NENS 230: Analysis Techniques for Neuroscience using MATLAB
Fall 2011

Assignment 6: T-tests, exponential fits, and PCA
DUE DATE: 9am, Monday, Nov 7th

Please submit this assignment to nens230@gmail.com, with the subject line "hw6". See list of deliverables at the end of the document. You should send these files in a .zip file attached to your email called hw6_SUID where SUID is your SuNET ID (e.g. hw6_djoshea.zip).

This third assignment is intended to demonstrate the use of simple statistical tests and annotating figures with significance markers, fitting curves to signals, and reducing the dimensionality of a dataset for exploratory data analysis.

Part I. Annotating the opsin frequency tracking comparison plot.

The first part of this assignment builds off of the function compareOpsinTracking.m you wrote in assignment 3. The purpose of that assignment was to analyze data recorded from several different neurons which were expressing a given opsin, and then to average their frequency tracking curves together. By repeating this process for several different opsins, we were able to see how one opsin performed relative to the other, in the sense of how well expressing neurons follow a periodic train of light pulses at various frequencies.

For this assignment, you will annotate these comparison curves with asterisks if the comparison at that frequency is significant. There are several approaches you might use to determine the significance. The simpler method is simply to run a t-test at each frequency, testing the null hypothesis that the number of spikes evoked by one opsin is the same as for the other opsin.

If you so desire, you can try correcting this for multiple comparisons, which means to adjust the threshold of each test to be a bit more stringent in order to ensure that the overall chance that *any* of the tests falsely rejects the null hypothesis remains at 5%. An easy way to do this is known as the Bonferroni correction[1], in which you simply make each test more stringent by dividing the required p value by the number of comparisons. So to ensure an overall false-positive rate of 5%, with 7 frequency comparisons, you would make the effective p-value 0.05 divided by 7. An alternative

[1] http://en.wikipedia.org/wiki/Bonferroni_correction

approach might be to run a 2-factor ANOVA, where the number of spikes evoked is a function of two factors, light pulse frequency and the expressing opsin. A main effect of the opsin factor would tell you that the curves are different overall, and if you wanted to drill down into which frequencies were different, you could use multcompare to do post-testing between pairs of frequency x opsin groups. But for this assignment, all you need to do is a two-sample, two-tailed t-test, where the two groups are spikes evoked at the current frequency for cells with opsin 1 versus for cells with opsin 2.

Start with the your solution to Assignment 3, specifically the compareOpsinTracking.m function. You'll need to modify this in some way to keep track of the nSpikesEvokedThisOpsin matrices for each opsin, rather than simply take the mean and std and then move on to the next opsin. Storing this inside opsinTracking(iOpsin) is a great way to to do this. Then once you've analyzed both the opsins, you should run ttest2's at each frequency. ttest2 can actually run a separate two-sample t-test the data in each column if you call it correctly (see help ttest2). You shouldn't need to loop over frequencies, but it's fine if you find the more explicit for loop approach preferable.

Once you have a p value for each frequency, after you plot the errorbar curves for both opsins, place a single asterisk above the frequencies which satisfy $p < 0.05$, and a double asterisk for $p < 0.01$. Use the text command to place text on the axes at a particular coordinate (see help text). The x-coordinate should be at the frequency, and the y-coordinate you should calculate. A reasonable y-coordinate would look something like: take the mean plus standard error for each opsin, take the maximum of these two values at that frequency (to figure out which one is higher on the plot) and then add a little bit of extra space for padding. This padding you could also compute by finding the range of y limits of the plot and multiplying this by 5% so that you're spacing scales automatically.

Tip: when calling text, specify 'FontSize', someFontSize, 'HorizontalAlignment', 'center' to make the font size big for the asterisks and center them horizontally over the x-coordinate.

Part II. Fitting inactivation kinetics.

In this code you will build off the skeleton code in fitInactivation.m to fit the inactivation kinetics of ChR2. When blue light illuminates a neuron expressing ChR2, many channels on the cell switch quickly from a closed, non-conducting state, to an open, conducting state. When these channels open, positive current flows into the

cell, and the cell's membrane voltage increases, leading to a spike. When the light turns off, the channels that remain in the open state will close or deactivate, ceasing the depolarizing current. However, ChR2, like many ion channels can also enter into an inactivated state. In this state, the ion channel does not conduct current, but it is unlikely to enter the open state. With time, these inactivated channels may return to the closed state or "deinactivate", and are then ready to be opened by light again. We are interested in measuring the kinetics of this inactivation. Since the rate at which open channels can inactivate is proportional the number of open channels, this inactivation process has an exponential time course.

One way to measure inactivation kinetics is to depolarize a neuron in voltage clamp. The voltage clamp is a method of recording intracellular currents inside a cell that uses feedback loop to inject whatever current is needed into the cell to "clamp" the cell at a constant voltage, e.g. -70 mV. The amount of current that the clamp needs to inject indicates the net current flowing in the opposite direction which the clamp is counteracting. Therefore, when ChR2 pushes positive current into the cell, the clamp injects negative current, and thus depolarizing positive currents deflect downwards on current vs. time plots in voltage clamp recordings.

You are provided with an abf file called photocurrent.abf. This is an intracellular voltage-clamp recording from a ChR2 expressing neuron. The first channel is the injected current, the second channel is the laser state (high = on, low = off). The laser is on for 1 second, and the inactivation is visible as the photocurrent has a large peak value but then decays to a much smaller steady-state value. You will fit an exponential curve from this peak to this steady state and return the time constant of this exponential fit.

This process will use the fit and fittype commands discussed in lecture today.
You can use exponentialFitDemo as a starting point, but in that script, the code assumes that the exponential decays to a steady state of zero, which is not true for the photocurrent. It also assumes that the exponential points upwards instead of downwards. You will need to account for this by using three parameters in the fit: amplitude, baseline, and tau, as explained in the skeleton code. You will also need to adjust the upper and lower bounds on the parameters accordingly so that they can take negative values. Start points, upper bounds, and lower bounds are specified to the fit function in alphabetical order of the parameter name.

See fitInactivation.m to get started. The basic abf loading code has been done for you to save time.

Tip: comment out the figure plotting code in countSpikesByLightPulseFrequency.m to make compareOpsinTracking.m run significantly faster.

Part III. PCA on Fly Motion Vision Dataset

You will be provided with a dataset which comes from Prof. Clandin's lab. Drosophila flies were placed in a device designed to assay their ability to process moving visual stimuli. In this device, the flies walk freely above a display which presents the moving gratings and patterns, and the behavioral responses of the flies (e.g. which way are they moving, which way are they oriented) were quantified at various points in time[2]. A small number of flies from a particular tube, all having the same genotype are assayed using this method, generating 703 measurements (dimensions) for that tube. This process was repeated for 215 tubes of flies, comprising 4 different genotypes which might have interesting effects on the fly motion vision system and thus lead to systematically different responses on this behavioral assay.

We'd like to use PCA to reduce the dimensionality of this high-dimensional dataset and to visualize the behavioral scores in 2 dimensions. Specifically, we'd like to see how the genotypes separate in this two dimensional space. We'll take data for all the tubes for all the measurement dimensions, run this through PCA, and then plot the scores of each tube on the first two principal components. We'll color each point on the basis of its genotype, enabling us to see whether the genotypes separate in this 2D space, something which is more difficult to see in the original full 703 dimensions.

See flyDataPCA.m to get started. You can keep this file as a script rather than a function.

---

[2] Katsov and Clandinin 2008. Motion Processing Streams in Drosophila Are Behaviorally Specialized. *Neuron*. SSN 0896-6273, 10.1016/j.neuron.2008.05.022. http://www.sciencedirect.com/science/article/pii/S0896627308004571

Deliverables:

Please submit the following

compareOpsinTracking.m
compare.png : the final opsin comparison curves with asterisks

fitInactivation.m
inactivation.png : the photocurrent trace annotated with your exponential fit and the title with the tau value in it. This is the "full trace" not the fragment, though please keep it zoomed in on the photocurrent (as the starter code does for you) so that I can see that the fit is correct.

flyDataPCA.m
flyPCA.png : the PCA plot for the fly data

Good luck!