A MULTIFACETED INTERACTIVE ANALYSIS OF THE BEHAVIOR OF A SINGLE NEURON MODEL

Elizabeth A. Gifford
Brandeis University, Department of Computer Science
Senior Honors Thesis, May 2006

TABLE OF CONTENTS

A MULTIFACETED INTERACTIVE ANALYSIS OF THE BEHAVIOR OF A SINGLE NEURON MODEL

Elizabeth A. Gifford

Introduction

In the course of my research, I developed several techniques to explore the contents of a large database of the various activity patterns of a single neuron model. I used feature extraction and k-means clustering to assign attributes to neuron models that could vary as the model's inputs vary and developed a visualization software tool to facilitate the exploration of that relationship.

Biological Background

Neurons make up the communication networks within living organisms, and are responsible for carrying information around the body. Information is conducted through the pathways by both electrical and chemical signaling between the cells (Kandel *et al.*, 2000). This signaling relies on changes in the electrical potential difference across cell membranes, and depends on the cells' ability to change its behavior based on changes in stimuli (Kandel *et al.*, 2000). The membrane potential, and thus the signal, is influenced by ion channels that exist in the cell's membrane (Kandel *et al.*, 2000). Ion channels control the current flow in and out of the cell, which pushes the electrical potential across the membrane away from its resting value (the electrical potential across the membrane in the absence of signaling) (Kandel *et al.*, 2000). As a result, the changes in the spontaneous firing pattern of a cell are determined by the membrane conductances (Prinz *et al.*).

Simulations and experiments have shown that the cell's activity can be drastically altered by small changes in one or more currents (Prinz *et al.*, 2003). Alternatively, wide variances in sets of conductances can also produce very similar types of activity (Prinz *et al.*, 2003). For this reason, it is important to understand how rigidly the conductances need to be controlled in an experiment, and how best to regulate them in order to ensure that the desired results are obtained. To understand these restrictions, it is important to understand how the properties of a neuron depend on membrane conductances (Prinz *et al.*, 2003).

Creating the Data Set

In their 2003 paper, Prinz *et al.* describe a brute force technique to generate a simulated database of neuronal activity which would allow for a detailed analysis of the effect of different combinations of conductances on the activity patterns of a neuron model.

The database was generated by independently varying the maximum values for the eight conductances in a model based on experiment measurements from lobster stomatogastric neurons (Prinz *et al.*, 2003). The values were varied over six equally spaced values ranging from 0 mS/cm^2 to maximum values that were specific to the particular conductance (Prinz *et al.*, 2003). This resulted in 6^8 (about 1.7 million) possible combinations of maximum conductance values, all of which were simulated and the results stored for future use (Prinz *et al.*, 2003).

The following terminology will be used throughout the paper, and is consistent with the definitions given by Prinz *et al*. A single combination of possible conductance values is referred to as a "neuron" or "model neuron" while the entire data set generated as a result of their technique will be called the "model" (Prinz *et al.*, 2003).

Once each neuron model had been simulated, several pieces of information about that neuron were written to a text file containing the database. Included in this information was a list of the time (t) and voltage value (V) of each local maximum or minimum value of the membrane potential (Prinz *et al.*, 2003). The different activity types (membrane potential activity patterns) were also classified using the local extrema, and assigned a classification of silent (no change in voltage level), tonically spiking (regularly spaced single rises and falls in voltage), bursting (regularly spaced groups of rising and falling voltage levels) or irregular (not definable by the other three categories) (Prinz *et al.*, 2003).

Because of the wide variety of conductance values and activity patterns contained in this database, much insight into the problem of how a neuron's conductances can affect its' activity pattern can be gleaned from this simulated data. The authors note that the database could be searched for different combinations of neuron properties (including, but not limited to, activity type, spike frequency or burst frequency) in order to find combinations of conductances that yield desired activity patterns (Prinz *et al.*, 2003). Additionally, these properties could be examined as different conductance values vary in order to gain insight to the direct effect of conductance values on any of the stored neuron properties.

Computational Goals

An interesting way to explore this database would be to find a way to examine the effect of slowly changing conductance values on different aspects of the neuron's behavior. This would allow one to gain some insight into the relationship (or lack thereof) between a particular conductance or set of conductances on a behavioral attribute.

One technique for analyzing this type of database viewing was proposed by Langton *et al.* in their paper published in the Visual Information Expert Workshop (VIEW) proceedings in 2006. This technique proposes that each neuron model be represented by a single pixel that can be colored according to a particular attribute specified by the user, resulting in a display of 1.7 million pixels allowing the user to see overall groupings in the data set. However, one drawback of this method is that the database constructed by Prinz *et al.* is admittedly fairly coarsely sampled. Computational limits made allowing for only six conductance values necessary, but on wide ranges this sampling does not make for a very accurate representation of the changes in behavior. To solve this problem, I decided that instead of creating the large 8-dimensional space of the original database, I would begin by examining small "slices" of the conductance space where only a few parameters were allowed to vary. The advantage of this approach is that the values could be much more densely sampled to allow for a better idea of the effect of conductance change on behavior. For these studies I used techniques similar to those developed by Astrid *et al.* but created the smaller densely sampled databases myself, which allowed me to decide what information to calculate and save for each of the database slices.

It was also important to decide on interesting trends to examine within the data set. For this project I decided initially to focus on the period length of periodic behaviors, the number of spikes per second, and the number of spikes per period. By looking at how these attributes changed as conductance values changed I would be able to see whether any of them were directly related to certain conductances or combinations of conductances.

Having the large data set created by Prinz *et al.* at my disposal also made for a rich data mining opportunity. In addition to the hand selected attributes mentioned above, I also decided to use the same extrema used by Astrid *et al.* to classify the different behaviors as material for my own data mining experiment. By using this approach I hoped to find out whether there were other highly similar "classes" that a human observer might not pick out by looking at the data manually. Additionally, I hoped to be able to create a system to automatically assign labels to the different neuron behaviors seen in the database which could in turn be used to see whether any of these newly discovered attributes were directly related to the conductances in the same method I planned to use on the other, more clearly defined, attributes.

Software Design

Compression

The simulator samples data at a very high frequency (20,000 Hz) that retains a lot of information, but is slow to read from file or plot. Additionally, if the entire database were to be stored in an uncompressed format, it would take about 35 terabytes of disk space. However, re-simulating the data each time it is needed is not really a viable option either, as it can take up to 90 seconds to generate 30 seconds worth of data on current consumer grade machines.

In order to give the software a more interactive feel and save on disk space, the voltage data was compressed using a technique that ensured that the voltage trace never deviated by more than one percent from the original simulated data and forcedly stored any local minimum or maximum points in the data (See Algorithm 1).

```
voltage = vector of voltage values
compressed = empty vector
P1 = voltage(1)
maxSlope = positive infinity
minSlope = negative infinity
slope = 0
append(P1, compressed)

for all points (i):
     slope = Slope(P1, Pi)
     if Slope(P1, Pi+error) < maxSlope
          maxSlope = Pi+error
     if Slope(P1, Pi-error) > minSlope
          minSlope = Pi-error
     if (slope > maxSlope) || (slope < minSlope) || isExtrema(P1)
          P1 = Pi-1
          append(P1, compressed)
          maxSlope = positive infinity
          minSlope = negative infinity
          slope = 0
```

Algorithm 1 - A pesudocode representation of the compression algorithm

The compression algorithm works by continually narrowing the range of allowable slopes until a line drawn between two points falls outside of the range of error. Starting with minimum and maximum slopes of positive and negative infinity, the slopes between the

first point in a curve and the next point plus or minus a constant error value are calculated. The minimum and maximum slope values are updated only when the next calculated minimum or maximum narrows the range of possible slopes. Next, the slope from the first point to the current point is evaluated. If this slope is within the range of acceptable slope values, the process begins again on the next slope. However, if the slope is outside the accepted range, the point immediately preceding the current point is stored to the compressed vector and set as the new "first" point in the curve and the slope values are restored to positive and negative infinity before the process proceeds.

Once compressed, data could be stored on a desktop computer and plotted in a relatively small amount of time (Figure 1). This, in turn, opened up the option of creating software that allows users to interact with the data in close to real time.

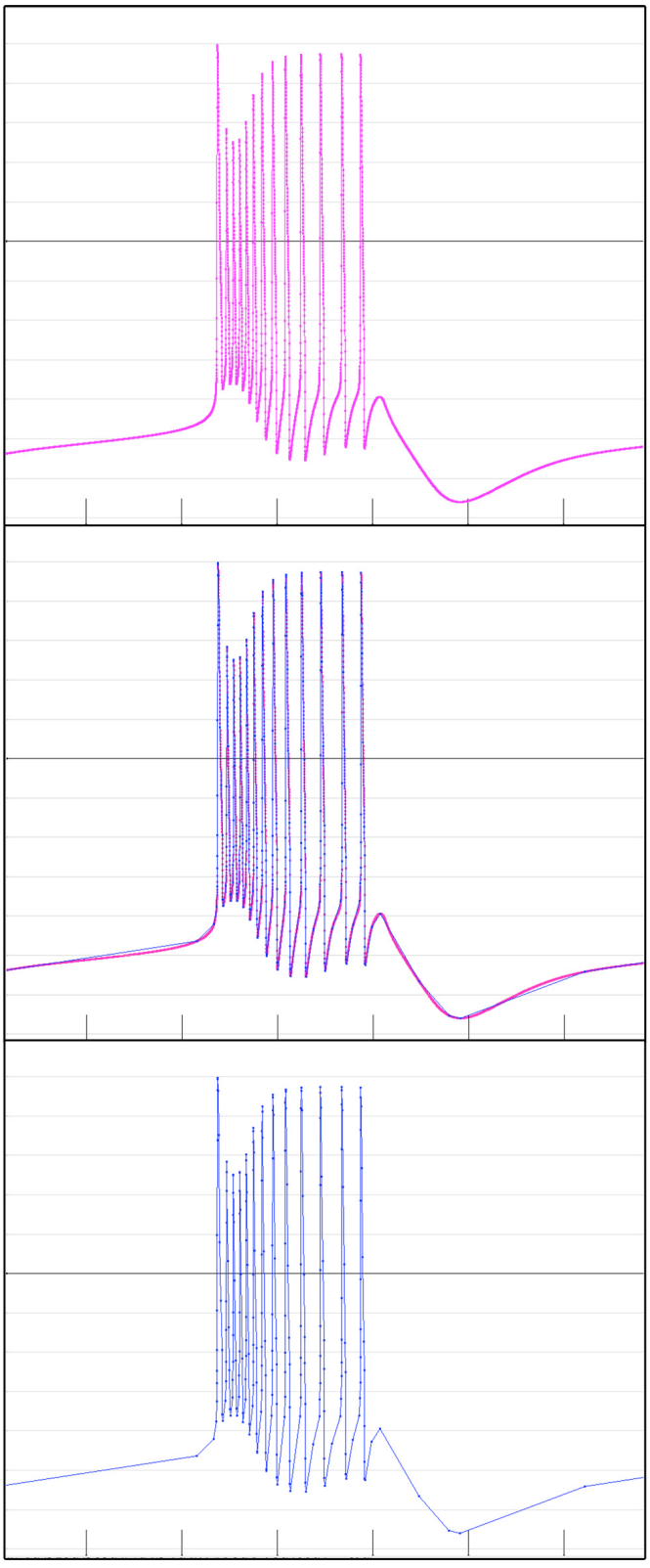


Figure 1- The pink dots represent stored data points prior to compression while the blue dots represent the compressed version. After compression, much of the shape of the data has been preserved while the amount of data needing to be stored is drastically lessened

.

Human Classification Tool

Once the data had been compressed into a more usable format, I wanted to make it more accessible to users so that they could easily look at many of the plots without having to plot each one individually. The first software tool, Neuron Tagger (Figure 2), allows a human to look through a pre-made data set sequentially. Functionalities in the Neuron Tagger include zooming in on sections of the plot, using a drop-down menu to select a different property of the current neuron model to view a plot for, re-simulating a plot to view the uncompressed version and plotting lines to show the period length of periodic behaviors. After looking at the behavior of a particular neuron, the user may select a neuron "type" using a drop-down menu of predefined class types. This allows the user to manually classify a set of neurons and store the classifications to display or construct a training corpus for machine learning purposes.

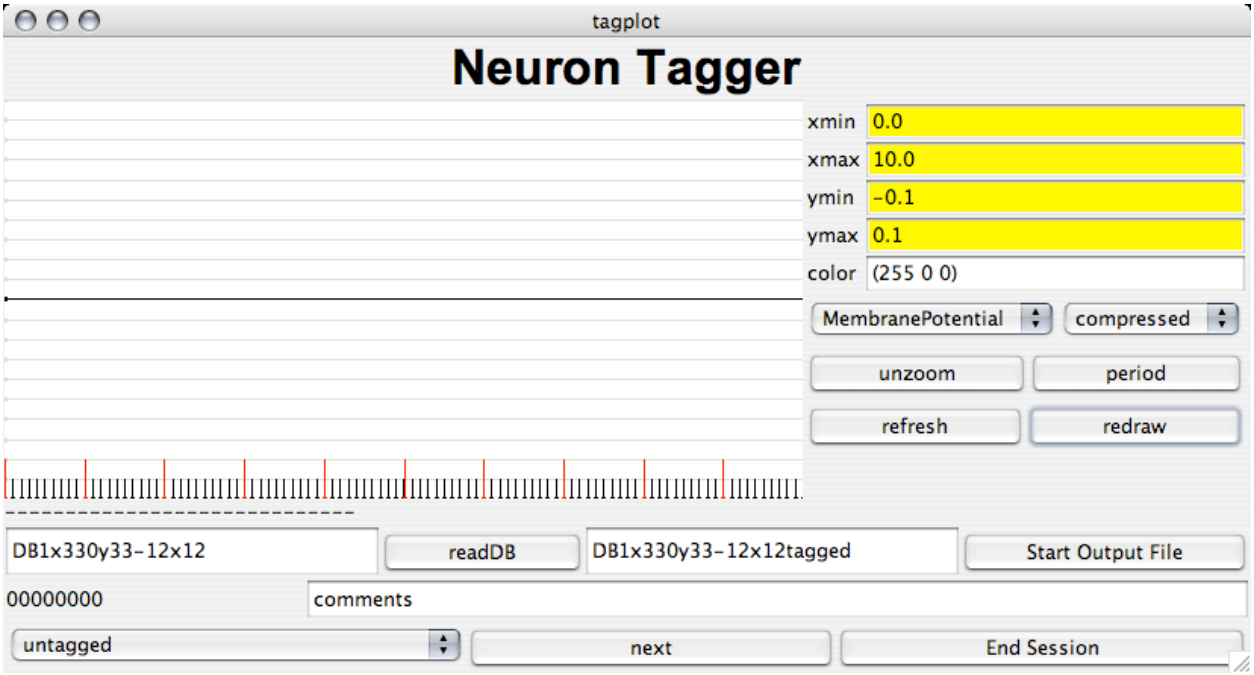


Figure 2: Users can use Neuron Tagger to look through the neuron models in a database individually and assign then to a class from a predetermined list of choices. This allows for limited exploration and training corpus construction.

This was useful for allowing us to look at the neurons that exist in a database and view the variety of behaviors present in the data. However, due to the sheer amount of data present, the depth of knowledge that could be gained from this approach was limited. After looking at more than a few plots individually, they begin to look similar and run together. To combat this problem, I needed to devise a way for the user to look at many plots at once.

Feature Extraction

To explore the data set more fully, the user should be able to look at multiple plots at once. However, plotting what could potentially be hundreds or thousands of plots and asking the user to compare them all at once was not a realistic option. Instead, I devised a simplified version of Langton *et al.'s* dimensional display which displays certain components or features of each plot simply enough so that they could be viewed and contrasted easily in large groups. To do this, first I needed a set of features that would be interesting in the context of this problem before I could define a way to present those features to the user. To this end, I developed several related feature extraction algorithms.

Many of the activity patterns found in the database are periodic, but do not necessarily have the same period length. Because of this, I decided that the period length might be a good feature to use in future auto-classification algorithms. The first feature extraction algorithm I wrote finds the period of cyclical spontaneous electrical activity. First, it finds the first local maximum (m_o). Next, it checks each following local maximum (m_i), and any local maxima found to be within a certain predefined threshold of m_o are stored as candidates. Using the points following mo sequentially as a predictor, the total error for the curve following each candidate is calculated. Once a predefined amount of the simulation has been scanned in this manner, the candidate with the lowest total error following m_o is selected as the beginning of the second period of the activity trace. The time of m_o is subtracted from the time of this point to determine the total period length of the trace (Figure 3).

Once the period of each trace could be calculated, it could then be used to determine the number of local maxima in each period. While calculating this, the number of maxima per period, it was trivial to calculate the number of local maxima per second as well. These features were the basis for preliminary exploration of the data set, which allowed users to view chunks of the data in order to look at trends of these features.

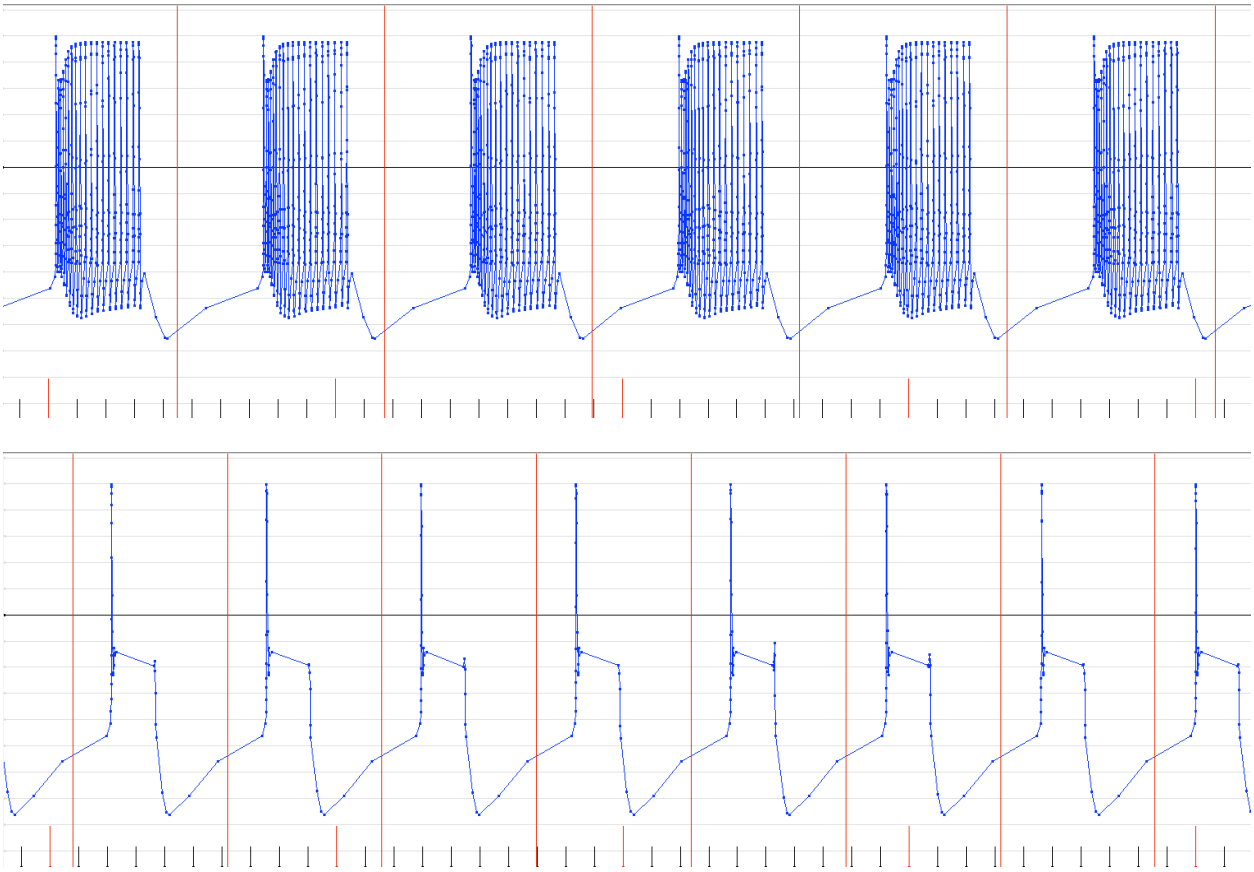


Figure 3: Two neuron behavior plots. The distance between red lines indicates the period length of each behavior as determined by the period finding algorithm described here.

Two-Dimensional Slice Viewer

Once several features had been designated, I decided that it would be useful to examine the changes of the features over the function of inputs rather than single instances. This would help to highlight patterns or trends as the inputs varied. I constructed many sets of models on a smaller scale, focused at predetermined areas of interest in the larger database so that we could look at a variety of information quickly. These slices were created by independently varying only two of the input parameters while the other six inputs were held constant.

Because this entailed working with a smaller subset of the database, we were able to simulate over a much higher frequency of inputs. The visualizer we developed (figure 4) allows the user to see the whole slice with each behavior model pictured as a colored box and the two varying inputs as the x and y axes. The boxes can be colored according to the features defined by our feature extractions to show how certain features change as the inputs are varied. The example pictured here shows the number of spikes per second, with darker shades of grey signifying a greater number of spikes per second. Users may then click on any box to have that particular model plotted in the top panel, and see more detailed information about the inputs of that plot. Users are also able to click a button to have the plot re-simulated and plotted without compression as well as click a button to have lines drawn at the period of the trace.

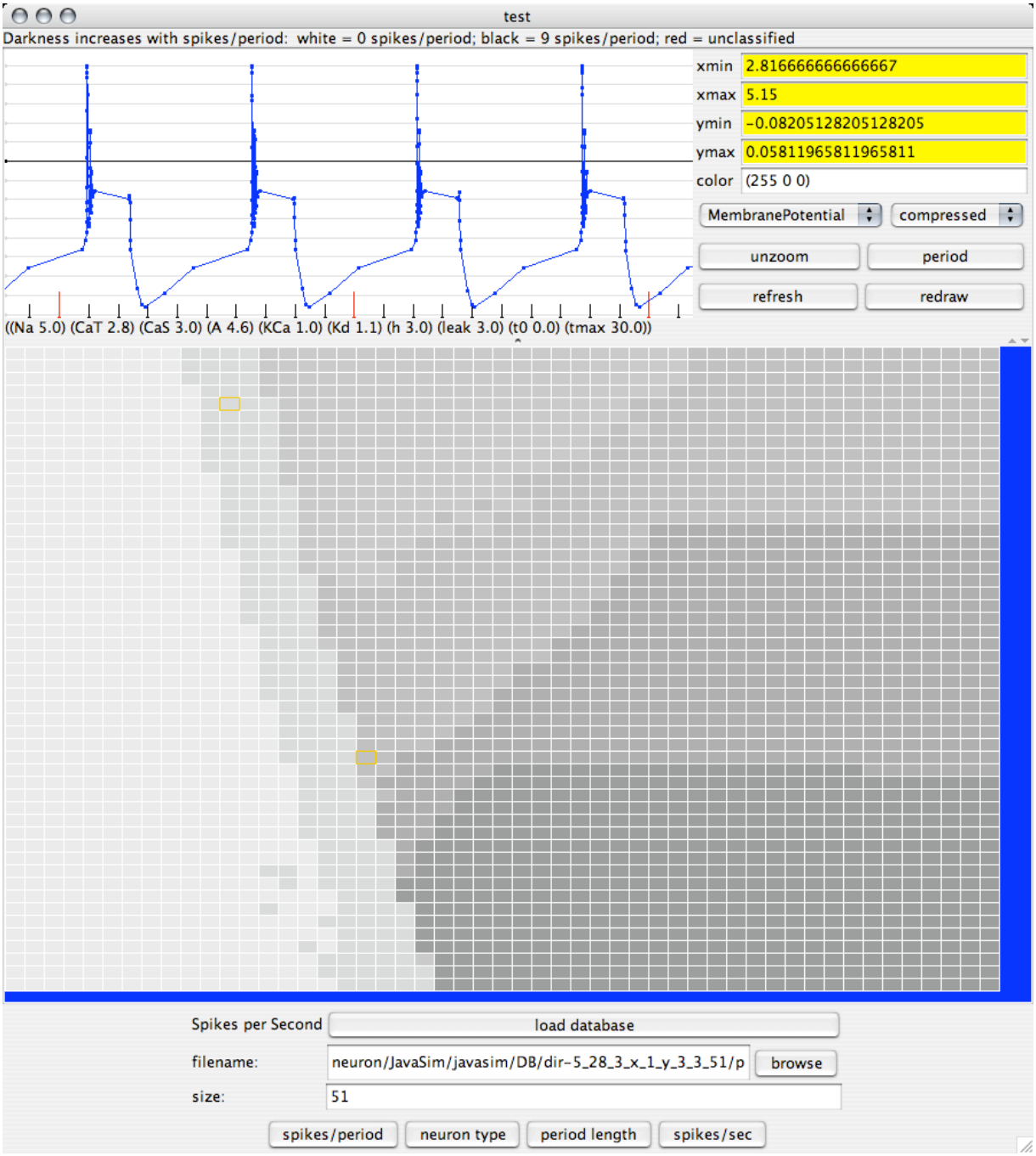


Figure 4: A screen shot of the two-dimensional slice viewer. The top panel is the same as the Neuron Tagger viewing panel. The bottom panel shows an entire two-dimensional database of neuron behaviors, with the x and y axes being the two varying conductances for the selected database. Each colored rectangle represents a single neuron model which can be plotted in the top panel by clicking on one of the rectangles. They are colored based on one of several attributes. The row of buttons across the bottom allows the user to select which attribute she would like to be used to color the neuron models.

Mining Activity Patterns Using K-Means Clustering

The database created by Prinz *et al.* includes vectors of all the minimum and maximum points for three periods of each voltage trace (or 1000 points of the non-periodic neurons). Therefore, a "single spike burster" voltage vector would contain six points: the three minimums and three maximums of the behavior pattern, while a more complex activity pattern might have many more stored points. These vectors store a very rough image of the voltage activity.

In my perception of the data, two activity patterns would be considered similar if the general shape of their activity is the same, regardless of duration. For example, if one voltage trace has the same pattern of voltage values as another, but is stretched over a longer period of time, they would still be considered to have the same behavior. So a single spike burster is always a single spike burster regardless of how far apart the spikes are spaced. Because I was interesting in finding similar voltage patterns regardless of the duration, I took out all of the time data, leaving each voltage description as a vector of the minimum and maximum voltage values it contains.

The next step I took was to discretize the voltage values. Rather than having an infinite number of possible voltage values for each minimum and maximum, I wanted to bin the data so that the minimum and maximum points could be represented as discrete values in order to make it easier to compare traces to each other.

Before deciding how to bin the data, I needed to show how equally distributed the minimum and maximum points were. Dividing the voltage space into a fixed number of equal bins could mean that some bins were almost empty while a great deal of information would be lost in bins where large percentages of the data were indiscriminately lumped together. To determine the distribution of the data, I created a histogram of the voltage minimum and maximum values.

Taking a random sampling of 0.1% of the voltage vectors, I then divided the voltage space between -0.1 and 0.1 into 20000 bins and summed the number of minimum and maximum points that fell in each bin. Plotting this data as a histogram proves that the minimum and maximum points are not evenly distributed in voltage space. In particular, there are clear peaks at about -0.02 and 0.045 (see Figure 5).

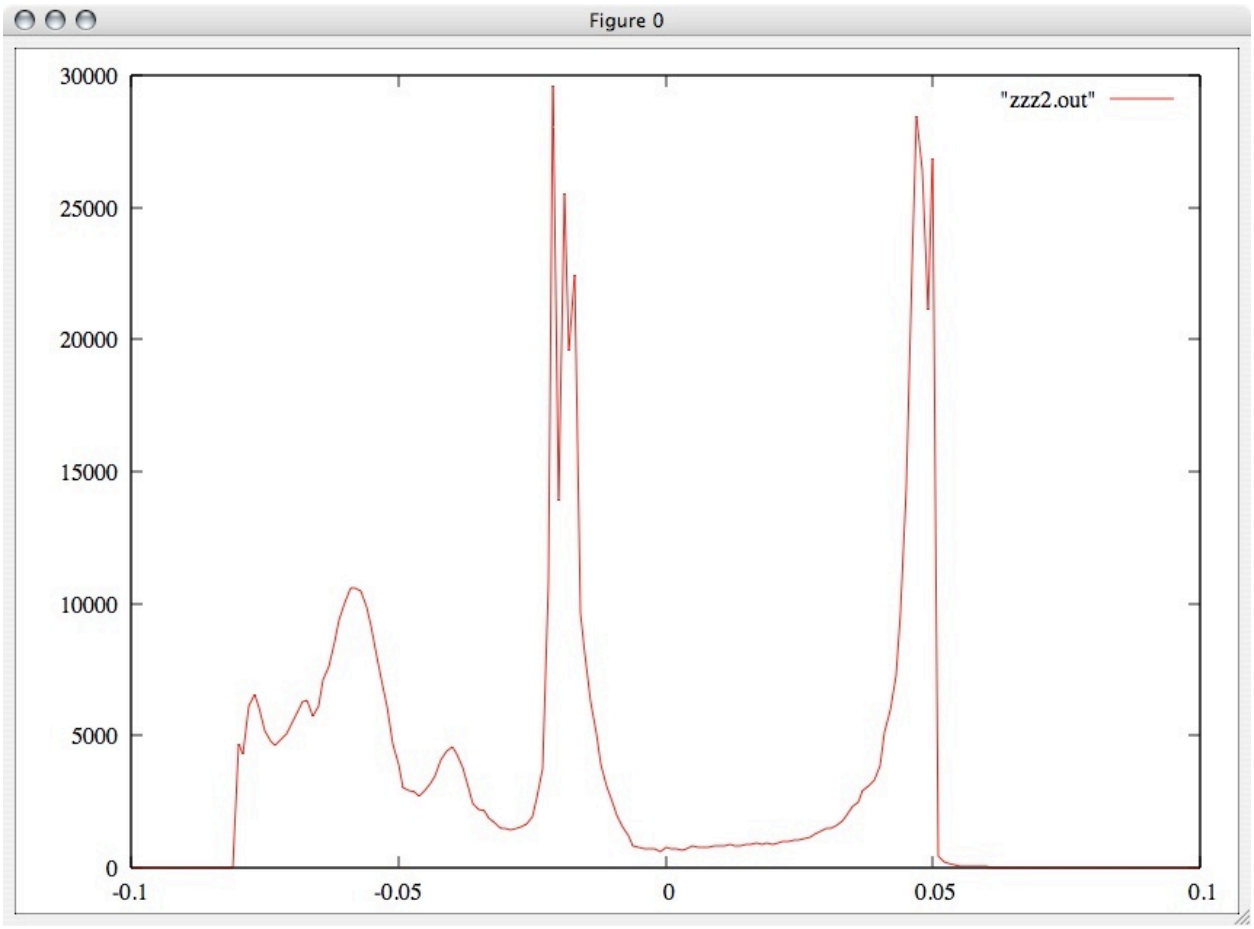


Figure 5 - In this graph, the x axis is voltage and the y axis is the number of points found at each voltage. Note that the data is not equally distributed throughout the space, but instead has several distinct levels.

Because the values were not equally distributed throughout the space, it made more sense to divide the data into bins which reflected the distribution of the data. Since there appear to be about five or six distinctive peaks in the voltage histogram, the data can most likely be divided into five or six bins. Rather than estimating the boundaries for these bins, I used k-means clustering (See Box 1) to determine one dimensional clusters, or bins, of voltage values.

K-means was attempted with both five and six cluster centers because they seemed to visually match the distribution in the histogram the best (Figure 6). After clustering the data using the k-means function in R, each point was labeled with the cluster it had been assigned to. I used a brute force method to find the highest and lowest value in each cluster, which defined the boundaries (Table 1). The clusters have been re-ordered in voltage space to illustrate the boundaries, which clearly shows that this method has effectively determined the edges of the bins with very little overlap.

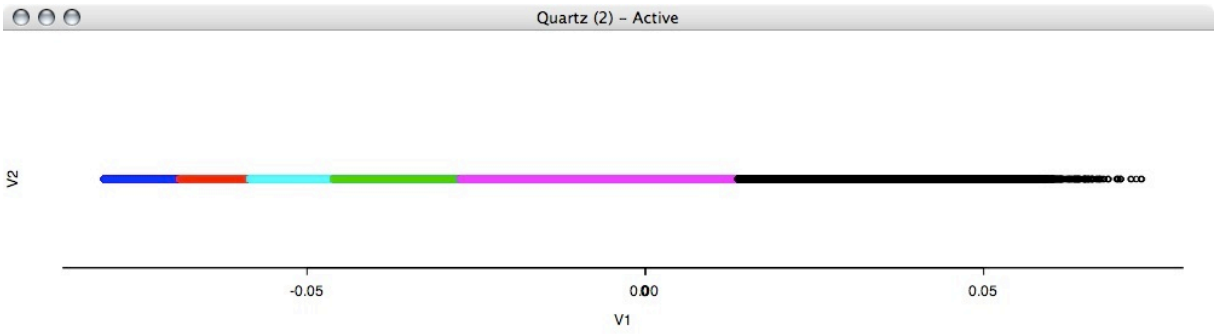
Box 1 - K-means Clustering: The reference manual for the R language and environment for statistical computing and graphics describes K-means Clustering as follows, "The data given by x is clustered by the k-means method, which aims to partition the points into k groups such that the sum of squares from points to the assigned cluster centres is minimized. At the minimum, all cluster centres are at the mean of their Voronoi sets (the set of data points which are nearest to the cluster centre)" (R Foundation, 2003). K-means is considered a staple of clustering methods because it is simple to implement and works well (Duda *et al,*, 2001).

In the following pseudocode example of a K-means implementation, n is the number of samples to classify and c is the desired number of clusters. Traditionally, c samples randomly selected from the data set are chosen as the initial cluster centers (Duda *et al,*, 2001).

K-means clustering algorithm (Duda *et al.*, 2001):

```
begin initialize n, c, μ1, μ2,..., μc
     do classify n samples according to nearest μi
          recompute μi
     until no change in μi
     return μ1, μ2,..., μc
end
```

6a



6b

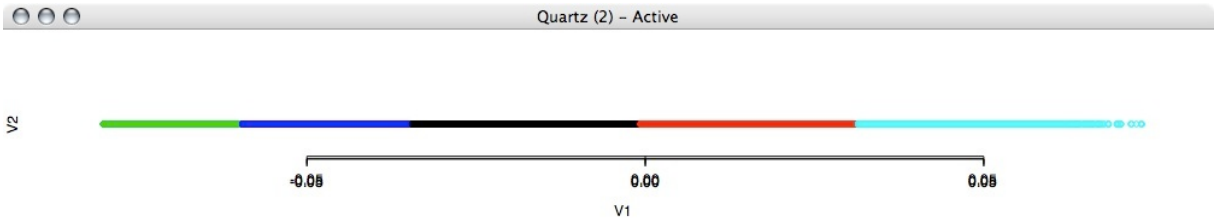


Figure 6 - The voltage values are plotted on the x axis, and colored based on the cluster they have been assigned to. 6a shows the values divided into six clusters while 6b shows five clusters.

Cluster ID	Lower Bound	Upper Bound
3	-0.079999	-0.059507
5	-0.059507	-0.034402
1	-0.034400	-0.000676
2	-0.000672	0.031483
4	0.031485	0.073395

Table 1 - The edges of each cluster in the five-center k-means clustering of the voltage levels of minimum and maximum points in the dataset were found using a brute force method.

Once the voltage space had been divided into bins, I used these values to write a java method that steps through the entire database and discritizes all of the voltage vectors (Table 2). The infinitely variable voltage points were each replaced with the numeric label of the bin their value fell into. This meant that each voltage point could only be one of five values. In the example below, the file containing all of the minimum and maximum points for a single voltage trace was loaded from a file, the time values (the left column) were ignored while the voltage values were checked to see which bin they fell into. Each new discrete voltage value was placed in a vector, and at the end of the trace the model ID number and the new discrete vector were written to a new file containing all of the newly discretized voltage vectors.

However, this procedure still leaves the data difficult to compare. Because each of the vectors holds three periods worth of points, and the periods could have any number of points in them, the data is left as a set of vectors that vary widely in length from six points to over a thousand. This makes it difficult to compare the vectors directly because even very similar voltage traces could have vastly different representations in this format. To solve this problem, each vector is further reduced to a vector of percentage values. Each model is assigned a vector of length equal to the number of bins the voltage space has been divided into (in this case, five). The percentage of points which fall into each bin is calculated and stored in the correct point in the vector. Using this strategy, each trace is described by 5 numbers, each on the same scale, so they can be easily compared to each other (Table 2).

996018

0.000000e+00	4.974533e-02
6.300000e-03	-1.097534e-02
1.110000e-02	-9.882242e-03
1.026000e-01	-7.579889e-02
7.241000e-01	4.974689e-02
7.304000e-01	-1.097532e-02
7.352000e-01	-9.882236e-03
8.267000e-01	-7.579889e-02
1.448200e+00	4.974609e-02
1.454500e+00	-1.097523e-02
1.459300e+00	-9.882241e-03
1.550800e+00	-7.579889e-02

becomes:

996018 4 1 1 3 4 1 1 3 4 1 1 3

And then:

0.5	0.0	0.25	0.25	0.0

Table 2 - The data starts out as dynamic values representing the precise voltage level of each minimum or maximum point. Using the voltage bins found with k-means, each voltage value is translated into a relative bin number. Finally, the percentage of each simulation represented by each voltage bin is calculated. As a result, each voltage simulation is represented by a vector of five percentage values that can easily be compared.

Using k-means, the data was clustered using the new five-dimensional percentage vectors (Figure 7). Five, six and seven cluster centers were tested. For simplicity's sake, the pass with five centers was selected to initially test the process. Admittedly, this technique requires some hypothesizing about how many groups there should be because there is not a predefined number of clusters to look for. Because R returns a list of labels for the data, the same technique that was applied to find the bins for the voltage levels could be employed to find rough boundaries for the clusters. As a first pass, this technique was used to define 5-dimensional boxes that describe each of the clusters. This allowed for a quick way to look at the types of voltage trace contained in each cluster before the boundaries between the clusters were determined.

Using the results from both the five-center and six-center clustering, a brute force method was once again applied to assign a cluster to each of the voltage simulations in the data set. Using the rough boundaries described above, about 75% of the simulations were able to be assigned to one of the clusters. The remaining simulations were assigned a dummy value. Once each simulation was labeled with a cluster assignment, examples of each cluster could be read from the original database and plotted.

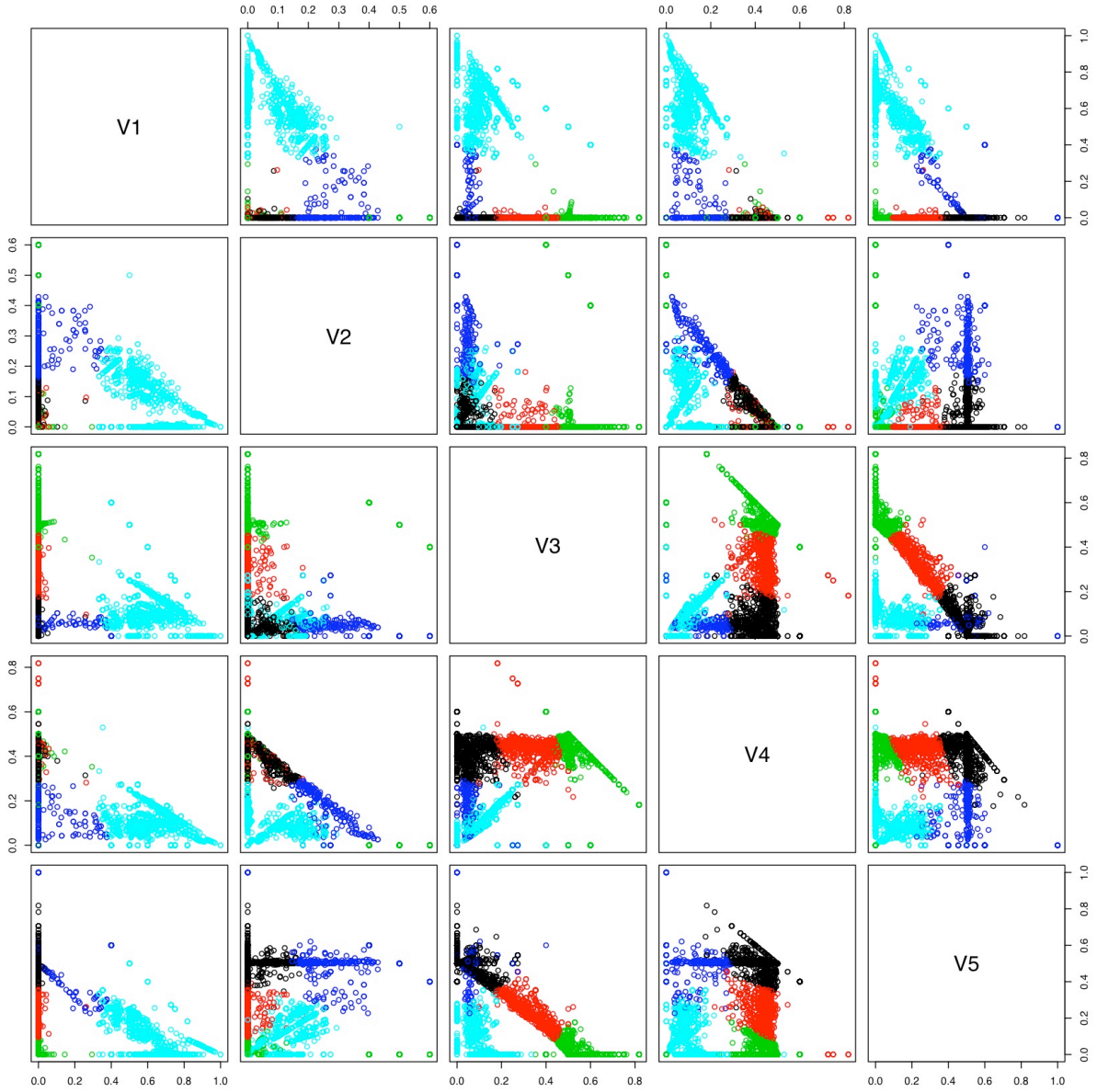


Figure 7 - A random sampling of voltage simulation vectors was converted to the five percentage representation described in Table 2, and then clustered according to k-means with five centers. The rows and columns in this figure represent each of the five bins in the percentage vector.

Once this rough classification method had been implemented, I simulated a new set of two-dimensional slice databases using the same technique described in the Software section above so that I could use the two-dimensional slice viewer to explore the classification results (Figure 8). In the example pictured, only three of the six behavior clusters are contained in the two-dimensional slice that was simulated (Figure 9). Only 79% of the models in this database were able to be classified, but the classification seems to be moderately accurate. Out of the 79% of classifiable neurons, only 6 of them (ie: 6% of the total neurons) were classified incorrectly. The incorrect classifications are located on the border between the tonic spikers and bursters, and all six of them are tonic spikers which were incorrectly classified as bursters. The activity patterns found in this database slice are one spike bursters, tonic spikers and bursting neurons.

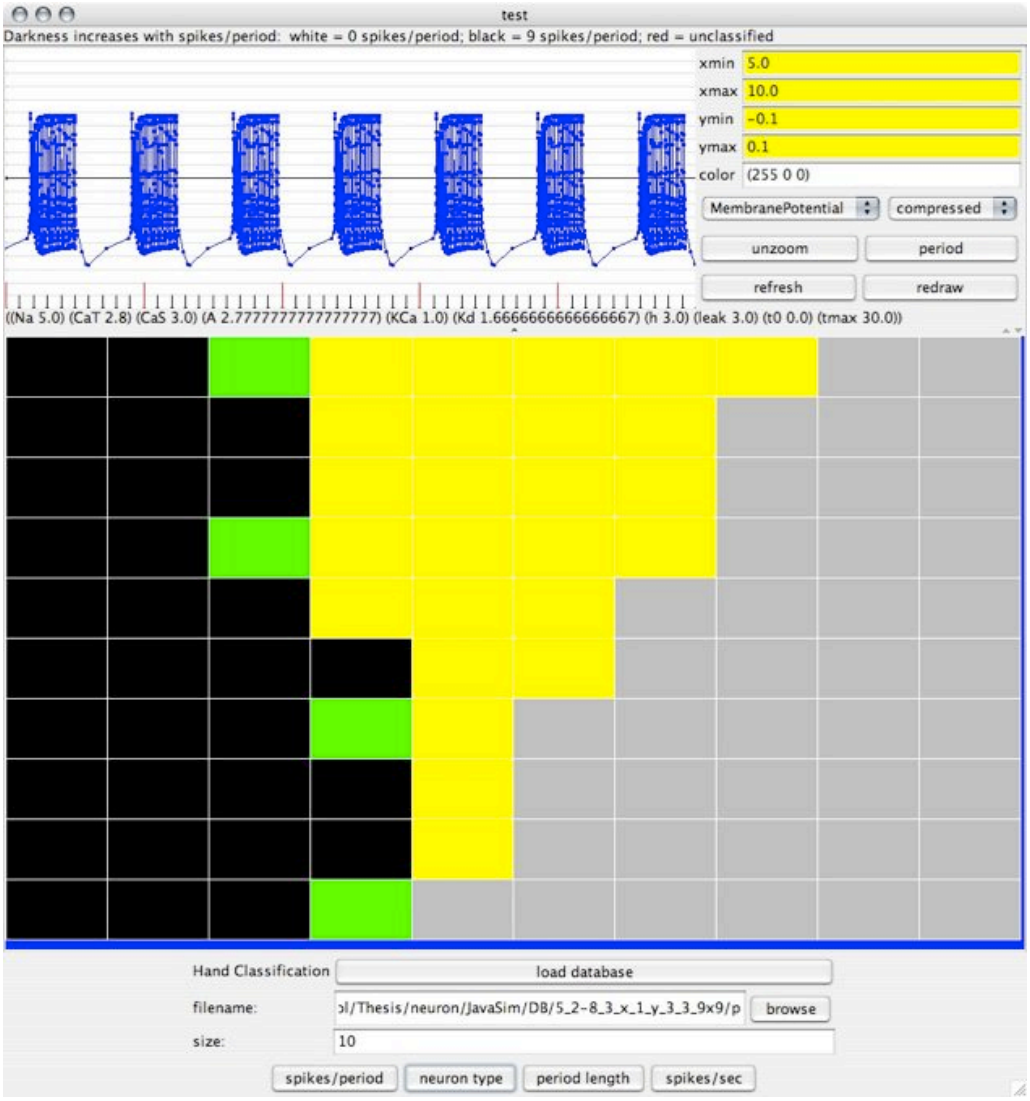
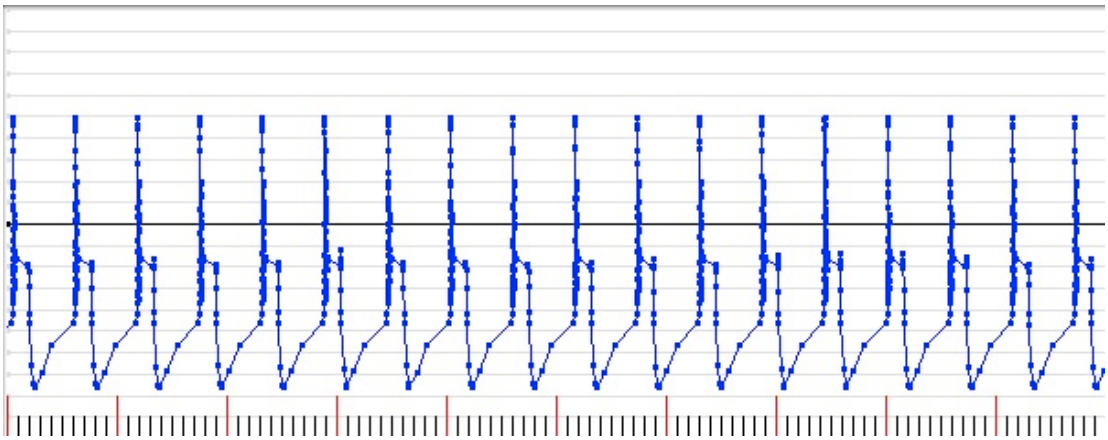
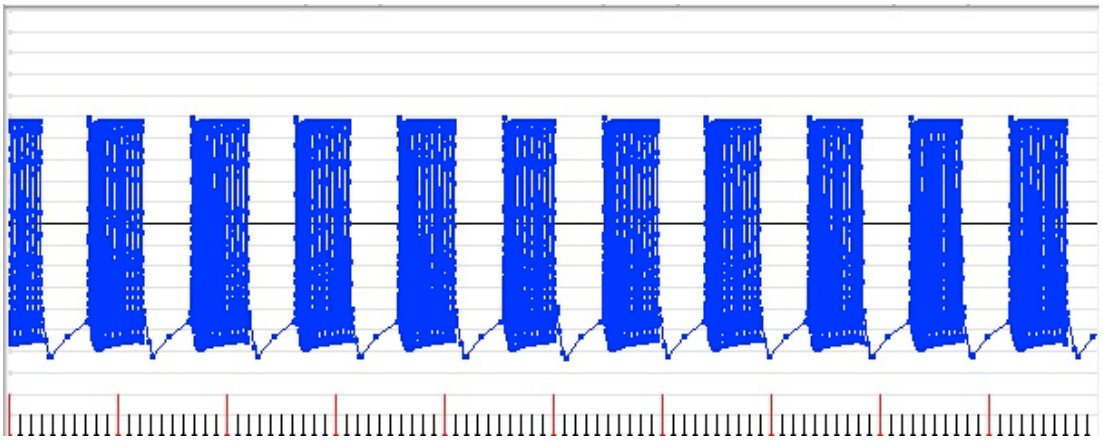


Figure 8 - A 2-dimensional database slice that has been classified using the results of the k-means clustering. Black squares indicate an unclassifiable neuron, green are one-spike bursters, yellow are bursting neurons and grey are tonic spikers.

9a)



9b)



9c)

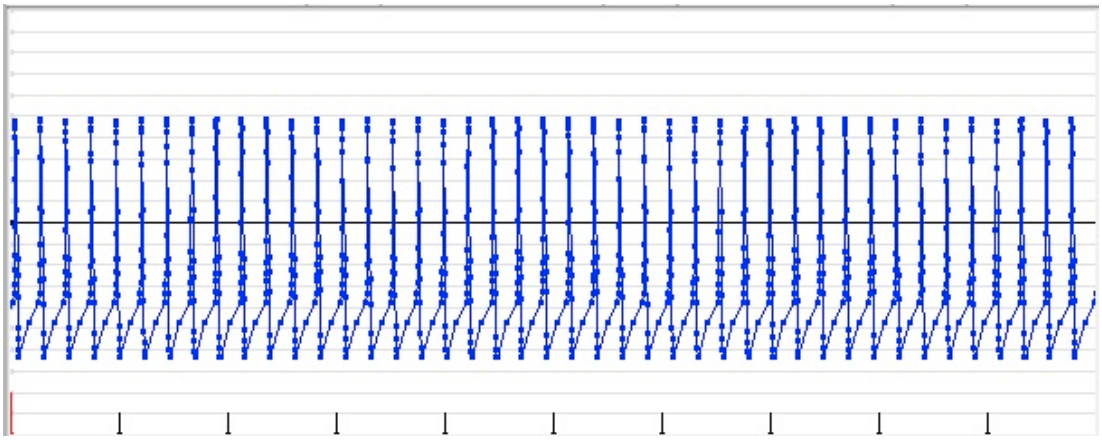


Figure 9 - Examples of activity patterns from the two-dimensional database shown in figure 8. 9a: The green colored neurons in this model all exhibit this behavior with a large first spike, several falling spikes, a wide shoulder and a final peak. 9b: All but six of the yellow colored neurons exhibit bursting behavior. 9c: All of the grey neurons exhibit tonic spiking behavior. The six yellow neurons which exhibit this behavior are located on the border between the grey and yellow regions.

Conclusion

Several approaches were successfully implemented to further the understanding of large databases of neuron behavior. Both the two-dimensional slice viewer and the activity pattern mining further the ultimate biological goal of exploring the relation between a neuron's conductances and its activity patterns.

The two-dimensional slice viewer is a good way to look at a simplified view of the effect of changing conductance values on different aspects of neuron behavior. Its functionality is enhanced by compression which allows for realtime interaction with the voltage plots as well as the ability to look at a much more highly sampled area than the original database and dimension stacking solution. The implementation of several feature extraction techniques facilitated the exploration of different aspects of behavior in conductance space.

Mining the activity patterns using clustering is still in its early stages, but looks like it could be a fruitful way to look at the data. The example shown in this project shows that with very little optimization, moderately accurate results can be found. The activity patterns were separated with a reasonable degree of success, and the two-dimensional viewer appears to continue to be a useful tool to examine this sort of data.

Future Work

The example shown in this project using extrema to look for similar voltage patterns is only one example of what could be mined for categorization of the data. Other attributes could be used to look for similarities between the different neurons contained in the database. Additionally, either fully implementing the k-means clustering rather than using the rough-box classification technique implemented in this paper or using other pattern classification techniques would probably increase the accuracy and efficacy of the final classification.

Another interesting problem would be to apply the results of the k-means clustering to live recorded data to see whether algorithms trained on simulated data would still be able to work on noisier recordings from life.

Finally, a long term goal for a project like this would be to create a predictor for recorded data. Because it is impossible to record more than a few conductances at once when working in vivo, it would be very useful to have a predictor that could classify the activity pattern and, based on that classification and the known inputs, predict what the other inputs likely were.

Acknowledgments

This project would have been impossible without the support and guidance of many other people. I would like to thank the following people for all the help I have received during the research and writing of my thesis.

My advisor, Timothy Hickey provided me with challenges and guided me past roadblocks that came up during the research and coding of this project.

Professor Pengyu Hong was instrumental in helping me to think through some of the machine learning and statistical analysis.

The rest of the NeuroVis group offered help and advice at all points along the way.

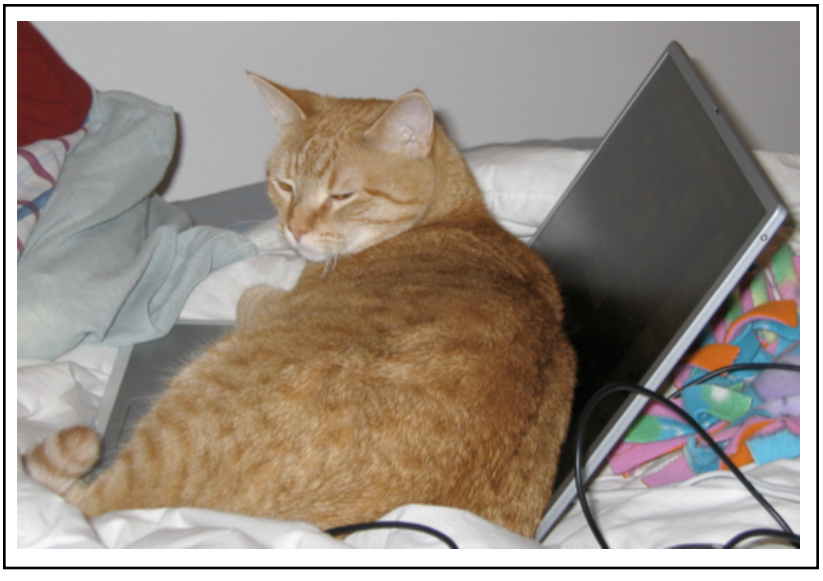
The IGERT Summer Research Program gave me the funding and opportunities necessary to devote several months of my summer to the beginnings of this project.

My parents have always supported me in my goals, taught me to love science and technology, left me to solve problems for myself when they knew I could but stepped in to help when they knew I needed it (even to the point of proof-reading my thesis the night before it was due).

Numerous other professors and friends have given me advice about and listened to me talk through ideas for this project. Thank you for gamely sitting through while I rambled.

dis-Acknowledgments

Chloe was the single largest hindrance to both my research and writing, with the steadfast convictions that she should always be the center of my attention and that my keyboard far surpassed any other seat in the house. She's lucky she's cute.



Bibliography

Duda, Richard O., Hart, Peter E., Stork, David G. Pattern Classification. New York: John Wiley & Sons, Inc., 2001.

Kandel, Eric R., Schwartz, James H., Jessell, Thomas M. Principles of Neuroscience. New York: McGraw-Hill, 2000.

Langton, John T., Wittenberg, David K., Hickey, Timothy J. "Leveraging Layout with Dimensional Stacking and Pixelization to Facilitate Feature Discovery and Directed Queries." Paris: The View Conference Proceedings, 2006.

Prinz, A.A., Billimoria, C.P, and Marder, E. (2003) An alternative to hand-tuning conductance-based models: construction and analysis of data bases of model neurons. J. Neurophysiol., 90:3998-4015.

R Development Core Team. "R: A Language and Environment for Statistical Computing and Graphics. - Reference Index." R Foundation for Statistical Computing, 2003.

Tools Used

Gnu Octave http://octave.sourceforge.net/ *Accessed May 12, 2006.*

GNU Octave Repository. http://octave.sourceforge.net/ *Accessed May 12, 2006*.

Sun Microsystems. **Java**. http://java.sun.com/ *Accessed May 12, 2006*.

Anderson, Ken, Hickey, Timothy J., Norvig, Peter. **JScheme** http://jscheme.sourceforge.net/jscheme/main.html *Accessed May 12, 2006*.

R Development Core Team. **The R Project for Statistical Computing.** http://www.r-project.org/ *Accessed May 12, 2006.*