Bioe 1586
Homework: Spike Sorting

When dozens of electrodes recordings are conducted simultaneously, for example using a multielectrode array in a brain-machine interface experiment, there isn't time to isolate individual neurons by hand. Automatic spike sorting algorithms are needed. In this assignment, you will build a spike sorter.

What exactly is spike sorting? Neurons fire action potentials, which can be recorded with an extracellular electrode. The shape of a particular neuron's action potential is consistent (albeit with some noisy variations). However, the electrode tip picks up signals from dozens of nearby cells. Often, in the signal recorded from a single electrode, you can see distinct waveform shapes. This indicates that more than one neuron is near the electrode. Since these cells could have very different functional properties, it is important to distinguish them, and analyze them separately. "Spike sorting" is identifying individual neurons in an electrode recording.

It is a hard problem for several reasons. Most importantly, action potentials from different neurons can look rather similar. They differ in the recording mostly based on the distance of the cell from the electrode tip (amplitude). Secondary differences may be present in the waveform shapes, due to which part of the neuron is near the tip of the electrode (cell body, dendrite, or axon), and the ion channel types present in the cell membrane.

When humans spike sort by eye, they focus on visually-distinct features like the peak height or width. The first automatic spike sorters tried to capture what people did by eye - they would measure specific features, and compare them. Better techniques were developed, and they are all pretty similar to the one you will implement:

(1) A voltage "threshold" is set for candidate action potentials. Every time the signal crosses that threshold, a snippet of it is saved to disk. (Threshold is typically set to 3x the root mean square amplitude of the signal.) Every time the voltage exceeds the threshold, we save a snippet of the waveform to disk as a candidate action potential. The snippet is saved from about 0.4 ms before the threshold crossing till about 1.6 ms after it.

(2) The snippet is sampled at 48 points. Each waveform snippet thus becomes a point in a 48-D space. Though it's hard to picture 48D space, you can imagine that since snippets arising from the same neuron will have similar waveforms, they will be near each other in the 48D space. The task of spike sorting becomes finding the tight cluster that correspond to all (and only) the snippets from the same neuron. If you're lucky, there is such a cluster. If you're very lucky, there's more than one, meaning you've isolated several neurons with the same electrode. Bonus data!

(3) Since you can't visualize clusters in the 48D space, we need to reduce the dimensionality of the original space. Principal Components Analysis (*princomp* in Matlab) is a good way to do this. What is PCA? It finds the dimensions of greatest variance in your original data set. The first principal component is the line through the original data set which captures the most variability in the original data. Think of it like

this: the value of a data point along the first PC is the most informative thing about it, since that is the scalar that marks the greatest difference among the data points; the second principal component captures the most variability in the data that remains after the first principal component has been removed, and so on, up to the dimensionality of the original data set. The key idea is that you can choose as many dimensions as you want, and get as close to the original data as you'd like, with a smaller amount of information. Or, looked at the other way, PCA can tell you how many independent dimensions actually exist in your data.

(4) We need an automatic method to identify the clusters. The k-means algorithm (*kmeans* in matlab) is a simple (but perfectly good) choice. You tell it how many clusters you think there are (that's k), and it will determine the best places for the centers of the clusters (that's the means) that allow each data point to be assigned to the nearest cluster. The algorithm is iterative: first, it chooses *k* region boundaries, then it assigns the data to the nearest region, then it adjusts the boundaries, till it converges (the boundaries adjust by less than epsilon from the previous iteration).

Note that there are more sophisticated algorithms - for example, the *EM algorithm* fits Gaussians instead of region boundaries. They are all iterative, and all involve two steps: one called "expectation" (assigning data to clusters according to the current model), the other "maximization" (adjusting the model so you can see if the next expectation step does better than the previous one).

The assignment takes you through the spike sorting process. Load the dataset *waveforms.mat*. It is the data recorded from one electrode during an experiment. Typically, recordings from 96 electrodes are made simultaneously, but the electrode tips are far enough apart that the recorded signals don't overlap, so the same algorithm is run on each electrode independently. The structure *data* contains two fields. The first, *wf*, are the snippets of neural activity around the time of a threshold crossing. The other field, *stamps*, is the time (in seconds) when the threshold crossing occurred.

Submit your code and the figures generated for each of these steps. Also provide a brief narrative answering the questions below:
(1) Plot 100 of the waveforms. Spread them out across the entire data session (that is, don't just plot the first or last 100.) Judging by eye, how many neurons do you think are present in this recording?

(2) Use principal components analysis to reduce the dimensionality of the waveforms. Show the waveforms in the principal components space by plotting their projections onto the first two PCs against each other. You can just use a 2D plot, or you can make a 3D plot, to get a clearer picture of the density of the clusters (the third dimension is the number of events. The *surface* function, or a color map, can do this). Now how many neurons would you say are present in the recording?

(3) Machine learning techniques are typically used to partition the principal components space into clusters or regions. Play with k, then decide once and for all how many

neurons you think are present here. Spike sorters have heuristics that allow them to guess pretty good k's, but currently they are not fully automatic, because the experimenter looks at the output, and then either merges or splits clusters (*i.e.* chooses a better k).

(4) Now plot 100 waveforms again, but this time color code them by their cluster assignment. This is the most visually informative part of the analysis. Did it work? That is, do waveforms of the same color appear more similar to each other than to waveforms of the other colors?

(5) The PCA space may seem a little esoteric. This question is to give some insight into what it does. Consider a unit vector along the first axis of the PC space. Rotate it back into the original space, and plot it as a waveform. What does it look like? Do this for the first three PCs. What intuition do they give you about what the PCA algorithm is doing?

(6) How many dimensions are present in the original data? The answer to this question is called an "eigenspectrum". It's a histogram with number of dimensions on the x axis, and the total variance accounted for by that many dimensions on the y axis. Obviously, with 48 dimensions, you can account for 100% of the variance. But, how many dimensions does it take to account for 90% of the variance? These eigenspectrum plots usually have an "elbow": a point at which adding more dimensions actually buys you very little explanatory power. That elbow is considered to reflect the natural dimensionality of the data. Where would you estimate the elbow to be in the eigenspectrum for these data?

(7) The final step in spike sorting is to assign the rasters to one neuron or another. For this you need the data.stamps field. These are the timestamps (in seconds) of when spikes occurred during the experiment. The entire experiment is over one hour long, so don't plot all the rasters. Instead choose a stretch of time about 10 seconds long and plot the rasters that occurred during that time. Of course, make sure you color-code the cells - that's the whole point of the assignment! Congratulations, your spike sorting is complete. Now, if we also had the behavioral data, we could plot separate tuning curves for each cell.

(8) A way to check whether your spike sorting worked is to plot the interspike interval histogram, which is the histogram of the time intervals between consecutive spikes. Knowing what you do about neurons' ion channel kinetics, you can predict a trait of the interspike interval histogram. What is it? Are your ISI histograms consistent with well-isolated single neurons?