Computational Neuroscience - Project 7
Neural Decoding I: Discrete Variables

Kadir Berat YILDIRIM

Jan 2023

Introduction

Neural decoding is a mathematical mapping from the brain activity to the outside world, which is the inverse of neural encoding which maps the outside world to the brain activity. By the use of neural decoding, we will be predicting the upcoming direction of movement from a population of neuronal signals recorded from motor areas of a macaque monkey.

We are going to be using 2 techniques; a population vector decoder and a maximum likelihood decoder, then we will be comparing these techniques.

The 'population vector' is a way to use cosine-tuning information from a population of neurons to decode movement direction.

For the population vector decoder, we assume each neuron contains information about its 'preferred' direction of movement and the magnitude of each neuron's response vector, which is the activity of the neuron during each trial, is the weight given to each neuron that will be the firing rate during the hold period. Then we will be taking the sum of all the response vectors from all neurons, as mathematically shown in equation 1, where n is the number of neurons, P is the population vector, ω_i is the weight, and C_i is a unit vector pointing in the ith neuron's preferred direction.

$$\vec{P} = \sum_{i=1}^{n} \omega_i \vec{C}_i \tag{1}$$

The maximum likelihood decoder is a more general decoding algorithm, where the assumption is that in an upcoming movement, neuron target firing rate will be corrupted by noise, so that for a given direction we will be observing a distribution of firing rates rather than a single firing rate. The goal is to find the form of this distribution - with its estimated parameters - and we will be assuming first a Gaussian, then a Poisson shape to characterize the firing rates. The probability of a set of n firing rates R for a given direction d can be mathematically expressed as in equation 2. However, since the natural logarithm is a monotonically increasing function, the choice of direction that maximizes the log-likelihood will also maximize the likelihood and for that reason, it is more convenient to pick the direction that maximizes the log-likelihood, which is shown in equation 3. This way we avoid taking products of very small numbers in MATLAB, which may become inaccurate.

$$P(R|d) = \prod_{i=1}^{n} P(r_i|d) \tag{2}$$

$$log[P(R|d)] = \sum_{i=1}^{n} log[P(R_i|d)] \tag{3}$$

Analysis

As the book states, in order to avoid overfitting, we need different data than the ones we use to train the model. To be able to do that, I have split the trial data into 2 with sizes 150 for training and 7 for testing. We want to use data from all neurons (stored in a variable called *unit*) and train and test how each neuron responded during trials data.

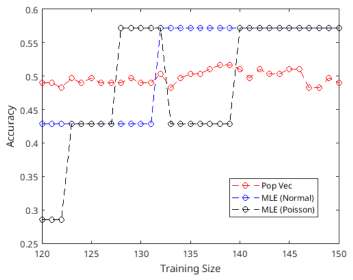


Figure 1: Accuracies (in percentages) for different train/test sizes for each model.

Population Vector Decoder

In order to create a population vector decoder, we are finding the preferred directions of all neurons using cosine fits to tuning curves. We are taking the maximum values of those fits done just as before in chapter 17. We save those parameters for each neuron and use those parameters for estimating the test datasets.

Maximum Likelihood Decoder

For the maxium likelihood decoder, we create the model using the distribution parameters; means and the standard deviations. As explained in the introduction, we estimate using the distribution parameters for the preferred direction of neurons. Again this is done for each neuron so that we have a guess of direction for every neuron in the dataset. Using the resulting model, we estimate for the test set again.

Comparison

Table 1 shows errors in percentages and mean squared errors for all models we have built in this project.

Table 1: Accuracies for models trained with *Chapter21_CenterOutTest* dataset with a train/test split of 150/7 trials.

-	PVD	MLE (Normal)	MLE (Poisson)
Percent	71.43	57.14	57.14
MSE	7.1429	7.7143	5.4286

From the table, for training set size of 150 trials, we can conclude that population vector decoder gives better results in predicting unknown data (test set trials) with extra 14% accuracy rate compared to MLE models. However, its MSE value is only lower than MLE model with normal distribution, suggesting that MLE model using Poisson assumption is better than population vector decoder (since a value closer to 0 is better for MSE results).

I have searched online and saw that sometimes these two accuracy methods do not agree. Absolute (percentage) and relative (MSE) metrics are measuring different aspects of the prediction. So one model is not better than the other in absolute sense. Which metric to value depends on the application, when the outcome range is wide and skewed, relative error measurements are better than absolute error

One small example for this topic is these 2 cases; real value was 99, prediction is 101 and real value was 5520, prediction is 5522. This would result in both cases, the absolute error of 2, but relative error in first case is much larger (2% - 2/101) than second case (0.035% 2/5520).

measurements. In that sense, if I assume a Poisson distributed neural activity, it would be better to look at MSE to decide which model is best representing my data, and that would be the maximum likelihood decoder using a Poisson assumption. It is probably no surprise that the normal distribution assumption on MLE decoder gives a higher MSE than the Poisson assumption one.

<– Conclusion

So to sum up, in percentages, the population vector decoder gives a better result than maximum likelihood decoders. This might be logical since we do not have a wide range of outcomes - our outcomes are 1 to 8 which are the directions. Although, MSE tells that maximum likelihood decoder using Poisson distribution is better than other models, and while I am not sure why this is the result, it may be because of the underlying physics of a neural activity is better represented with a Poisson distribution.

I have also tested change in accuracies for all models for train sizes 120 to 150 (used the remaining data for testing in all cases). Results are shown in figure 1. In smaller sample sizes, for reasons yet unknown to me, maximum likelihood estimate decoders work better than population vector decoder. On the contrary, for sample sizes above around 135, population vector decoder starts to perform better. These results may be better understood with a further analysis that is not done in the scope of this project.

Appendix

Overfitting the data

If we were to not split the data into train and test sets, the model would overfit, predicting all the data we have with a really good accuracy. In order to see that this is the case, I have tried training and testing with all 157 trials we have on our data, which resulted in a 98% accuracy for maximum likelihood decoder, that is so high as expected. All the accuracies in percentages can be seen in table 2.

PVD	MLE (Normal)	MLE (Poisson)
48	98	98

Table 2: Overfitted model results in percentages.

A visualization of average firing rate of neurons using heatmap

I do not know if this is common, but while studying this project, I have created a heatmap figure 2 from calculated average firing rate values of neurons. I have done this to visualize preferred directions of a multitude of neurons with 1 plot.

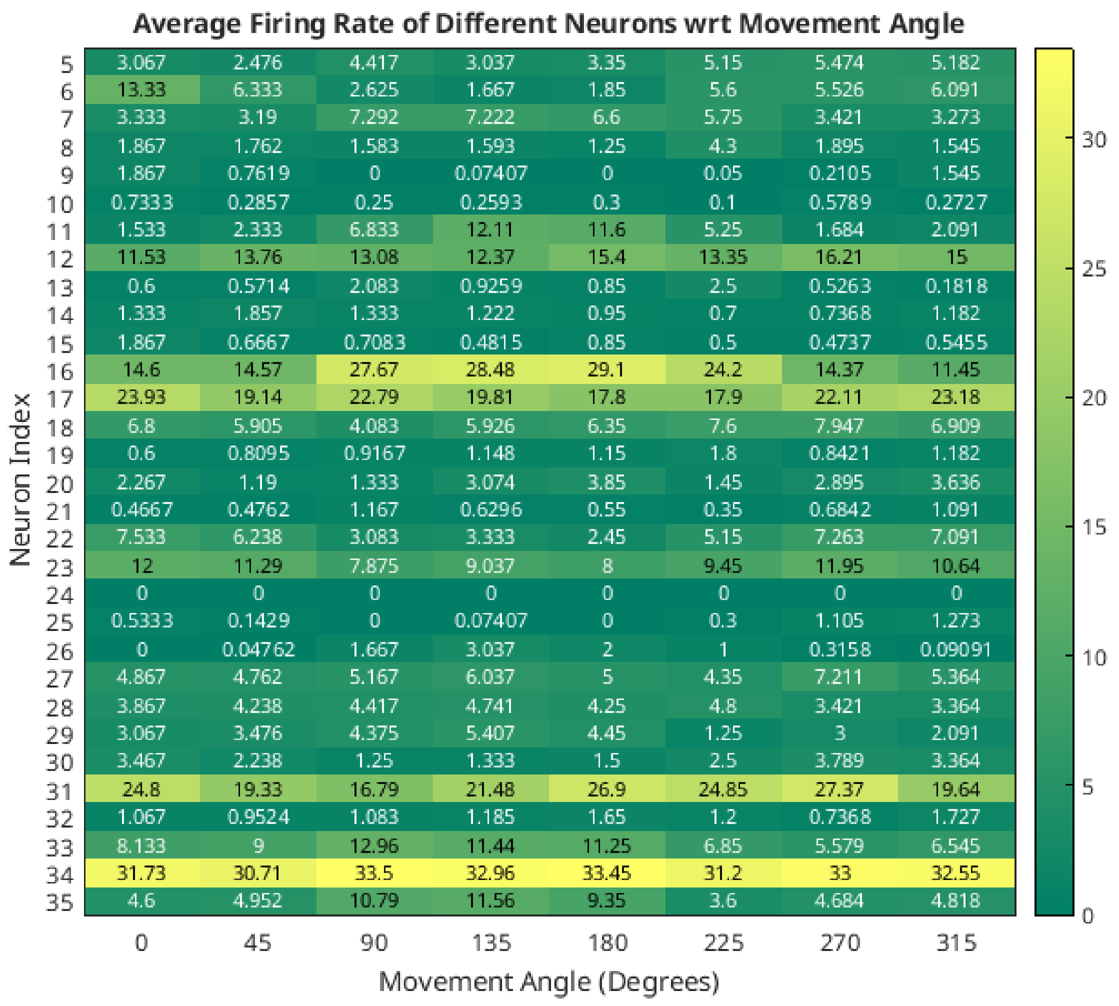
Plotting all 143 neurons in the heatmap is visually less appealing, so in figure 2, we can see the average firing rates of neurons indexed 5 to 35 where a 'yellower' color indicates a more firing in the given direction.

One example is the neuron 34 which apparently fires to movement in all directions, since it is above 30 in numbers, or bright yellow in colors for all degrees of movement.

Another example is neuron 24 which has the value 0 - or dark green - for all directions, meaning it does not fire for any direction.

Lastly, neuron 16 seems to prefer 90 to 180 degrees angle of direction, meaning a top-left movement of macaque monkey fires this neuron more than other directions.

With the help of this heatmap, I was able to visualize more than one neuron's average firing rate and check to see the most active ones - or most interesting ones, which I wanted to include here in this report.



Neuron Index	0	45	90	135	180	225	270	315
5	3.067	2.476	4.417	3.037	3.35	5.15	5.474	5.182
6	13.33	6.333	2.625	1.667	1.85	5.6	5.526	6.091
7	3.333	3.19	7.292	7.222	6.6	5.75	3.421	3.273
8	1.867	1.762	1.583	1.593	1.25	4.3	1.895	1.545
9	1.867	0.7619	0	0.07407	0	0.05	0.2105	1.545
10	0.7333	0.2857	0.25	0.2593	0.3	0.1	0.5789	0.2727
11	1.533	2.333	6.833	12.11	11.6	5.25	1.684	2.091
12	11.53	13.76	13.08	12.37	15.4	13.35	16.21	15
13	0.6	0.5714	2.083	0.9259	0.85	2.5	0.5263	0.1818
14	1.333	1.857	1.333	1.222	0.95	0.7	0.7368	1.182
15	1.867	0.6667	0.7083	0.4815	0.85	0.5	0.4737	0.5455
16	14.6	14.57	27.67	28.48	29.1	24.2	14.37	11.45
17	23.93	19.14	22.79	19.81	17.8	17.9	22.11	23.18
18	6.8	5.905	4.083	5.926	6.35	7.6	7.947	6.909
19	0.6	0.8095	0.9167	1.148	1.15	1.8	0.8421	1.182
20	2.267	1.19	1.333	3.074	3.85	1.45	2.895	3.636
21	0.4667	0.4762	1.167	0.6296	0.55	0.35	0.6842	1.091
22	7.533	6.238	3.083	3.333	2.45	5.15	7.263	7.091
23	12	11.29	7.875	9.037	8	9.45	11.95	10.64
24	0	0	0	0	0	0	0	0
25	0.5333	0.1429	0	0.07407	0	0.3	1.105	1.273
26	0	0.04762	1.667	3.037	2	1	0.3158	0.09091
27	4.867	4.762	5.167	6.037	5	4.35	7.211	5.364
28	3.867	4.238	4.417	4.741	4.25	4.8	3.421	3.364
29	3.067	3.476	4.375	5.407	4.45	1.25	3	2.091
30	3.467	2.238	1.25	1.333	1.5	2.5	3.789	3.364
31	24.8	19.33	16.79	21.48	26.9	24.85	27.37	19.64
32	1.067	0.9524	1.083	1.185	1.65	1.2	0.7368	1.727
33	8.133	9	12.96	11.44	11.25	6.85	5.579	6.545
34	31.73	30.71	33.5	32.96	33.45	31.2	33	32.55
35	4.6	4.952	10.79	11.56	9.35	3.6	4.684	4.818

Figure 2: Heatmap to visualize the mean firings of neurons based on direction of movement.