15-386 Neural Computation

Problem Set 4
Bayesian decoding of neural activity

Assigned: March 17, 2023.
Due: March 31, 2023 at 11:59 p.m.
Credit: 12

Collaboration and submission guidelines: You should work alone on this assignment. You should submit a hardcopy of your report in class on the due day (we are going to have class that day), as well as a zip file containing all the codes and your report to the Blackbox assignment HW5 folder before noon on the due day. Running part1.m, part2.m, part3.m, part4.m and part5.m should reproduce your key results.

Overview of assignment In this assignment you will take data recorded from an actual experiment, courtesy of David Redish's lab, and decode the neural data to determine the behavioral variable being represented, in our case, locations in the environment.

First, unzip the data and code file from the assignment folder, and `load('hippocampus_data.mat')`.

You will be provided with the following data structures:

spike_matrix; spike_matrix is an NxT matrix, where N is the number of simultaneously recorded neurons and T is the number of time bins throughout the experiment. Each index (entry) in the matrix contains the number of spikes a particular neuron fired in a particular time window. The recording session was approximately 40 minutes long and spikes were binned into 10 ms time windows, so T is on the order of 40*60*100.

pos; pos is a Tx2 matrix describing the animal's location in the environment during each time window. The first column is x location and the second is y location. This is called the tracking data and each row of this matrix is referred to as a position sample. Do a quick plot to see what the animal's behavior was like during this experiment: `plot(pos(:,1),pos(:,2),.)`

behavior_time; behavior_time is a 2x2 matrix, with each row containing the start and stop time bins for decoding the animal's trajectory in Part 2.

replay_time; replay_time is 3x2 matrix, with each row containing the start and stop time bins for decoding replay events in Part 3.

Part 1: Plotting tuning curves (2 points).

In this part of the assignment you should learn to see how tuning curves (place fields in our application) for each cell is constructed. Your job then is simply to plot out the tuning curves (surfaces) in two different ways, and to study the cells' tuning curves.

The tuning curves will be stored in a matrix called 'tuning matrix', which will be an NxP matrix, where 'N' is the number of neurons and 'P' is the number of spatial bins or pixels. You will want to bin the tracking data into P spatial bins by constructing a BxB spatial grid where B*B is equal to P. (hint: choose B to be between 25 and 30. You can map `pos` data to a grid, e.g. by `[ gridloc = ceil(pos)+5)]`). To construct the tuning curve for each cell, we can identify the times when the animal occupied each pixel in the environment from the tracking data, and from those times determine how many spikes the neuron fired over that period (from spike_matrix). Each element in a tuning matrix should be in units of Hertz (spikes per second).

For example, if the animal occupied a particular pixel three times throughout the experiment and each time a particular neuron fired 2, 1, 2 spikes, respectively, then the firing rate of the neuron at the pixel will be (2+1+2)/(3*10 ms)=5/(3*0.01) Hertz. We need to compute this for every neuron and for every pixel in your grid to construct the tuning matrix.

At some spatial bins in the environment, there will be few or no position samples. You should choose a threshold below which you place a 'nan' in the tuning matrix, because you don't have enough data to be confident about the neuron's average firing rate at that location in the environment. For example, you can set a default value, say 0.0001, for the entries in a tuning matrix that correspond to the pixels the animal never visited.

The function for creating tuning matrix is provided. It did the computation above and return you a `tuning_matrix`.

```
function [tuning_matrix] = createTuningMatrix(spike_matrix, pos)
```

What to include in your writeup: Study and understand the tuning matrix and report what you observe about the tuning curves of the neurons. Plot out the tuning curves of five neurons, one at a time, support your observation. Include the 2D tuning curves of four cells with especially nice place fields in your report, including neuron number 8. This can be done from the information in tuning matrix. (hint: use imagesc, reshape, axis equal xy). Next to each of these tuning curves, create a separate plot showing the location of the animal each time the neuron spiked (in red) on top of all of the tracking data (in gray). Make sure that these corresponding plots agree with each other. Label each plot pair with the cell's number (row of spike_matrix). Some of these plots are also shown in the lecture slides.

Part 2: Decode paths taken by the animal during behavior (5 points)

Now that you have created tuning curves for each cell, which contain the average firing rate of each neuron at each location in the environment, we will use this information to decode the animal's location in the environment given a vector of spike counts ($\mathbf{n}$) observed during a small time window. This vector is N elements long, where each element is the number of spikes one neuron fired during the time window. As discussed in the lecture, we can use Bayes rule to go from $P(\mathbf{n} \mid \mathbf{x})$, the probability of the spike count vector given a location in the environment, to $P(\mathbf{x} \mid \mathbf{n})$, the likelihood of each location in the environment given the spike count vector. From $P(\mathbf{x} \mid \mathbf{n})$ we can determine the most likely location being represented by finding the location with the highest probability (the mode of the distribution). The equation for $P(\mathbf{n} \mid \mathbf{x})$ is shown below, which calculates (assuming a Poisson distribution) the likelihood that a particular spike count vector was observed at a particular location (with the mean firing rate of the location given by the tuning curve).

$$P(\mathbf{n} \mid \mathbf{x}) = \prod_{i=1}^{N} P(n_i \mid \mathbf{x}) = \prod_{i=1}^{N} \frac{(\tau f_i(\mathbf{x}))^{n_i}}{n_i!} \exp(-\tau f_i(\mathbf{x}))$$

where τ is the time window in which the spikes are counted, $f_i(\mathbf{x})$ is the average firing rate of neuron i at location $\mathbf{x}$ (it constitutes the tuning curve) and n_i is the number of spikes observed from neuron i over the time window τ.

We can then calculate $p(\mathbf{x} \mid \mathbf{n})$ by applying Bayes rule and reorganizing the equation slightly:

$$P(\mathbf{x} \mid \mathbf{n}) = C(\tau, \mathbf{n}) P(\mathbf{x}) (\prod_{i=1}^{N} f_i(\mathbf{x})^{n_i}) \exp(-\tau \sum_{i=1}^{N} f_i(\mathbf{x}))$$

where $C(\tau, n)$ is a normalization constant.

However, we want to compute $P(\mathbf{x} \mid \mathbf{n})$ completely independently from the animal's behavior, so we don't want to take into account $P(\mathbf{x})$. Additionally, we can change the product to a sum as shown in the equation below. The normalization constant is unnecessary since we only care about the location of the peak of the distribution. Therefore the equation simplifies to

$$P(\mathbf{x} \mid \mathbf{n}) = \exp(\sum_{i=1}^{N} \log(f_i(\mathbf{x})^{n_i})) \exp(-\tau \sum_{i=1}^{N} f_i(\mathbf{x}))$$

The first line of your decoding function should look like this:

```
function [decoded_locations]=bayesDecode(spike_matrix,tuning_matrix,behavior_time_indiv,τ
)
```

When you call this function you will send in rows of behavior time, which will be assigned to the variable behavior_time_indiv. *I will give you a version of this routine with several lines commented out, so all you need to do it to fill in a few lines.*

As described above, rows of behavior_time contain the start and end time of the decoding period. Say that behavior time(1,:) is [100 200]. This means that we are trying to decode the neural spike counts contained in spike matrix(:,100:200). As a reminder, each column of this matrix contains the spike counts of all neurons in a 10 ms time window. This is a very small time window, and when you perform decoding, you will want to look at a larger time window (on the order of hundreds of milliseconds). Thus, when decoding the location that the population of neurons are representing at time 100, you should look at a window of activity centered at time bin 100. For example, sum(spike matrix(:,80:120),2) will give you a spike count vector over 410 ms (τ = 0.41 s), centered at time 100. The output of the function, decoded locations, is a tx2 matrix where t is equal to behavior time(1,2) - behavior time(1,1) + 1, with each row containing the decoded location [x,y]centered at each time bin in the period defined by behavior time. The outline of your function bayesDecode may look like this:

```
...
window = floor(100*tau/2);
```

```
for i=behavior_time_indiv(1):behavior_time_indiv(2)
    n = sum(spike_matrix(:,i-window:i+window),2);
    ...
    px_given_n = ...
    [r,c] = coordinates that maximize px_given_n
    decoded_locations( i-behavior_time_indiv(1)+1, :) = ...
end
```

Now, we will determine the average error between the decoded locations and the animal's actual location. To do this, we can calculate the Euclidean distance between the animal's location and the decoded location at each point and time and take the mean. This can be done by the following commands:

```
range = behavior_time_indiv(1):behavior_time_indiv(2);
diff = pos(range,:)  - decoded_locations;
sqrerr = sum( diff.^2, 2 );
error = mean( sqrt(sqrerr) );
```

What to include in your writeup: Include your bayesDecode routine. Run your function bayesDecode using a few different values of τ and calculate the error for each run. For three of the runs, create a plot showing all the tracking data (in gray), the rat's actual path during the decoding time window (in blue), and the decoded locations (with red Xs, hint: plot(...,...,'rx')). Underneath each plot, include the mean error and the τ you used. Since there are two periods of time to decode over, you should have a total of 6 plots. Additionally, answer the following question. BTW, don't worry if it's not perfect. Real data is noisy!

Question 1: We can increase the amount of data we use for decoding by increasing τ . However, there is also a downside of doing this. Explain this tradeoff in a couple of sentences.

Question 2: By discounting P(x) in decoding the rat's position, we are effectively throwing away our knowledge about where the rat will likely be behaviorally. Now, use P(x), as deduced from the behavioral data, to do the decoding. Does this improve the decoding accuracy? How much? Under what task scenario might you actually want to incorporate P(x) in your decoding of the rat's position?

Question 3: How would throwing away certain percentage of your neurons randomly, adversely affect the performance of your decoder? Plot a graph that shows the decoding errors when 100 percent, 75 percent, 25 percent and 10 percent of the neurons in the samples are used. You should try 10 or more random subsets of neurons for each percentage to get an average error estimate. Include the routines for computing this as part of your part2.m routine submitted to CANVAS.

Part 3: Models of Neuronal Variability (2 points)

In Part 2, we assume the neuronal variability follows a Poisson distribution. How true is this assumption? Study Part 1 to see how tuning curve is computed. Design an analysis to assess whether the spike counts at each location for window duration of your choice follows the Poisson

distribution (try at least $\tau = 0.5$ and 1). Support your conclusions with evidence in terms of calculated numbers and graphs. This question is slightly open-ended in the sense that the answer could vary depending on your choice of spatial bin and time bin. So you have to report your design choices, experiments and observations to back up your observations and conclusions.

Part 4: Decode paths expressed mentally, while the animal is paused in the environment (2 points)

For this part, you will use the function you created in Part2 (bayesDecode). However, instead of using behavior_time, you will input rows of replay_time. Replays, as talked about in lecture, occur on a faster time scale than neural activity during behavior, thus you will need to use smaller values of τ (obviously have to be integer multiples of 0.01 ms). Play around with this parameter and see how it changes your output. In order to display the decoded trajectories and observe the affects of changing τ, use the provided code `plotTrajectory.m`. This will allow you to visualize the temporal structure of the trajectory on a 2D plot. An example part4.m is provided to help you do that.

What to include in your writeup: A single plot for each replay event, using the value of τ that produced the nicest results (visually). Make sure to label each plot with the following information: the number it was (row in replay time), and the τ you used. Additionally, briefly describe your observations with using different values of τ.

Part 5: Creative Exploration (1 point required) with up to 3 bonus points

In this assignment, we have just given you some basic tools to plot tuning curves and visualize the decoded trajectories and locations. With the spike data matrix and the position matrix, you should be able to do many interesting creative exploration. This creative exploration part is **required**, with 1 point. You should form at least one question of your own and go about analyzing the data to answer it. The question has to be related to this problem set and utilizes the neural data. Thus, it is a creative exploration on neural data analysis. As this assignment is relatively simple, this is an opportunity for you to do some in-depth creative exploration. Outstanding answers could earn up to 3 points in bonus credit.