**MSc Neuroscience**

**Computational Neuroscience II – Reverse correlation**

**Hilary 2020**

**Ben Willmore**

**benjamin.willmore@dpag.ox.ac.uk**

Just like last week, this analysis will guide you through a neurophysiological data analysis using MATLAB. This time, instead of simulated data, you will use real neural data recorded from ferret auditory cortex. And instead of just plotting the neural responses, you’ll characterise the neural responses using reverse correlation.

The first part of the exercise involves looking at and plotting the data, using the techniques you learned last week. It may well be helpful to refer to the handout from last time.

In the second part, you’ll estimate a spectro-temporal receptive field (STRF) – i.e. a linear model of your neuron – using reverse correlation and a separable kernel model.

**As last time, a bold line beginning Q is a question whose answer should be pasted into a Word document.**

## Part I: Visualising the stimulus

1. Download one of the files <http://willmore.eu/comp-neuro/spiketimes-revcorr.N.mat> where N is a number between 1 and 4. Don’t choose the same one as the person next to you. Load it into Matlab.

**Q1. What variables does the file contain? What kind of variables are they? How big are they?**

2. The variable stimulus\_ft represents a sound that was presented to an anaesthetised ferret during a neurophysiology experiment. Rather than the sound itself, stimulus\_ft is the Fourier Transform of the sound. It describes the amplitude of the sound, in decibels (dB) at each of 34 frequencies ranging from 500Hz (fairly low pitch) to 22.4kHz (too high for most humans to hear).

**Q2. How long is the stimulus, in samples?**

3. Each sample lasts 6.25ms. Make a vector, t, containing the times of the samples in ms. It should begin at zero, have a spacing of 6.25, and have the same length as the stimulus. The easiest way to do this is to first work out the time of the last sample:

>> last\_sample\_time = 6.25 \* (size(stimulus\_ft, 2)-1);

Then make a vector, t, containing values from 0, in steps of 6.25 up to last\_sample\_time. [Refer to Part I from last time if you need a refresher.]

4. Make another vector, f, describing the frequencies in the stimulus:

>> f = logspace(log10(500), log10(22400), 34)

**Q3. What does logspace do?**

5. Now, you have all the data you need to visualise the stimulus on the correct axes. You will need the ‘imagesc’ command to get this right– try ‘help imagesc’ and ‘help image’ (look for the line beginning IMAGE(X,Y,C)).

If you’ve got it right, the x-axis will be in milliseconds, and the y-axis in Hz. Label the axes clearly.

To make this plot clearer, use:

>> colormap(jet);

This will highlight the differences between different parts of the stimulus. You can also use:

>> axis xy;

This will put the lowest frequency at the bottom of the y-axis, and the highest frequency at the top.

6. Add a colour bar to the plot using ‘colorbar’. Add a title to the plot indicating what the colours represent (see step 2 above if you’ve forgotten).

**Q4. Put the resulting figure in your Word doc. Describe the stimulus in a couple of sentences: do you think it is a natural or synthetic sound? What aspect of it is random? What aspect is systematic?**

## Part II: Visualising the neuronal response

1. Have a look at the variable spiketimes, either by typing its name with no semicolon, or by looking at the variable browser in the top right-hand corner. It is a cell array. If you’ve forgotten about cell arrays, or didn’t reach the last part of last week’s exercise, read about them in last week’s handout, or google ‘matlab cell array’.

Each cell represents action potentials (spikes) recorded from a cortical neuron during one presentation of the stimulus.

**Q5. How many times was the stimulus repeated? How many spikes were recorded on each of the first two presentations? Why does spiketimes need to be a cell array, rather than a normal array?**

2. Plot a raster showing the spikes from the first repeat. First, set a variable, single\_trial, equal to the values in the first cell. You’ll need to use curly brackets {} to select the first cell.

Now, plot the values in single\_trial, just as you did in Part I, step 5 last time.

**Q6. Label the axes and paste into your Word doc.**

3. Now, write a program that plots a raster showing the spikes in all the cells of spiketimes. First, edit an m-file called plotraster.m:

>> edit plotraster

4. In the editor, make a ‘for’ loop where a variable, ii, counts up from 1 to length(spiketimes). [Refer to Part II, step 6 from last time.] Inside the for loop, set single\_trial equal to the ii-th element of spiketimes. Then plot single\_trial, just as you did above, except you’ll need to change the y-value from 1 to ii.

At the end of the for loop (just before the ‘end’), put ‘hold on’

**Q7. What does ‘hold on’ do? How do you turn it off again?**

5. Return to the Matlab command window, and run your program. [Refer to Part II, step 4 from last time.] Check that the plot looks right – does it have the right number of rows, etc.?

**Q8. Label the raster and put it into your Word doc. Can you interpret it? If not, what aspects of the neuronal response, or the experiment, are preventing you?**

7. As you found last time, peri-stimulus time histograms (PSTHs) are often easier to interpret than raster plots, so now let’s plot one for spiketimes. First, extract all the spikes from spiketimes into a single vector:

>> allspiketimes = [spiketimes{:}];

Plot a PSTH of allspiketimes, using the commands ‘histc’ and ‘bar’. [Refer to Part I, steps 6-8 from last time.] You can use the variable t as the edges of the bins of the PSTH. Remember to divide by the number of repeats, and the size of the time bins, to get a mean spike rate in spikes/sec.

**Q9. Label the PSTH and put it into your Word doc. Does the PSTH help you interpret the neuronal response? Compare the response to the stimulus. Can you say anything about the relationship between the stimulus and the response? If not, why not?**

## Part III: Reverse correlation

To work out what the relationship between the stimulus and response is, we’ll use reverse correlation. This comes in several forms, of which we’ll look at two – a Spike-Triggered Average (STA) and a separable kernel model.

The STA is the simplest form of reverse correlation. To find the STA, we first find the time of every action potential. We then take the stimulus immediately before that time. We add all these stimuli together, and divide by the number of spikes to get the mean stimulus that led up to a spike. This is the spike-triggered average.

1. To begin with, we’ll extract the stimulus that preceded the first spike. After that, we’ll make a program that extracts the stimuli that preceded all spikes, and averages them.

The first thing to do is make an array that will contain the STA. This array will have the same number of rows as the stimulus – one per frequency. It will have 15 columns, corresponding to 15 time steps. To start with, we’ll fill it with zeros. You’ll need something like:

>> sta = zeros(…) [\*]

where you need to fill in the missing parameters (‘…’) yourself.

2. Now, find the time of the first spike in allspiketimes. Set a variable, tm, equal to this time:

>> tm = …; [\*\*][!!]

3. Now, find the column of the stimulus that corresponds to this time:

>> col = ceil(tm/6.25); [\*\*]

4. Now, extract the 15 columns of the stimulus that were presented leading up to this time, and add them to the STA:

>> sta = sta + …; [\*\*]

5. Finally, display the part of the stimulus that you’ve cut out:

>> imagesc(…); [\*\*]

6. Of course, just one frame of the stimulus probably won’t look like much. Now you need to make a program to average the stimuli that preceded all spikes. To do this, first create a new m-file:

>> edit getsta.m

7. In your m-file, first add the line marked [\*] above. After it, make a loop that will step through every spike time contained in allspiketimes, and put all the lines marked [\*\*] inside the loop, in order. Finally, add the command ‘drawnow’ inside your loop. You should end up with something like:

sta = zeros(…);

for ii = 1:…

col = …

sta = sta + …

imagesc …

drawnow;

end

Last thing: in the line marked [!!], you’ll need to change the 1 to ii, so that each time round the loop, it uses a different spike time.

8. Of course, nothing will actually happen until you save the m-file and run it by typing ‘get\_sta’ in the command window. Try it now.

**Q10. While the program runs, you’ll see the figure changing, first rapidly, and later more slowly. What is happening? What does the figure look like to begin with? What does it end up as?**

If you get bored watching it, you can run it again more quickly by taking out the ‘imagesc’ and ‘drawnow’ lines. You can plot the result at the end using imagesc.

**Q11. Once the program has run completely, label the final figure and paste it into your Word doc. What does the STA tell you about the relationship between stimulus and response? What’s the frequency tuning of the neuron? What’s its response latency in ms?**

4. Download the file <http://willmore.eu/comp-neuro/separablekernel.m> . This is an m-file that will calculate a separable kernel model of the neural responses. Try it:

>> kernel= separablekernel(stimulus\_ft, histc(allspiketimes, t), 15);

Plot the result using imagesc.

**Q12. Put the labelled separable kernel plot into your word doc. How do the separable kernel and the STA compare? Which is more informative? Why?**

**That’s it!**