Computational Neuroscience (Autumn 2022)

Project III

Submission: Due Date 4th November 5 pm. Please email me only 1 m-file: A single MATLAB m-file code that loads the matfile shared with you and produces all the figures you present in the write up. Please submit a cogent write up (in my locker, labelled R-206) with figures appended at the end, as asked below; the writing should not exceed 2 pages reference to figures by numbers.

PLEASE NAME YOUR m-FILE AS "FIRSTNAME_ROLLNUMBER_PROJECT3.m".

This project is designed to implement topics learned in the second part of the course on encoding and decoding. You are provided data of spike times from 4 neurons (in an auditory area of the brain) in responses to a slowly varying stimulus, $s(t)$, which is the envelope (in dB scale) of a broadband noise carrier. The neurons are sensitive to particular features in the envelope. Hence, the noise carrier which is modulated by $s(t)$ is not provided as it is unimportant for the project. Your task would be find the envelope features important to or coded by each of the neurons. The same stimulus is presented and the data are, as though, collected simultaneously from the 4 neurons on 4 different electrodes. The 20 seconds long stimulus is repeated 50 times.

Format of the data:

In the matfile provided to you there are 2 variables, namely *All_Spike_Times* and *Stimulus*.

All_Spike_Times is a cell array/matrix of size 4 x 50. The spike times (in seconds) in response to the stimulus for the i^{th} neuron (i = 1, 2, 3 and 4) at the j^{th} (j = 1, 2, …, 50) repetition is given by the vector stored in:

All_Spike_Times { *i* , *j* }.

For example, if you do:

v=All_Spike_Times{1,1}; *v* is a vector of 344 elements giving the times of the 344 spikes that occurred in the first neuron in the first repetition of the stimulus.

$v(1)$ being the time of the first spike, which is 0.0419 seconds or 41.9 ms and $v(344)$ being the last spike occurring at 19.9076 seconds or 19907.6 ms.

The variable *Stimulus* is a vector of 20000 elements, with sampling rate of 1 kHz, it gives the 20 seconds long stimulus envelope $s(t)$ in units of dB.

Division of data for prediction purposes: For all the modelling work given below, consider first 15 seconds of data for estimating the model/receptive field and use the last 5 seconds of data for prediction purposes. Unless otherwise mentioned estimate everything at 1 kHz sampling rate that is at a temporal resolution of 1 millisecond.

EVERYONE SHOULD DO 1 through 6

1) Stimulus nature

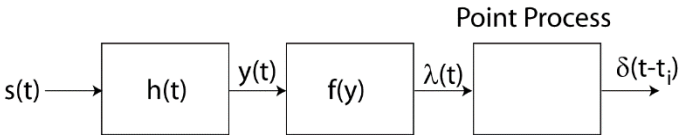
As you have learned to apply spike triggered average correctly the stimulus must be Gaussian distributed (white noise). Is it true? What is the autocorrelation function of the stimulus $R(\tau)$ for $\tau = -50 \text{ to } 50$ ms in steps of 1 ms (make necessary assumption about the stimulus)?

2) *PSTH* and mean firing rate

Obtain the PSTHs of the neurons (in spikes/sec) in response to the stimulus and plot them. You should be getting rates in 1 ms bins from the 50 repetitions of the stimulus. The firing rates should be of the order of 10s of spikes/sec (up to approximately 100). So you should have 4 plots, 1 each for each neuron, x-axis is time (0-20s) and y-axis is rate is spikes/sec showing the firing rate $r(t)$ – where t is at a temporal resolution of 1 ms. Keep the $r(t)$ of eah neuron for the last 5 seconds separately, which will used for prediction performance of the models you estimate.

3) *Poisson or non-Poisson*

For each neuron consider time bins of 10 ms, 20 ms, 50 ms, 100 ms, 200 ms and 500 ms. Get number spikes in each bin for every repetition. For example, in the 100 ms case for each repetition you will have 200 (20s/100ms) spike count values. Now with 50 repetitions you will get 50 samples of each spike count. From a scatter plot of mean and variance (obtained from 50 repeats) comment on the Poisson or non-Poisson nature of the spiking for each neuron at each time scale (10 ms to 500 ms).



4) Spike triggered average (STA) [and correction for non-Gaussianity?]

Now, as taught in class, obtain an estimate of the linear filter, $h(t)$ (left), that may represent each of the neurons, using the first 15 seconds of the data (spike times and stimulus). Assume a 100 ms length of the filter (*ie* assume spiking depends only on the previous 100 ms of the stimulus). Comment on the kind of filtering that such filters would be performing for each neuron. Based on your results in 1), make any correction(s) necessary.

5) Determining the output nonlinearity

In class, in the theoretical derivation of the filter (you determine in 4), the output static nonlinearity, function $f(\ .\ )$ above, was assumed to be arbitrary. In order to use the STA for prediction purposes, the output nonlinearity needs to be determined. We are not going to add the spike time generator, Point Process above; based on your results in 3) that may be done as needed. Our objective is to fit the PSTH (which is an estimate of $\lambda(t)$ that drives the point process) and also to predict $\lambda(t)$ for the last 5 seconds and compare with the observed PSTH of the last 5 seconds (for each of the neurons). By making a scatter plot of the estimated $y(t)$ and measured $\lambda(t)$ for each t try to estimate the function $f(\ .\ )$ – possibly a sigmoidal function. The other approach you can take is making a mean plot for $\lambda(t)$ for binned $y(t)$ values.

6) Prediction performance and pruning of filter parameters

A common problem in model estimation is over fitting. The more the number of parameters the easier it is to fit the model to the data, as ultimately the model tarts fitting the noise too. However, when you use this over fitted model to predict data that has not been used in fitting an over fitted model fails. (Generalization or) Prediction performance is thus a good way to test that. Use the models (for the 4 different neurons) to predict response rates or measured $\lambda(t)$ or the PSTH for the last 5 seconds. Use a scatter plot to depict the prediction performance, with actual measured values in x-axis and predicted values in the y-axis. Calculate r^2 or square of the correlation coefficient (between actual and predicted values) to quantify the prediction performance for each neural model. Next, we will throw out values in the estimated $h(t)$ that are not useful and replace them with 0. Your $h(t)$ has 100

elements (100 ms long). Start with the least magnitude value in $h(t)$. Replace it with 0. Now calculate r^2 (predication performance) with the new $h(t)$. If prediction improves or does not change significantly replace that element of $h(t)$ with 0. Go to the next smallest magnitude value and repeat the process, until you do not see (significant) fall in prediction performance. For each neuron you should have a plot of prediction performance versus parameter number (1 through maximum 100). This way you may be able to prune the filter parameters and get better models of each neuron. Plot the final filters and comment on their filtering properties by taking its Fourier Transform (use magnitude of fft in MATLAB).

DO EITHER A or B

A) Maximally informative dimensions (1 vector case and 2 vectors case)
The title above is self-explanatory. Based on your reading, start with the STA above as the starting vector (v). Project successive 100 stimulus segments (1 ms steps) onto v and get the projection x. Now use the algorithm to update your vector until you converge to a maximum. Test if you are stuck at a local optimum as described in your reading material. Once finished, again estimate the output nonlinearity. Use this for prediction as described and prune the parameters. For some of 4 neurons you may get a good prediction performance but not for others. For the other cases use two vectors starting the first of the two vectors at the result of the first step and the other as a vector orthogonal to the first. You may take hints from the reading material about how to start the second vector. In this case the output nonlinearity is a 2-dimensional function – you can try fitting $f(\ .\ ,\ .\ )$ a product of 2 sigmoidal functions like
$f(x,y) = \frac{A}{(1+e^{-a(x-b)})(1+e^{-c(y-d)})}$, where x and y are the projections on the 2 vectors.
Does the 2 vectors method improve your prediction performance? Which neurons?

OR

B) Discrimination based on Victor and Purpura (VP) Spike Distance Metric (SDM) – optimal timescales of decoding
For the discussion in class on discrimination we used different stimuli, with different identities. In this case we have a single long stimulus. We will artificially create different stimuli by randomly choosing segments.
Randomly choose 8 non-overlapping 100 ms segments from the 20 seconds of responses collected. The corresponding 8 segments in the stimulus are your 8 different stimuli. So you have 50 repetitions of each stimulus and 50 spike trains in response to each stimulus. Go through the procedure of creating the confusion matrix (size 8 x 8) based on the response spike trains and the VP SDM for different values of q – 0, 0.001, 0.01, 0.1, 1, 10, and 100. Get mutual information (MI) as a function of q – let us call it $MI_1(q)$. Repeat the process 100 times for new sets of 8 non-overlapping segments (these may overlap with the previous set of segments but not among these 8). Obtain $MI_j(q)$, where j goes from 1 through 100. Make a plot of the mean MI(q) from the 100 cases with 90% confidence intervals. Comment on the optimal timescales ($1/q$) at which MI peaks, for the different neurons. How does it relate to the STA, especially in case the STA has good prediction performance?