PROBLEM SET 2

Neural coding in linear and linear-nonlinear models

1 NEURAL CODING IN RETINAL GANGLION CELLS

1.1 BACKGROUND

Sensory systems do a remarkable job of extracting behaviorally relevant information from the environment and communicating that information to other brain regions. The field of neural coding explores this by asking: What information do neurons encode or represent? and How do biophysical mechanisms contribute to that encoding?

We can get a handle on these questions by formulating a quantitative model that characterizes the relationship between the external environment and neural activity. These *encoding models* allow us to distill our understanding of a sensory system, highlight key computational features, tackle normative questions (e.g. optimality), and uncover biophysical limits that have shaped the evolution of neural circuits.

For this problem, you will be analyzing data recorded from a salamander retinal ganglion cell (RGC) in response to a white noise visual stimulus. You will formulate a particular encoding model known as the linear-nonlinear (LN) model [2] that predicts the response of the cell to a given stimulus, and write code to learn the parameters of this model. You will also perform an analysis known as spike-triggered covariance (STC) analysis [4], to further characterize features that this particular ganglion cell is sensitive to.

For starters, download the required files (RGC data and the template script) from the course website. All of the code you need to write will be in the rgc_analysis.m script or the python notebook.

1.2 EXPERIMENT DETAILS

The data you will be exploring is stored in an hdf5 file called rgc_data.h5 (see the template script for how to load this data). The data consists of a 16.67 minute recording of an OFF ganglion cell from the salamander retina. The stimulus was flickering white noise bars, sampled at a frame rate of 100Hz. The stimulus array has dimensions (30x100000) corresponding to the pixel value of the 30 bars over 100000 frames. The time array contains the time of the stimulus presentation for each stimulus frame. Finally, the spike_times array contains the spike times of an isolated retinal ganglion cell (RGC) recorded in response to the stimulus.

1.3 Spike-triggered analysis

To analyze this cell, you will first need to compute the *spike-triggered ensemble* (STE). This is a matrix containing the stimulus that directly preceded a particular spike, for every recorded spike. Think of the STE as a cloud of points in the high-dimensional stimulus space. As we discussed in class, the mean of this set of points is known as the *spike-triggered average*. We will also characterize this point cloud by its *covariance*, this leads us to spike-triggered covariance (STC) analysis (see [4] for more information).

Part 1 Spike-triggered analysis

Loop over the set of spike times, and for each one, extract the stimulus that occurred right before that spike and store it in a matrix.

1. What is the dimensionality of the spatiotemporal filter? (This is the product of the number of spatial dimensions and the number of temporal samples in your filter).
2. First, you need to initialize the matrix that will store the spike-triggered ensemble (the variable ste in the script).
3. Then, fill out the code in the for loop that loops over the set of spike times, and for each spike, store the stimulus preceding that spike in the spike-triggered ensemble (STE). Note that you need to flatten the stimulus to be a vector (rather than a matrix) in order to store it in the STE.
4. Compute the Spike Trigerred Average (STA). Do this by taking the mean of the STE over the spikes dimension. This produces the average of all the spike-triggering stimuli, and gives us an approximate idea of the neuron's "preferred stimulus".
5. The provided visualization code will generate an image of the STA. Visualize the STA, add appropriate labels.
6. Describe what the spike-triggered average looks like. What does this tell you about what this ganglion cell encodes?

   For this part, turn in plots of the spike-triggered average. This plotting code has been written for you, but *make sure to add appropriate axes labels and titles to the figures.*

   Optional (in italics)

   1. Explore the spike-triggered covariance (STC)

a. *Perform PCA. The first step in any basic PCA implementation is to collect the data points we want to analyze. We've already done this by storing all spike-triggering stimuli in the STE matrix.*

b. *Next, compute the covariance matrix of this data. You can do this using the formula for covariance:* $Cov[x]= E\left[\left[(\bar{x} - x)(\bar{x} - x)^T\right]\right]$ *, or by using the cov function. Note that the transpose is on the second term here, because we used row vectors for the stimulus in class and column vectors here. Remember to make sure that the dimensions of the covariance matrix are correct–it should be a square matrix where one side has length given by the dimensionality of the filter (not the number of spikes).*

c. *Find the eigenvectors and eigenvalues of the covariance matrix. For this, use the eig function built into matlab or find the equivalent in numpy, which returns the eigenvectors and eigenvalues of a matrix. This is the essence of principal components analysis.*

d. *The provided visualization code will generate an image of the STA, the eigenspectrum, and the STC eigenvectors. Remember, each eigenvector of the STC matrix is a spatiotemporal feature that has been unrolled as a vector. Add appropriate labels to each of these plots.*

e. *PCA is commonly used to produce an approximate, lower-dimensional description of data by 1) expressing the data in the PCA basis, and 2) truncating the data to a small number of components. The eigenvalue spectrum (a list of all the eigenvalues) is an important tool in this process: recall that it tells us how much the data spreads out in each direction in stimulus space. Look at the eigenvalue spectrum plot generated by the code. If we have to discard a certain number of dimensions in stimulus space while keeping as much of the data's variability as possible, which ones should we discard? Discarding these, how would you use the eigenspectrum plot to decide how good or bad this approximation is?*

f. *How many eigenvalues in the eigenvalue spectrum are significant, i.e. above the noise floor? (you can estimate this by simple visual inspection of the eigenvalue spectrum, but a better approach would be to quantitatively estimate the noise distribution of eigenvalues by shuffling the data and repeating the procedure). What does this number tell you about the dimensionality of the subspace of stimuli that this cell is sensitive to?*

g. *Describe what the eigenvectors look like. How do they compare to the STA?*

h. *Computing the STC requires a lot of data. How can we be sure that we have computed enough to accurately estimate the STC eigenvalue spectrum? Can* *you describe in words (you don't have to do it) a simple way to test if we need more data (without recording more data)?*

1.4 Linear-nonlinear (LN) models

In the same analysis script, you will also estimate a nonlinearity–a function that describes the threshold and amount of amplification necessary to best predict the ganglion cell response given the stimulus and the STA. To do this, you will need to do the following.

Part 2 LN modeling

1. First, we need to compute the linear projection (or dot product) of the stimulus onto the linear filter (STA) that we computed earlier. Note that you need to flatten the stimulus to be a vector (rather than a matrix) in order to compute this dot product. In the loop over time, compute the projection of each stimulus slice onto the STA and store it in the variable u.
2. Now, you need to bin the spike times into an array that stores the number of spikes observed at a particular time, which we will call spike counts. Bin spike times using the bins given by the time variable.
3. Now we are ready to compute the nonlinearity of the LN model. Remember, the nonlinearity is the mean number of spikes (y-axis) vs. the projection of the stimulus onto the STA (x-axis). Loop over discretized values of the projection (the variable ub, the discretization allows us to average out noise), and for each value of ub, you need to average the spike counts array at times where the projection happens to lie within each particular bin.

For this part, turn in your plot of the estimated nonlinearity (if you use the sample code, you need to add axis labels and a title). Answer the following:

1. What shape does the nonlinearity have? What does this imply about how the neuron responds to multiple inputs (e.g. the combination two flashes as opposed to an individual flash)?
2. Estimate (just by eye) a threshold of the nonlinearity.
3. For this stimulus, what fraction of the time is this model neuron above threshold?
4. To fit this linear-nonlinear model, we used a stimulus with zero mean and fixed contrast. Natural stimuli, on the other hand, have many changes in mean luminance and contrast. What does this mean for our LN model? How can the LN model deal with such stimuli?

References

[1] Blaise Aguera y Arcas and Adrienne L Fairhall. What causes a neuron to spike? *Neural Computation*, 15(8):1789–1807, 2003.

[2] EJ Chichilnisky. A simple white noise analysis of neuronal light responses. *Network: Computation in Neural Systems*, 12(2):199–213, 2001.

[3] Aljadeff, J., Lansdell, B. J., Fairhall, A. L., & Kleinfeld, D. (2016). Analysis of neuronal spike trains, deconstructed. *Neuron*, *91*(2), 221-259.

[4] Odelia Schwartz, Jonathan W Pillow, Nicole C Rust, and Eero P Simoncelli. Spike-triggered neural characterization. *Journal of Vision*, 6(4):13–13, 2006.