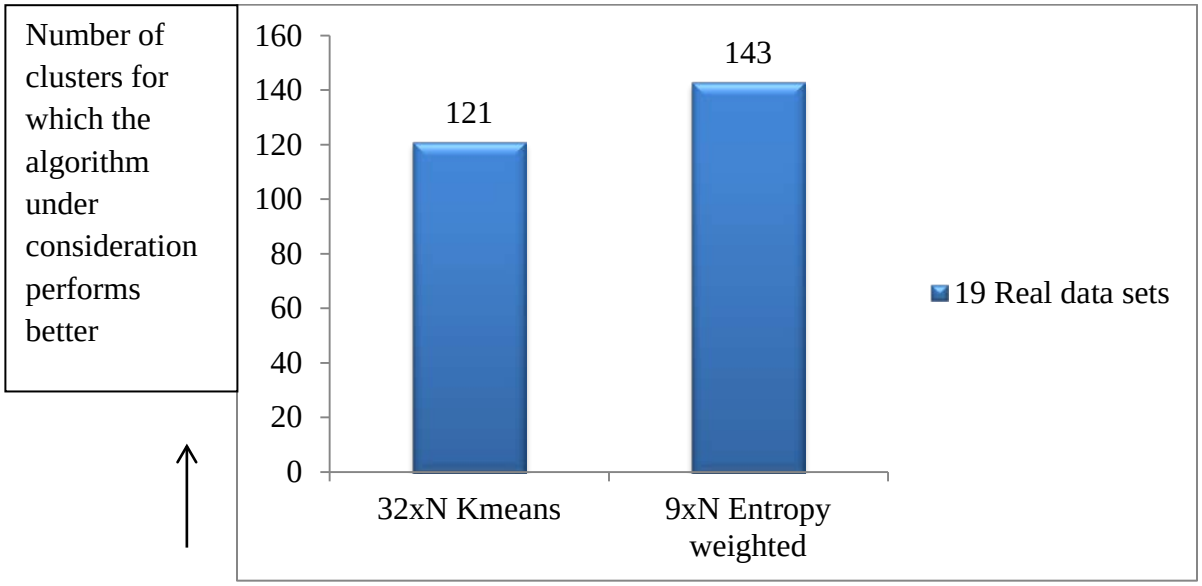

Figure 4.3: Comparison between results of 32xN K-means and Entropy weighted 9xN data

4.4 Comparison between results of K-means on 32xN data and Eigenvalue weighted 9xN data

This section provides a comparison between the results of K-means on the original 32xN data sets and the results of K-means on feature-based, eigenvalue weighted 9xN data sets.

Table 4.5: Results of K-means on 32xN data and Eigenvalue weighted 9xN data

Data set	**Metric**	**Cluster**	**32xN Kmeans**	**K-means on PCA transformed, Eigenvalue weighted 9xN**
sorted050207channel_12	precision	C1	0.9821	0.9501
		C2	0.8964	0.7542
		C3	0.9163	0.9212
	recall	C1	0.9948	0.9974
		C2	0.8715	0.861
		C3	0.9268	0.7874
	missed	C1	0.0052	0.0026
		C2	0.1285	0.139
		C3	0.0732	0.2126
	false	C1	0.0181	0.0523
		C2	0.1008	0.2806
		C3	0.0847	0.0674
sorted050207channel_13	precision	C1	0.9695	0.7842
		C2	0.9307	0.9013
	recall	C1	0.9014	0.8783
		C2	0.9791	0.8212
	missed	C1	0.0986	0.1217
		C2	0.0209	0.1788
	false	C1	0.0283	0.2417

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
		C2	0.0729	0.09
sorted050207channel_16	Precision	C1	0.1274	0.1358
		C2	0.9961	0.9998
	Recall	C1	0.966	0.9985
		C2	0.5694	0.5864
	Missed	C1	0.034	0.0015
		C2	0.4306	0.4136
	False	C1	6.6157	6.3549
		C2	0.0022	0.0001
sorted050207channel_31	Precision	C1	0.9564	0.9144
		C2	0.9975	0.9987
		C3	0	0.0014
	Recall	C1	0.9937	0.9864
		C2	0.6302	0.6221
		C3	0	0.3478
	Missed	C1	0.0063	0.0136
		C2	0.3698	0.3779
		C3	0	0.6522
	False	C1	0.0453	0.0924

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
		C2	0.0016	0.0008
		C3	0	253.2174
sorted050207channel_34	Precision	C1	0.0513	0.0556
		C2	1	1
	Recall	C1	1	1
		C2	0.522	0.5609
	Missed	C1	0	0
		C2	0.478	0.4391
	False	C1	18.5077	17
		C2	0	0
sorted050207channel_46	Precision	C1	0.0006	0
		C2	0.6775	0.6518
		C3	0.0028	0.0025
		C4	0.9988	1
	Recall	C1	0.0166	0
		C2	0.9967	1
		C3	0.7442	0.4419
		C4	0.3444	0.4262
	Missed	C1	0.9834	0

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
		C2	0.0033	0
		C3	0.2558	0.5581
		C4	0.6556	0.5738
	False	C1	27.4679	0
		C2	0.4745	0.5342
		C3	265.4419	179.6512
		C4	0.0004	0
sorted050207channel_57	Precision	C1	0.9886	0.9894
		C2	0.962	0.9824
		C3	0.0033	0.0021
	Recall	C1	0.8995	0.7925
		C2	0.324	0.3463
		C3	1	0.5
	Missed	C1	0.1005	0.2075
		C2	0.676	0.6537
		C3	0	0.5
	False	C1	0.0103	0.0085
		C2	0.0128	0.0062
		C3	304	234

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
sorted050207channel_67	Precision	C1	0.9797	0.8687
		C2	0.9784	0.9208
	Recall	C1	0.9725	0.9029
		C2	0.9841	0.8922
	Missed	C1	0.0275	0.0971
		C2	0.0159	0.1078
	False	C1	0.0201	0.1364
		C2	0.0217	0.0767
sorted050207channel_78	Precision	C1	0.3414	0.1208
		C2	0.9527	0.7027
		C3	0.9749	0.9841
	Recall	C1	0.918	0.2937
		C2	0.9117	0.8817
		C3	0.6832	0.6209
	Missed	C1	0.082	0.7063
		C2	0.0883	0.1183
		C3	0.3168	0.3791
	False	C1	1.7707	2.1372
		C2	0.0452	0.373
		C3	0.0176	0.01

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
sorted052307channel_16	Precision	C1	0.9888	0.9731
		C2	0.9782	0.3867
		C3	0.9962	0.9638
		C4	0.9812	0.6324
		C5	0.0046	0
		C6	0	0
		C7	0.003	0.0075
	Recall	C1	0.5473	0.9671
		C2	0.534	0.2466
		C3	0.7689	0.3923
		C4	0.7009	0.4054
		C5	0.5	0
		C6	0	0
		C7	0.1667	0.3333
	Missed	C1	0.4527	0.0329
		C2	0.466	0.7534
		C3	0.2311	0.6077
		C4	0.2991	0.5946
		C5	0.5	0
		C6	0	0
		C7	0.8333	0.6667

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
	False	C1	0.0062	0.0267
		C2	0.0119	0.3912
		C3	0.0029	0.0147
		C4	0.0134	0.2357
		C5	108.5	0
		C6	0	0
		C7	56.1667	44.1667
sorted052307channel_23	Precision	C1	0.9996	0.9982
		C2	0.8632	0.8905
	Recall	C1	0.9765	0.9819
		C2	0.9972	0.9884
	Missed	C1	0.0235	0.0181
		C2	0.0028	0.0116
	False	C1	0.0004	0.0017
		C2	0.1581	0.1216
sorted052307channel_32	Precision	C1	0.001	0.8793
		C2	1	1
		C3	0.0021	0.0007
	Recall	C1	0.0022	0.9634
		C2	0.4661	0.487

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
		C3	1	0.6667
	Missed	C1	0.9978	0.0366
		C2	0.5339	0.513
		C3	0	0.3333
	False	C1	2.1983	0.1322
		C2	0	0
		C3	473.6667	946.3333
sorted052307channel_37	Precision	C1	0.9989	0.998
		C2	0.0122	0.1814
		C3	0.9242	0.8986
		C4	0.8735	0.6895
		C5	0.0066	0
		C6	0	0.0013
	Recall	C1	0.9103	0.9744
		C2	0.2	0.975
		C3	0.3245	0.3783
		C4	0.5413	0.4799
		C5	0.4286	0
		C6	0	1
	Missed	C1	0.0897	0.0256

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
		C2	0.8	0.025
		C3	0.6755	0.6217
		C4	0.4587	0.5201
		C5	0.5714	0
		C6	0	0
	False	C1	0.001	0.002
		C2	16.225	4.4
		C3	0.0266	0.0427
		C4	0.0784	0.2161
		C5	65	0
		C6	0	788
sorted052307channel_47	Precision	C1	0	0.0893
		C2	0.632	0.0004
		C3	1	0.9975
	Recall	C1	0	1
		C2	1	0.0047
		C3	0.5315	0.5367
	Missed	C1	0	0
		C2	0	0.9953
		C3	0.4685	0.4633

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
	False	C1	0	10.2
		C2	0.5822	11.1502
		C3	0	0.0013
			32xN	
sorted052307channel_54	Precision	C1	0.9882	0.9906
		C2	0.9618	0.988
	Recall	C1	0.9731	0.9917
		C2	0.9831	0.9863
	Missed	C1	0.0269	0.0083
		C2	0.0169	0.0137
	False	C1	0.0117	0.0094
		C2	0.0391	0.012
sorted052307channel_63	Precision	C1	0.9995	0.9841
		C2	0.9124	0.9549
		C3	0	0.0004
		C4	0.0317	0.0438
		C5	0	0
	Recall	C1	0.2811	0.2955
		C2	0.9967	0.9922
		C3	0	0.0161

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
		C4	0.2435	0.313
		C5	0	0
	Missed	C1	0.7189	0.7045
		C2	0.0033	0.0078
		C3	0	0.9839
		C4	0.7565	0.687
		C5	0	0
	False	C1	0.0001	0.0048
		C2	0.0957	0.0469
		C3	0	39.2581
		C4	7.4261	6.8391
		C5	0	0
sorted052307channel_64	Precision	C1	0.9754	0.9685
		C2	0.0213	0.9351
		C3	0.9193	0.9842
		C4	0.0091	0.0017
		C5	0	0.0043
	Recall	C1	0.3611	0.5201
		C2	0.0075	0.621
		C3	0.9804	0.6684

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
		C4	0.9615	0.1154
		C5	0	0.3333
	Missed	C1	0.6389	0.4799
		C2	0.9925	0.379
		C3	0.0196	0.3316
		C4	0.0385	0.8846
		C5	0	0.6667
	False	C1	0.0091	0.0169
		C2	0.3454	0.0431
		C3	0.086	0.0107
		C4	104.6923	67.6923
		C5	0	76.381
sorted052307channel_71	Precision	C1	0.0529	0.0542
		C2	1	1
	Recall	C1	1	1
		C2	0.5744	0.585
	Missed	C1	0	0
		C2	0.4256	0.415
	False	C1	17.9111	17.4667
		C2	0	0

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
sorted052307channel_46	Precision	C1	0.703	0.7881
		C2	0.0014	0.1055
		C3	0.0163	0
		C4	0.1639	0.0246
		C5	0.9973	0.9492
		C6	0.0721	0.855
		C7	0.0009	0
	Recall	C1	0.8838	0.9876
		C2	0.0187	0.4112
		C3	0.4444	0
		C4	0.9602	0.1036
		C5	0.2845	0.326
		C6	0.123	0.4207
		C7	0.5	0
	Missed	C1	0.1162	0.0124
		C2	0.9813	0.5888
		C3	0.5556	0
		C4	0.0398	0.8964
		C5	0.7155	0.674
		C6	0.877	0.5793
		C7	0.5	0

Data set	Metric	Cluster	32xN Kmeans	K-means on PCA transformed, Eigenvalue weighted 9xN
	False	C1	0.3734	0.2656
		C2	13.514	3.486
		C3	26.75	0
		C4	4.8964	4.1155
		C5	0.0008	0.0175
		C6	1.584	0.0714
		C7	584	0

Again, if we take a vote as means for evaluation considering each cluster as a separate entity, then the results of K-means on eigenvalue weighted 9xN data are better than the baseline results of K-means on the original 32xN data in more than 56% of the cases. The following plot depicts this comparison.

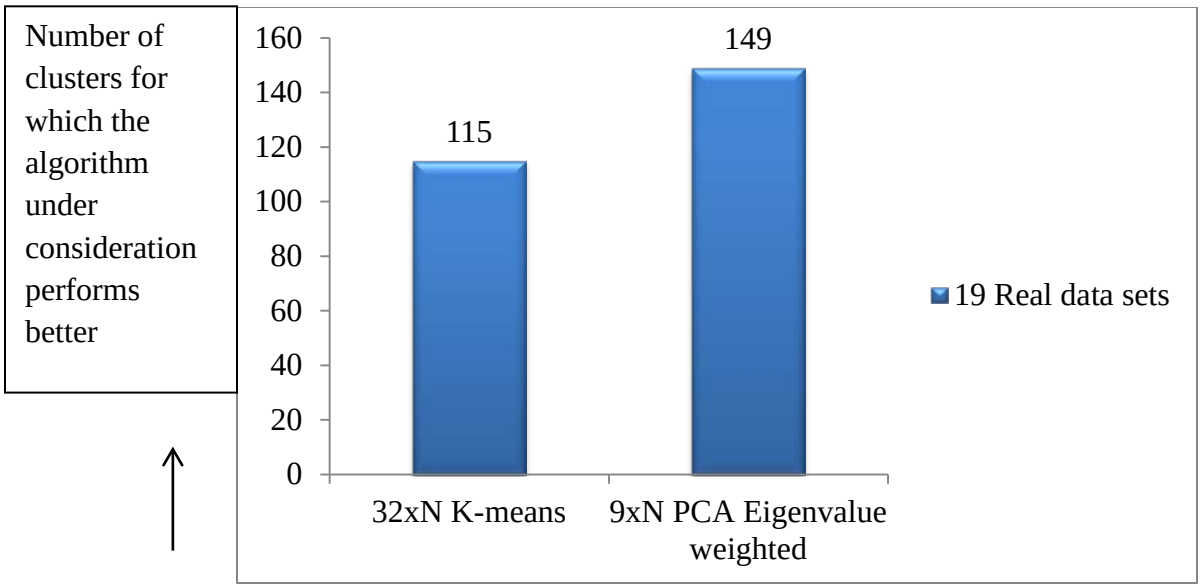

Figure 4.4: Comparison between 32xN K-means and 9xN PCA Eigenvalue weighted data

4.5 Comparison between results of K-means on 32xN PCA data and 9xN PCA Eigenvalue weighted data

This section provides a comparison between the results of PCA on the original 32xN data set and the results of K-means on the feature-based, eigenvalue weighted 9xN data set.

Table 4.6: Results of K-means on 32xN PCA data and 9xN PCA Eigenvalue weighted data

Data set	**Metric**	**Cluster**	**K-means with PCA transformed 32xN**	**K-means with PCA transformed, Eigenvalue weighted 9xN**
sorted050207channel_12	precision	C1	0.9963	0.9501
		C2	0.9625	0.7542
		C3	0.8775	0.9212
	recall	C1	0.9803	0.9974
		C2	0.8156	0.861
	missed	C1	0.0197	0.0026
		C2	0.1844	0.139
		C3	0.0096	0.2126
	false	C1	0.0036	0.0523
		C2	0.0317	0.2806
		C3	0.1382	0.0674
sorted050207channel_13	precision	C1	0.9301	0.7842
		C2	0.9622	0.9013
	recall	C1	0.9497	0.8783
		C2	0.9473	0.8212
	missed	C1	0.0503	0.1217
		C2	0.0527	0.1788
	false	C1	0.0713	0.2417
		C2	0.0372	0.09

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
sorted050207channel_16	Precision	C1	0.0683	0.1358
		C2	0.9457	0.9998
	Recall	C1	0.5463	0.9985
		C2	0.5147	0.5864
	Missed	C1	0.4537	0.0015
		C2	0.4853	0.4136
	False	C1	7.4568	6.3549
		C2	0.0295	0.0001
sorted050207channel_31	Precision	C1	0.1018	0.9144
		C2	0.9783	0.9987
		C3	0.0043	0.0014
	Recall	C1	0.452	0.9864
		C2	0.4498	0.6221
		C3	0.8696	0.3478
	Missed	C1	0.548	0.0136
		C2	0.5502	0.3779
		C3	0.1304	0.6522
	False	C1	3.9891	0.0924
		C2	0.01	0.0008
		C3	202.2174	253.2174

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
sorted050207channel_34	Precision	C1	0.0546	0.0556
		C2	1	1
	Recall	C1	1	1
		C2	0.5531	0.5609
	Missed	C1	0	0
		C2	0.4469	0.4391
	False	C1	17.3	17
		C2	0	0
sorted050207channel_46	Precision	C1	0.0019	0
		C2	0.6599	0.6518
		C3	0.0013	0.0025
		C4	0.9992	1
	Recall	C1	0.0499	0
		C2	0.9924	1
		C3	0.3721	0.4419
		C4	0.3483	0.4262
	Missed	C1	0.9501	0
		C2	0.0076	0
		C3	0.6279	0.5581
		C4	0.6517	0.5738

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
	False	C1	25.8812	0
		C2	0.5114	0.5342
		C3	277.2326	179.6512
		C4	0.0003	0
sorted050207channel_57	Precision	C1	0.9689	0.9894
		C2	0.9568	0.9824
		C3	0.0033	0.0021
	Recall	C1	0.907	0.7925
		C2	0.3112	0.3463
		C3	1	0.5
	Missed	C1	0.093	0.2075
		C2	0.6888	0.6537
		C3	0	0.5
	False	C1	0.0291	0.0085
		C2	0.014	0.0062
		C3	306.5	234
sorted050207channel_67	Precision	C1	0.9858	0.8687
		C2	0.9634	0.9208
	Recall	C1	0.9524	0.9029

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
		C2	0.9891	0.8922
	Missed	C1	0.0476	0.0971
		C2	0.0109	0.1078
	False	C1	0.0137	0.1364
		C2	0.0376	0.0767
sorted050207channel_78	Precision	C1	0.3476	0.1208
		C2	0.9599	0.7027
		C3	0.9671	0.9841
	Recall	C1	0.9042	0.2937
		C2	0.863	0.8817
		C3	0.7158	0.6209
	Missed	C1	0.0958	0.7063
		C2	0.137	0.1183
		C3	0.2842	0.3791
	False	C1	1.6971	2.1372
		C2	0.0361	0.373
		C3	0.0243	0.01
sorted052307channel_16	Precision	C1	0.9839	0.9731
		C2	0.967	0.3867

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
		C3	0.9919	0.9638
		C4	0.9718	0.6324
		C5	0.0041	0
		C6	0	0
		C7	0.0025	0.0075
	Recall	C1	0.5021	0.9671
		C2	0.7976	0.2466
		C3	0.6028	0.3923
		C4	0.5043	0.4054
		C5	0.5	0
		C6	0	0
		C7	0.1667	0.3333
	Missed	C1	0.4979	0.0329
		C2	0.2024	0.7534
		C3	0.3972	0.6077
		C4	0.4957	0.5946
		C5	0.5	0
		C6	0	0
		C7	0.8333	0.6667
	False	C1	0.0082	0.0267
		C2	0.0272	0.3912

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
		C3	0.0049	0.0147
		C4	0.0147	0.2357
		C5	121	0
		C6	0	0
		C7	67.6667	44.1667
sorted052307channel_23	Precision	C1	0.9971	0.9982
		C2	0.2446	0.8905
	Recall	C1	0.5459	0.9819
		C2	0.9893	0.9884
	Missed	C1	0.4541	0.0181
		C2	0.0107	0.0116
	False	C1	0.0016	0.0017
		C2	3.0552	0.1216
sorted052307channel_32	Precision	C1	0.9796	0.8793
		C2	0.9993	1
		C3	0	0.0007
	Recall	C1	0.9986	0.9634
		C2	0.5134	0.487
		C3	0	0.6667
	Missed	C1	0.0014	0.0366

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
		C2	0.4866	0.513
		C3	0	0.3333
	False	C1	0.0208	0.1322
		C2	0.0003	0
		C3	0	946.3333
sorted052307channel_37	Precision	C1	1	0.998
		C2	0.0333	0.1814
		C3	0.944	0.8986
		C4	0.905	0.6895
		C5	0.0037	0
		C6	0	0.0013
	Recall	C1	0.9103	0.9744
		C2	0.4	0.975
		C3	0.3272	0.3783
		C4	0.5752	0.4799
		C5	0.2857	0
		C6	0	1
	Missed	C1	0.0897	0.0256
		C2	0.6	0.025
		C3	0.6728	0.6217

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
		C4	0.4248	0.5201
		C5	0.7143	0
		C6	0	0
	False	C1	0	0.002
		C2	11.625	4.4
		C3	0.0194	0.0427
		C4	0.0604	0.2161
		C5	76.1429	0
		C6	0	788
sorted052307channel_47	Precision	C1	0.0084	0.0893
		C2	0.098	0.0004
		C3	0.9977	0.9975
	Recall	C1	0.64	1
		C2	0.6432	0.0047
		C3	0.4135	0.5367
	Missed	C1	0.36	0
		C2	0.3568	0.9953
		C3	0.5865	0.4633
	False	C1	75.56	10.2
		C2	5.9202	11.1502

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
		C3	0.001	0.0013
sorted052307channel_54	Precision	C1	0.9927	0.9906
		C2	0.9574	0.988
	Recall	C1	0.9697	0.9917
		C2	0.9896	0.9863
	Missed	C1	0.0303	0.0083
		C2	0.0104	0.0137
	False	C1	0.0071	0.0094
		C2	0.044	0.012
sorted052307channel_63	Precision	C1	0.9981	0.9841
		C2	0.9233	0.9549
		C3	0.0395	0.0004
		C4	0.0106	0.0438
		C5	0	0
	Recall	C1	0.4143	0.2955
		C2	0.5822	0.9922
		C3	0.8387	0.0161
		C4	0.1	0.313
		C5	0	0

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
	Missed	C1	0.5857	0.7045
		C2	0.4178	0.0078
		C3	0.1613	0.9839
		C4	0.9	0.687
		C5	0	0
	False	C1	0.0008	0.0048
		C2	0.0484	0.0469
		C3	20.4032	39.2581
		C4	9.3565	6.8391
		C5	0	0
sorted052307channel_64	Precision	C1	0.9994	0.9685
		C2	0.8856	0.9351
		C3	0.9795	0.9842
		C4	0.0155	0.0017
		C5	0	0.0043
	Recall	C1	0.4947	0.5201
		C2	0.4533	0.621
		C3	0.9361	0.6684
		C4	0.9615	0.1154
		C5	0	0.3333

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
	Missed	C1	0.5053	0.4799
		C2	0.5467	0.379
		C3	0.0639	0.3316
		C4	0.0385	0.8846
		C5	0	0.6667
	False	C1	0.0003	0.0169
		C2	0.0585	0.0431
		C3	0.0196	0.0107
		C4	61.2308	67.6923
		C5	0	76.381
sorted052307channel_71	Precision	C1	0.0447	0.0542
		C2	1	1
	Recall	C1	1	1
		C2	0.4921	0.585
	Missed	C1	0	0
		C2	0.5079	0.415
	False	C1	21.3778	17.4667
		C2	0	0
sorted052307channel_46	Precision	C1	0.7076	0.7881

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
		C2	0.0005	0.1055
		C3	0.0266	0
		C4	0.1623	0.0246
		C5	0.9908	0.9492
		C6	0.232	0.855
		C7	0.001	0
	Recall	C1	0.8838	0.9876
		C2	0.0093	0.4112
		C3	0.75	0
		C4	0.8685	0.1036
		C5	0.2933	0.326
		C6	0.2723	0.4207
		C7	0.5	0
	Missed	C1	0.1162	0.0124
		C2	0.9907	0.5888
		C3	0.25	0
		C4	0.1315	0.8964
		C5	0.7067	0.674
		C6	0.7277	0.5793
		C7	0.5	0
	False	C1	0.3651	0.2656

Data set	Metric	Cluster	K-means with PCA transformed 32xN	K-means with PCA transformed, Eigenvalue weighted 9xN
		C2	20.1402	3.486
		C3	27.4167	0
		C4	4.4821	4.1155
		C5	0.0027	0.0175
		C6	0.9014	0.0714
		C7	519.5	0

Based on the same criterion of voting, the above table shows that the results of K-means on PCA transformed and eigenvalue weighted 9xN data are better than the results of K-means on the PCA transformed 32xN data in more than 53% of the cases. The following plot depicts this comparison.

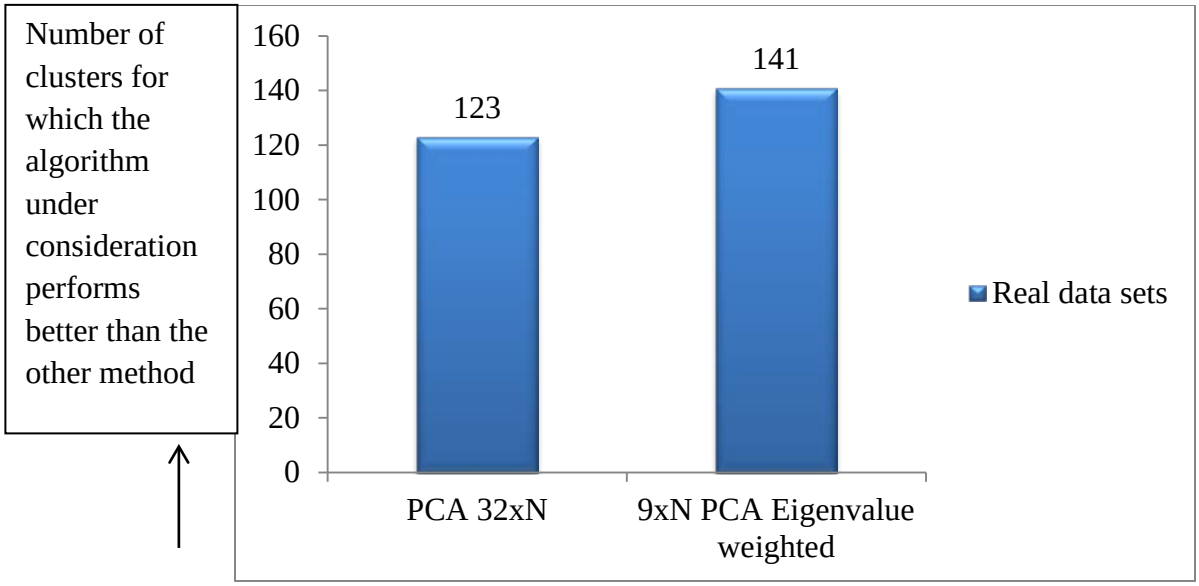

Figure 4.5: Comparison between K-means on PCA 32xN and 9xN PCA Eigenvalue weighted data

Following are the plots of the various clusters of data set sorted050207channel_12using different clustering approaches. It helps us to observe the clustering results visually. It also shows why cluster assignment is a big problem as we discussed in section 3.6.

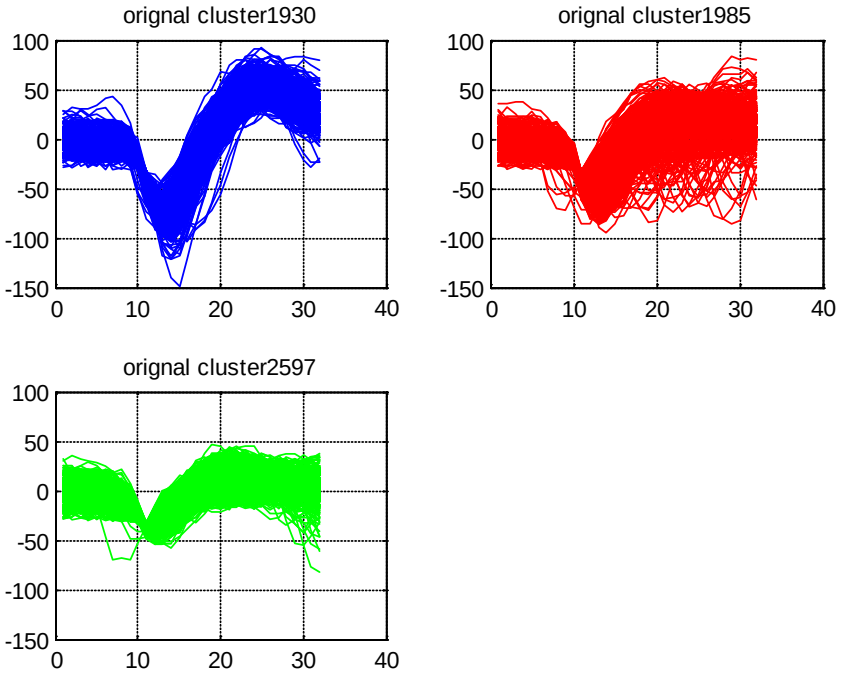

Figure 4.6: Plots of clustering based on gold standard (manual sorting).

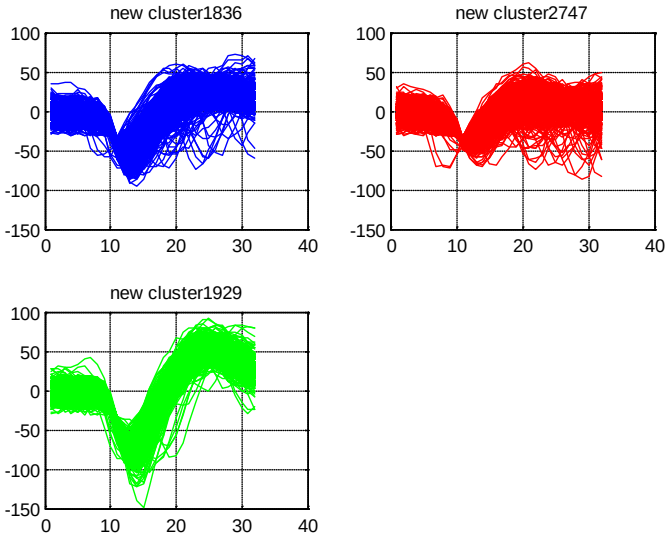

Figure 4.7: Plots of clustering based on K-means algorithm over original 32xN data

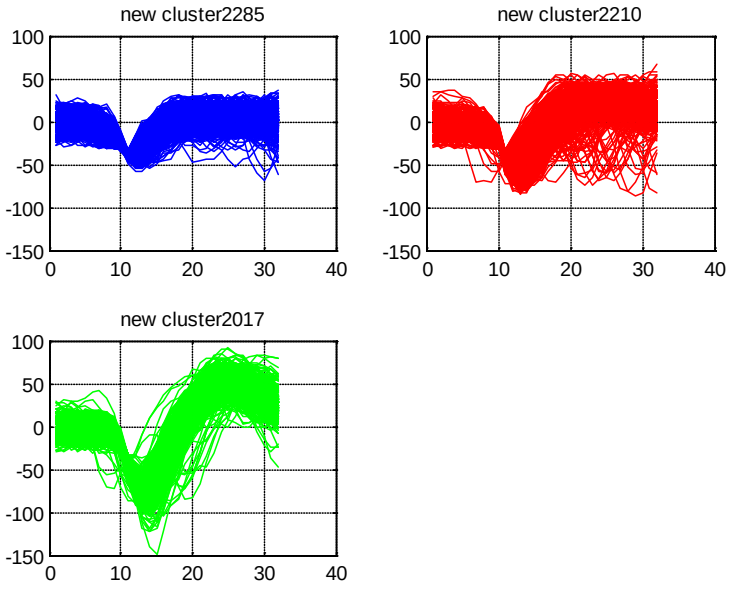

Figure 4.8: Plots of clustering based on entropy weighted 9xN data using K-means

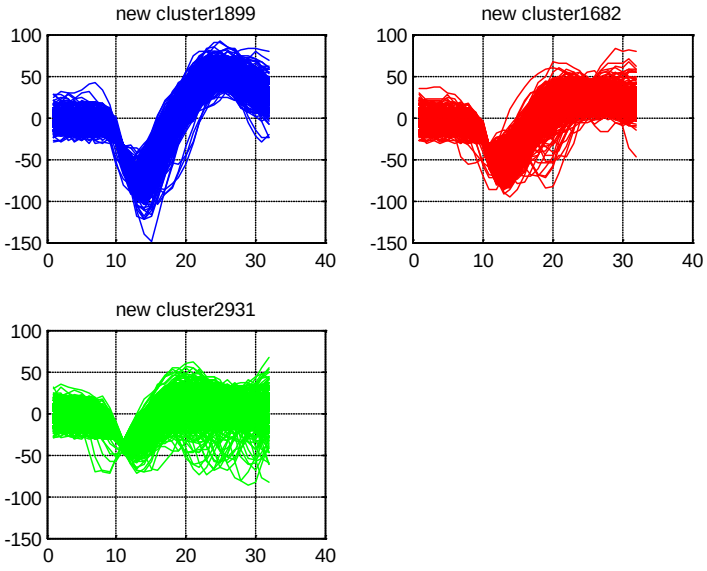

Figure 4.9: Plots of clustering based on PCA over original 32xN data

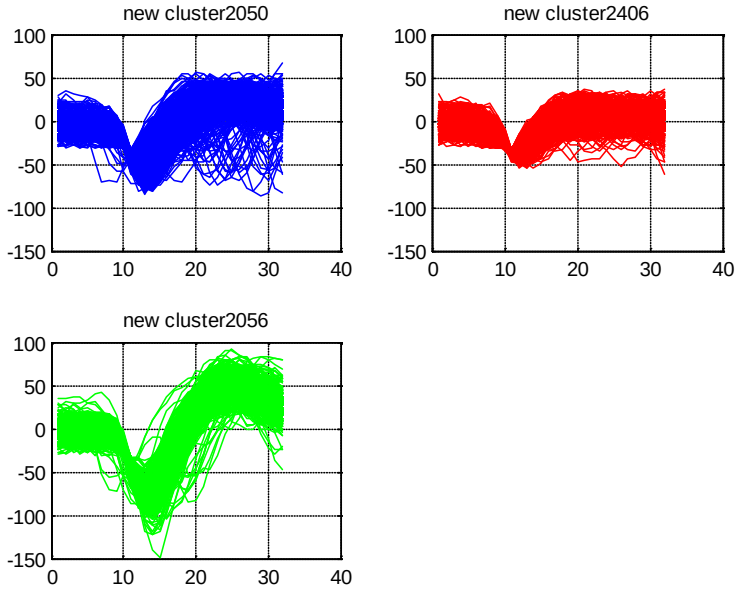

Figure 4.10: Plots of clustering based on eigenvalue weighted 9xN data using K-means

CHAPTER 5

CONCLUSION AND FUTURE DIRECTIONS

All the goals of the thesis mentioned in section 1.5 have been met. Nine different features were calculated from the original data sets based on their geometric properties. Two different intelligent weighting techniques were developed; first based on entropy of the features and second based on the principal component approach using eigenvalues. These techniques were then incorporated into the clustering procedure along with the features calculated separately. This resulted in an improved clustering performance in more than half of the data sets under study. Although it showed negative or no improvement in the remaining few data sets, the new approach still holds promise for further research due to the improvements shown over the baseline results.

With time and a few modifications, the clustering results could be improved even further to determine which spike comes from which neuron. A few other and possibly better methods of weighting the features could be implemented by selecting different heuristics such as mutual information. All the algorithms studied in this thesis perform poorly when the cluster sizes vary a lot in a given data set. K-means splits the data halfway between the cluster means leading to suboptimal splits. Other clustering algorithms such as Expectation-Maximization (EM) algorithm could be used to overcome these shortcomings since EM uses both variances and covariances and hence is able to accommodate clusters of variable size in a much better way than K-means.

APPENDIX A

SOURCE CODE

A.1 feature_calculation.m

```
% Code to calculate all features
% Authors: Zhijun Cai and Kaustubh Patwardhan

clear all;
clc;
close all;


%% Load data % data loading
[filename,pathname] = uigetfile('*.mat','Select Spike
Data','C:\Users\Kaustubh Patwardhan\Documents\Matlab\Kaustubh\data\');
load_file = [pathname filename];
load(load_file);

DataSets = res1; %comment it for simulated data, uncomment it for actual data

ClusColor = {'b.','r*','go','c.','k*','m.','y*'};
ClusColorLine = {'b','r','g','c','k','m','y'};

[sorted_ID indices] = sort(ID);

%Samples = res1';
wav = DataSets(:,indices);
id = sorted_ID;
[sampleLength,numSamples]   = size(DataSets); %for metrics
clear DataSets ;% clear ID


%% find the cluster number and corresponding vector, clear noise and
% unlabeled spikes
noise_ind = find(id==256);
id(noise_ind,:) = [];
wav(:,noise_ind) = [];

err_ind = find(id==255);
id(err_ind,:) = [];
wav(:,err_ind) = [];

err_ind = find(id==0);
id(err_ind,:) = [];
wav(:,err_ind) = [];

[sampleLength,numSamples]   = size(wav); % dimensions of data

numClusters    = max(id); % number of clusters in the data
idcopy   = id;
for ii = 1:numClusters;
     IDold{ii}  = find(id==ii);
```

```
end

%% find cluster sizes

for i=1:numClusters
clust_size_orig(i,1) = nnz(sorted_ID==i);
end

% calculate features based on the geometric properties of the data

%%  1 find the minimum point happening in the first half
% set the spike with minimum happening before the half 1 and ones minimum
% happening after half 0

[minM, minMind] = min(wav);
minL            = find(minMind<=16);
minR            = find(minMind>16);
minPos          = ones(1,numSamples);
minPos(minR)    = 0;

%% 2 number of minimums

minNums = ones(1,numSamples);
for ii = 1:numSamples
  OutInd  = localMinimum(wav(:,ii),5);
  minNums(1,ii) = numel(OutInd)+1;
  clear OutInd;
end

minNums         = minNums/max(minNums);

%% 3 magnitude in frequency domain

wavFFT          = abs(fft(wav));
[magF, maxMind] = max(wavFFT(1:16,:));
magF            = magF/max(magF) ;

%% 4 magnitude in time domain( maximum value minus   minmum value)

magT         = max(wav) - min(wav);
magT         = magT/max(magT);

%% 5 varaiance

varWav         = var(wav);
varWav         = varWav/max(varWav);

%% 6, 7, 8, 9 Right Peak and left peak difference, right peak slope and the
%   polynomial fitting for the peak-to-peak, valley-to-peak

   slopeR          = zeros(1,numSamples);
   peakRLD         = zeros(1,numSamples);
   MinCur          = zeros(1,numSamples);
   MinEndCur       = zeros(1,numSamples);
```

```
   for ii = 1:numSamples
        [peakL, indL]  = max(wav(1:minMind(ii),ii));
        [peakR, indR]  = max(wav(minMind(ii):end,ii));
        peakRLD(ii)    = peakL - peakR;
        slopeR(ii)     = (peakR-minM(ii))/(indR + minMind(ii) - 1);
        x              = (indL:indR+minMind(ii)-1)/32;
        y              = wav(indL:indR+minMind(ii)-1,ii);
        pp             = polyfit(x',y,2);  clear x y;
        MinCur(ii)     = pp(1);
        x2              = (minMind(ii):32)/32;
        y2              = wav(minMind(ii):32,ii);
        if minMind(ii)>=30
          MinEndCur(ii)  = 0;
        else
          pp2            = polyfit(x2',y2,2);  clear x2 y2;
          MinEndCur(ii)  = pp2(1);
        end
   end

 peakRLD  = peakRLD/max(abs(peakRLD));
 if mean(peakRLD)<0; peakRLD = -peakRLD ;end;

 slopeR   = slopeR/max(slopeR);

 MinCur   = MinCur/max(abs(MinCur));
 if mean(MinCur)<0; MinCur = -MinCur ;end;

 MinEndCur  = MinEndCur/max(abs(MinEndCur));
 if mean(MinEndCur)<0; MinEndCur = -MinEndCur ;end;

 %% 10 Total absolute area under positive and negative going peaks
  % find mean level of each spike

 spike_mean_level = mean(wav);
 total_abs_area = (sum(abs(wav -
repmat(spike_mean_level,sampleLength,1))))/sampleLength;
 total_abs_area = total_abs_area/max(total_abs_area);

 %% 11 RMS distance

 sq_dist = ((wav-repmat(spike_mean_level,sampleLength,1)).*(wav-
repmat(spike_mean_level,sampleLength,1)));
 rms_dist = sqrt(sum(sq_dist))/sampleLength;
 rms_dist = rms_dist/max(rms_dist);

%% Clustering results;

% All Features [minPos; minNums; magF; magT;varWav; peakRLD; slopeR; MinCur ;
% MinendCur; total abs area; rms dist]

 AllFeatures    = [];
    %minPosW     = 1;   AllFeatures = [AllFeatures; minPos*minPosW];
    %minNumsW    = 1;   AllFeatures = [AllFeatures; minNums*minNumsW];
    magFW      = 1;   AllFeatures = [AllFeatures; magF*magFW ];
```

```
    magTW      = 1;   AllFeatures = [AllFeatures; magT*magTW];
    varWavW    = 1;   AllFeatures = [AllFeatures; varWav*varWavW ];
    peakRLDW   = 1;   AllFeatures = [AllFeatures; peakRLD*peakRLDW];
    slopeRW    = 1;   AllFeatures = [AllFeatures; slopeR*slopeRW];
    MinCurW    = 1;   AllFeatures = [AllFeatures; MinCur*MinCurW];
    MinEndCurW = 1;   AllFeatures = [AllFeatures; MinEndCur*MinEndCurW];
    total_abs_area_W = 1; AllFeatures = [AllFeatures;
total_abs_area*total_abs_area_W];
    rms_dist_W      = 1;   AllFeatures = [AllFeatures; rms_dist*rms_dist_W ];

entropy_weighting;
```

A.2 entropy_weighting.m

```
%% Calculate entropy of all features
% weight each feature with its entropy value
% Kaustubh Patwardhan

 [numSamples,sampleLength] = size(AllFeatures');
 samples=AllFeatures';

numBins = 100;
bin_size = 2/numBins; % 2 is the range of sample values, max=1 and min=-1

edge = -1:bin_size:1;
prob_mass_func=zeros(numBins+1,sampleLength);

% calculate probability mass function using a histogram

for i=1:sampleLength
    hist(samples(:,i),edge)
    prob_mass_func(:,i)=(histc(samples(:,i),edge))/numSamples;
end

bin_entropy = zeros(numBins+1,sampleLength);

% calculate entropy of each feature

for i=1:sampleLength
    for j=1:numBins+1
        bin_entropy(j,i) = (prob_mass_func(j,i)*log(prob_mass_func(j,i)));
    end
end

bin_entropy(isnan(bin_entropy))=0;
entropy = -(sum(bin_entropy));
total_entropy = sum(entropy);
[dominant_features,dominant_features_ind]=sort(entropy,'descend');
disp('Entropy value for each feature'), disp(entropy')

% weights each feature vector with its entropy value

for i=1:sampleLength
     samples(:,i) = samples(:,i) * entropy(i);
```

```
end

cluster_spikes;
```

A.3 eigenvalue_weighting.m

```
% Kaustubh Patwardhan
% Calculate covariance matrix, its eigenvectors and eigenvalues
% Obtain transformed data
% weight each PC / feature vector with its eigenvalue

%% PCA using covariance method

[signals, PC,V,mn]=pca(AllFeatures);
samples=signals';
[numSamples,sampleLength] = size(samples);
V=V/max(V);

%% EV weighting

for i=1:sampleLength
    samples(:,i) = samples(:,i) * V(i,:);
end

cluster_spikes;
```

A.4 pca.m

```
% Kaustubh Patwardhan
%==============inputs are M*N matrix:M dimensions,N sample numbers=========
%
function [signals, PC,V, EV, mn]=pca(data)
% pca1-perform pca using covariance
% data- M*N matrix of input data
% signals- M*N projected data
% PC-each column is a PC (what is a PC?)
%  V-M*1 matrix of variances

[M,N]=size(data);

%subtract off the mean for each dimension
mn=mean(data,2); %find the mean along the 2nd(row)dimension
data=data-repmat(mn,1,N); %subtract mean from data

%calculate the covariance matrix
covariance=1/(N-1)*(data*data'); %covariance matrix:M*M(12*12)

%find the eigenvectors and eigenvalues
%V: the covariance of each component PC:the corresponding row vector
[PC,V]=eig(covariance); %PC:eigenvectors; V:eigenvalues
```

```
%extract diagonal of matrix as vector
EV=diag(V); %make a size(V)*size(V) diagonal matrix

%sort the variances in decreasing order
%For matrices, SORT(X) sorts each column of X in ascending order.
[junk, rindices]=sort(-1*EV); %rindices are the index of components of V
EV=EV(rindices);
PC=PC(:,rindices);

%profect the original data set
signals=PC'*data;
```

A.5 cluster_spikes.m

```
%% Kmeans algorithm

[IDX, Mu] = kmeans(samples,numClusters,'Distance','cityblock');

%% cluster assigment

cluster_assignment;
```

A.6 cluster_assignment.m

```
% compute precision and recall for each observed id being assigned to
groundtruth-id's

    % start with GT id 1, find all the corresponding observed IDs
        % #observed nnz(id==1), will be true-positives (for observed id = 1)
        % #observed nnz(id==2), will be true-positives (for observed id = 2)
    % repeat for GT IDs 2,..

% compute the cost matrix for assigment between Obesrved IDs and GT IDs

% use the Hungarian algorithm for making the observed id to GT id assignment

%   gt1  gt2  gt3
%o1----|----|----|
%o2----|----|----|
%o3----|----|----|
%o4----|----|----|

% [Matching,Cost] = Hungarian(Perf)

% sorted_id = GT arranged in ascending order
% IDX = observed IDs

%calculate true positives/negatives and false positives/negatives,
%precision and recall
% Kaustubh Patwardhan

%find cluster size
for i=1:numClusters
```

```
    clust_size_orig(i,1) = nnz(id==i);
end
for i=1:numClusters
    clust_size_clustered(i,1) = nnz(IDX==i);
end

total_samples = numel(ID);

ID_precision = zeros(numClusters); % define matrices
ID_recall = zeros(numClusters);
ID_f_measure = zeros(numClusters);
Adj_f_measure = zeros(numClusters);
adj = zeros(1,numClusters);
ID_rand_index = zeros(numClusters);
missed_classification= zeros(numClusters);
false_classification= zeros(numClusters);
ID_Matthews_corr_coeff = zeros(numClusters);

for i=1:numClusters
    adj(i) = (exp(-(clust_size_orig(i)/total_samples))/exp(-1));
end

for i=1:numClusters
    for j=1:numClusters

        ID_tp = nnz(id==i & IDX==j); % calculate true positives, true
negatives, false positives and false negatives
        ID_tn = nnz(id~=i & IDX~=j);
        ID_fp = nnz(id~=i & id~=0 & IDX==j);
        ID_fn = nnz(id==i & IDX~=j);

        precision = (ID_tp)/((ID_tp)+(ID_fp));
        recall = (ID_tp)/((ID_tp)+(ID_fn));
        missed_classification(i,j) = (ID_fn)/clust_size_orig(i);%
        false_classification(i,j) = (ID_fp)/clust_size_orig(i);%

        f_score = (2*precision*recall) / (precision + recall);
        ID_f_measure(i,j) = f_score;
        Adj_f_measure(i,j) = f_score*adj(i);

        %ID_f_measure(i,j) = (2*precision*recall) / (precision + recall);
        ID_rand_index(i,j) = (ID_tp + ID_tn)/(ID_tp + ID_tn + ID_fp + ID_fn);
        ID_Matthews_corr_coeff(i,j) = ((ID_tp*ID_tn)-
(ID_fp*ID_fn))/sqrt((ID_tp+ID_fp)*(ID_tp+ID_fn)*(ID_tn+ID_fp)*(ID_tn+ID_fn));
        ID_precision(i,j) = precision;
        ID_recall(i,j) = recall;

    end
end

%cost_mat = -ID_f_measure;
cost_mat = -Adj_f_measure; % cost function matrix, cost function = -(adjusted
F measure)
%cost_mat = -ID_Matthews_corr_coeff;
cost_mat(isnan(Adj_f_measure)) = inf;
```

```
%cost_mat(isnan(ID_f_measure)) = inf;
[Matching,Cost] = Hungarian(cost_mat); % hungarian algorithm to assign
clusters

f_measure = ID_f_measure;
f_measure(isnan(ID_f_measure)) = 0; % for display purposes

%disp('Precision Matrix')
disp(diag(Matching*ID_precision'))

%disp('Recall Matrix')
disp(diag(Matching*ID_recall'))

% disp('F Measure Matrix'), disp(diag(Matching*f_measure'))
% disp('Rand Index Matrix'), disp(diag(Matching*ID_rand_index'))

%disp('Missed')
disp(diag(Matching*missed_classification'))

%disp('False-classified')
disp(diag(Matching*false_classification'))

%disp('Cost function Matrix'), disp(cost_mat)

%disp('Cluster assignment that minimizes the cost')
disp(Matching)

cluster_plots;
```

A.7 Hungarian.m [28]

```
function [Matching,Cost] = Hungarian(Perf)
%
% [MATCHING,COST] = Hungarian_New(WEIGHTS)
%
% A function for finding a minimum edge weight matching given a MxN Edge
% weight matrix WEIGHTS using the Hungarian Algorithm.
%
% An edge weight of Inf indicates that the pair of vertices given by its
% position have no adjacent edge.
%
% MATCHING return a MxN matrix with ones in the place of the matchings and
% zeros elsewhere.
%
% COST returns the cost of the minimum matching

% Written by: Alex Melin 30 June 2006


 % Initialize Variables
 Matching = zeros(size(Perf));

% Condense the Performance Matrix by removing any unconnected vertices to
```

```
% increase the speed of the algorithm

  % Find the number in each column that are connected
    num_y = sum(~isinf(Perf),1);
  % Find the number in each row that are connected
    num_x = sum(~isinf(Perf),2);

  % Find the columns(vertices) and rows(vertices) that are isolated
    x_con = find(num_x~=0);
    y_con = find(num_y~=0);

  % Assemble Condensed Performance Matrix
    P_size = max(length(x_con),length(y_con));
    P_cond = zeros(P_size);
    P_cond(1:length(x_con),1:length(y_con)) = Perf(x_con,y_con);
    if isempty(P_cond)
      Cost = 0;
      return
    end

    % Ensure that a perfect matching exists
      % Calculate a form of the Edge Matrix
      Edge = P_cond;
      Edge(P_cond~=Inf) = 0;
      % Find the deficiency(CNUM) in the Edge Matrix
      cnum = min_line_cover(Edge);

      % Project additional vertices and edges so that a perfect matching
      % exists
      Pmax = max(max(P_cond(P_cond~=Inf)));
      P_size = length(P_cond)+cnum;
      P_cond = ones(P_size)*Pmax;
      P_cond(1:length(x_con),1:length(y_con)) = Perf(x_con,y_con);

%************************************************
% MAIN PROGRAM: CONTROLS WHICH STEP IS EXECUTED
%************************************************
  exit_flag = 1;
  stepnum = 1;
  while exit_flag
    switch stepnum
      case 1
        [P_cond,stepnum] = step1(P_cond);
      case 2
        [r_cov,c_cov,M,stepnum] = step2(P_cond);
      case 3
        [c_cov,stepnum] = step3(M,P_size);
      case 4
        [M,r_cov,c_cov,Z_r,Z_c,stepnum] = step4(P_cond,r_cov,c_cov,M);
      case 5
        [M,r_cov,c_cov,stepnum] = step5(M,Z_r,Z_c,r_cov,c_cov);
      case 6
        [P_cond,stepnum] = step6(P_cond,r_cov,c_cov);
      case 7
        exit_flag = 0;
    end
```

```
  end

% Remove all the virtual satellites and targets and uncondense the
% Matching to the size of the original performance matrix.
Matching(x_con,y_con) = M(1:length(x_con),1:length(y_con));
Cost = sum(sum(Perf(Matching==1)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   STEP 1: Find the smallest number of zeros in each row
%           and subtract that minimum from its row
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [P_cond,stepnum] = step1(P_cond)

  P_size = length(P_cond);

  % Loop throught each row
  for ii = 1:P_size
    rmin = min(P_cond(ii,:));
    P_cond(ii,:) = P_cond(ii,:)-rmin;
  end

  stepnum = 2;

%*************************************************************************
%   STEP 2: Find a zero in P_cond. If there are no starred zeros in its
%           column or row start the zero. Repeat for each zero
%*************************************************************************

function [r_cov,c_cov,M,stepnum] = step2(P_cond)

% Define variables
  P_size = length(P_cond);
  r_cov = zeros(P_size,1);  % A vector that shows if a row is covered
  c_cov = zeros(P_size,1);  % A vector that shows if a column is covered
  M = zeros(P_size);        % A mask that shows if a position is starred or
primed

  for ii = 1:P_size
    for jj = 1:P_size
      if P_cond(ii,jj) == 0 && r_cov(ii) == 0 && c_cov(jj) == 0
        M(ii,jj) = 1;
        r_cov(ii) = 1;
        c_cov(jj) = 1;
      end
    end
  end

% Re-initialize the cover vectors
  r_cov = zeros(P_size,1);  % A vector that shows if a row is covered
  c_cov = zeros(P_size,1);  % A vector that shows if a column is covered
  stepnum = 3;

%*************************************************************************
```

```
%   STEP 3: Cover each column with a starred zero. If all the columns are
%           covered then the matching is maximum
%**************************************************************************

function [c_cov,stepnum] = step3(M,P_size)

  c_cov = sum(M,1);
  if sum(c_cov) == P_size
    stepnum = 7;
  else
    stepnum = 4;
  end

%**************************************************************************
%   STEP 4: Find a noncovered zero and prime it.  If there is no starred
%           zero in the row containing this primed zero, Go to Step 5.
%           Otherwise, cover this row and uncover the column containing
%           the starred zero. Continue in this manner until there are no
%           uncovered zeros left. Save the smallest uncovered value and
%           Go to Step 6.
%**************************************************************************
function [M,r_cov,c_cov,Z_r,Z_c,stepnum] = step4(P_cond,r_cov,c_cov,M)

P_size = length(P_cond);

zflag = 1;
while zflag
    % Find the first uncovered zero
      row = 0; col = 0; exit_flag = 1;
      ii = 1; jj = 1;
      while exit_flag
          if P_cond(ii,jj) == 0 && r_cov(ii) == 0 && c_cov(jj) == 0
            row = ii;
            col = jj;
            exit_flag = 0;
          end
          jj = jj + 1;
          if jj > P_size; jj = 1; ii = ii+1; end
          if ii > P_size; exit_flag = 0; end
      end

    % If there are no uncovered zeros go to step 6
      if row == 0
        stepnum = 6;
        zflag = 0;
        Z_r = 0;
        Z_c = 0;
      else
        % Prime the uncovered zero
        M(row,col) = 2;
        % If there is a starred zero in that row
        % Cover the row and uncover the column containing the zero
          if sum(find(M(row,:)==1)) ~= 0
            r_cov(row) = 1;
            zcol = find(M(row,:)==1);
            c_cov(zcol) = 0;
```

```
          else
            stepnum = 5;
            zflag = 0;
            Z_r = row;
            Z_c = col;
          end
      end
end

%*************************************************************************
% STEP 5: Construct a series of alternating primed and starred zeros as
%         follows.  Let Z0 represent the uncovered primed zero found in Step
4.
%         Let Z1 denote the starred zero in the column of Z0 (if any).
%         Let Z2 denote the primed zero in the row of Z1 (there will always
%         be one).  Continue until the series terminates at a primed zero
%         that has no starred zero in its column.  Unstar each starred
%         zero of the series, star each primed zero of the series, erase
%         all primes and uncover every line in the matrix.  Return to Step 3.
%*************************************************************************

function [M,r_cov,c_cov,stepnum] = step5(M,Z_r,Z_c,r_cov,c_cov)

  zflag = 1;
  ii = 1;
  while zflag
    % Find the index number of the starred zero in the column
    rindex = find(M(:,Z_c(ii))==1);
    if rindex > 0
      % Save the starred zero
      ii = ii+1;
      % Save the row of the starred zero
      Z_r(ii,1) = rindex;
      % The column of the starred zero is the same as the column of the
      % primed zero
      Z_c(ii,1) = Z_c(ii-1);
    else
      zflag = 0;
    end

    % Continue if there is a starred zero in the column of the primed zero
    if zflag == 1;
      % Find the column of the primed zero in the last starred zeros row
      cindex = find(M(Z_r(ii),:)==2);
      ii = ii+1;
      Z_r(ii,1) = Z_r(ii-1);
      Z_c(ii,1) = cindex;
    end
  end

  % UNSTAR all the starred zeros in the path and STAR all primed zeros
  for ii = 1:length(Z_r)
    if M(Z_r(ii),Z_c(ii)) == 1
      M(Z_r(ii),Z_c(ii)) = 0;
    else
      M(Z_r(ii),Z_c(ii)) = 1;
```

```
    end
  end

  % Clear the covers
  r_cov = r_cov.*0;
  c_cov = c_cov.*0;

  % Remove all the primes
  M(M==2) = 0;

stepnum = 3;

% ************************************************************************
% STEP 6: Add the minimum uncovered value to every element of each covered
%         row, and subtract it from every element of each uncovered column.
%         Return to Step 4 without altering any stars, primes, or covered
lines.
%************************************************************************

function [P_cond,stepnum] = step6(P_cond,r_cov,c_cov)
a = find(r_cov == 0);
b = find(c_cov == 0);
minval = min(min(P_cond(a,b)));

P_cond(find(r_cov == 1),:) = P_cond(find(r_cov == 1),:) + minval;
P_cond(:,find(c_cov == 0)) = P_cond(:,find(c_cov == 0)) - minval;

stepnum = 4;

function cnum = min_line_cover(Edge)

  % Step 2
    [r_cov,c_cov,M,stepnum] = step2(Edge);
  % Step 3
    [c_cov,stepnum] = step3(M,length(Edge));
  % Step 4
    [M,r_cov,c_cov,Z_r,Z_c,stepnum] = step4(Edge,r_cov,c_cov,M);
  % Calculate the deficiency
    cnum = length(Edge)-sum(r_cov)-sum(c_cov);
```

A.8 cluster_plots.m

```
%% Plot Clusters

plottype1 = {'blue';'red';'black';'m'};
plottype2 = {'bo';'r*';'k+';'m^';'gs';'yd';'cp'};

nn = ceil(numClusters/2);

figure;

% original clusters

for ii = 1:numClusters
```

```
    subplot(nn,2,ii);
    plot(wav(:,IDold{ii}),ClusColorLine{ii});grid;
    %ylim([-200 200]);
    title(strcat('orignal cluster',num2str(numel(IDold{ii}))));
end

for ii = 1 : numClusters
    IDnew{ii} = find(IDX==ii);
end

figure;

% new clusters

for ii = 1:numClusters
    subplot(nn,2,ii);
    plot(wav(:,IDnew{ii}),ClusColorLine{ii}); grid;
    title(strcat('new cluster',num2str(numel(IDnew{ii}))));
end
```

REFERENCES

1. Busskamp V, Duebel J, Balya D, et al. Genetic reactivation of cone photoreceptors restores visual responses in retinitis pigmentosa. Science. 2010;329(5990):413–417.

2. "Sensory Reception: Human Vision: Structure and functioon of the Human Eye" vol. 27, Encyclopaedia Britannica, 1987

3. Barnett MW, Larkman PM (June 2007). "The action potential". Pract Neurol 7 (3): 192–7.

4. Koenekoop, R.K. (2003). Novel RPGR mutations with distinct retinitis pigmentosa phenotypes in French-Canadian families. American journal of ophthalmology 136(4), pp. 678-68

5. Stasheff SF. Emergence of sustained spontaneous hyperactivity and temporary preservation of OFF responses in ganglion cells of the retinal degeneration (rd1) mouse. Journal of Neurophysiology. 2008;99(3):1408–1421.

6. Jacobson SG, Cideciyan AV (2010) Treatment possibilities for retinitis pigmentosa. N Engl J Med 363:1669–1671

7. Bramall AN, Wright AF, Jacobson SG, McInnes RR. The genomic, biochemical, and cellular responses of the retina in inherited photoreceptor degenerations and prospects for the treatment of these disorders. Annu Rev Neurosci 2010;33:441-472

8. Cideciyan AV. Leber congenital amaurosis due to RPE65 mutations and its treatment with gene therapy. Prog Retin Eye Res 2010;29:398-427

9. 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.

10. Buzsaki G (2004) Large-scale recording of neuronal ensembles. Nature Neuroscience

7:446-451.

11. Spike Sorting. D. N. Hill, D. Kleinfeld and S. B. Mehta. In Observed Brain Dynamics by P. P. Mitra and H. Bokil, 2007, Oxford Press, 9:257-270.

12. Hartigan, J. A. (1975). Clustering Algorithms. Wiley. ISBN 0-471-35645-X. MR0405726

13. Jolliffe, I. T. (1986). Principal Component Analysis. Springer-Verlag. pp. 487. doi:10.1007/b98835. ISBN 978-0-387-95442-4.

14. Corinna Cortes and V. Vapnik, "Support-Vector Networks", Machine Learning, 20, 1995.

15. Wood F, Black MJ, Vargas-Irwin C, Fellows M, Donoghue JP. On the variability of manual spike sorting. IEEE Trans. Biomed. Eng. 2004;51:912–8.

16. K. D. Harris, D. A. Henze, J. Csicsvari, H. Hirase, and G. Buzsáki, “Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements,” J. Neurophysiol., vol. 81, no. 1, pp. 401–414, 2000.

17. Teknomo, Kardi. K-Means Clustering Tutorials. http:\\people.revoledu.com\kardi\ tutorial\kMean\

18. Lindsay I Smith. A tutorial on Principal Components Analysis, http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

19. Hans-Peter Kriegel , Peer Kröger , Erich Schubert , Arthur Zimek, A General Framework for Increasing the Robustness of PCA-Based Correlation Clustering Algorithms, Proceedings of the 20th international conference on Scientific and Statistical Database Management, July 09-11, 2008, Hong Kong, China

20. Horváth (2003) in Suykens et al. p 392

21. M. S. Lewicki, “A review of methods for spike sorting: the detection and classification of neural action potentials,” Network, vol. 9, no. 4, pp. R53–R78, 1998.

22. Schmidt EM. Computer separation of multi-unit neuroelectric data: a review. Journal of Neuroscience Methods. 1984;12(2):95–111.

23. O’Connell, R.J., Kocsis, W. A. and Schoenfeld, R.L. (1973) Minicomputer identification and time of nerve impulses mixed in a single recording channel, Proc. IEEE, 61: 1615-1621.

24. Schneider, T.D, Information theory primer with an appendix on logarithms, National Cancer Institute, 14 April 2007.

25. Qingjiu Zhang, Shiliang Sun: “Weighted Data Normalization Based on Eigenvalues for Artificial Neural Network Classification”. Proceedings of the 16th International Conference on Neural Information Processing: Part I, pp. 349-356, 2009.

26. R.E. Burkard, M. Dell'Amico, S. Martello: Assignment Problems. SIAM, Philadelphia (PA.) 2009. ISBN 978-0-89871-663-4

27. Olson, David L.; Delen, Dursun ”Advanced Data Mining Techniques” Springer; 1 edition (February 1, 2008), page 138, ISBN 3540769161

28. Alex Melin, Hungarian Algorithm: http://www.mathworks.com/matlabcentral/fileexchange/11609