**Sensory population response decoding**

**Introduction:** In the nervous system, information about a sensory stimulus or an impending movement is represented by the profile of activity across a population of neurons. Neurons tend to be tuned for specific properties of a stimulus or movement, responding most for a preferred stimulus or movement and less in relation to how much the stimulus or movement differs from the preferred one. [1] To guide behavior, the brain must correctly decode this population response and extract the sensory information as reliably as possible [2]. Here we ask how we can learn what the stimulus is by looking at the neural responses of a population of neurons and identify the mathematical operation each neuron computes.

**Related work:** This work is built on an existing model from Theunissen and Miller’s paper in 1991 [4]. The four “low-velocity” interneurons of the cricket had sensitivity to stimuli spanning the entire 360 degree range of possible directions in the horizontal plane. The stimulus-response curve of each cell had a characteristic “tuning-curve” shape: the response was maximal for a particular air-current stimulus direction, and the response decreased as the stimulus was rotated from this optimal directions. The tuning curves of these four cells were equally spaced in angular separation. The goal of this work is to use this model to calculate the direction of the air-current stimuli and learn about the properties of each cell based on their responses.

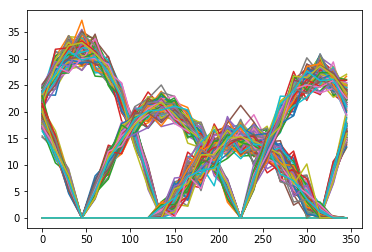
**Methods:** I used matlab to randomly generate four arrays representing the firing rate of each neuron, and store them in the file ‘tuning.mat’. Each neuron is stimulated by wind from all directions and each recordings last 10s. Each column of the array represents the firing rate of the neuron in response to each of the stimuli. The rows of the arrays represent 100 trials. Then I used Python 3 to plot the tuning curves with the following code.

plt.plot(np.transpose(data['stim']), np.transpose(data['neuron1']))

plt.plot(np.transpose(data['stim']), np.transpose(data['neuron2']))

plt.plot(np.transpose(data['stim']), np.transpose(data['neuron3']))

plt.plot(np.transpose(data['stim']), np.transpose(data['neuron4']))



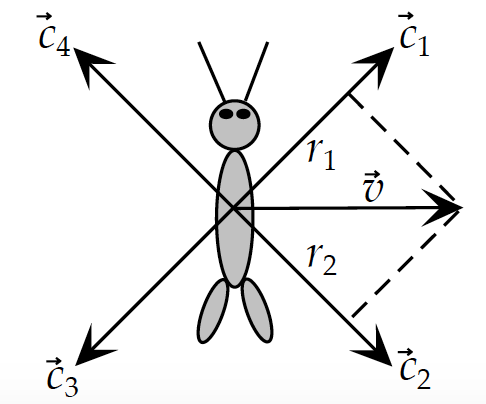
To test if the neurons are Poisson, I plotted the mean-variance curve for each neuron with the following code to learn about the Fano factor:

plt.plot(np.mean(data['neuron']\*10, axis = 0), np.var(data['neuron']\*10, axis = 0))

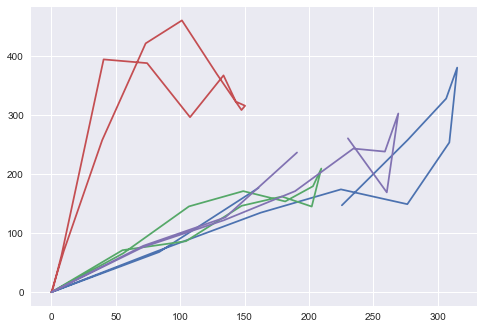
To model spike generation of each neuron, I will need the spike trains of each neuron and a series of stimuli that generate the spike trains. For this part, I used data from the H1 neuron of the fly from the lab of Dr Robert de Ruyter van Steveninck. I calculated the spike-triggered average by computing the average stimulus 300ms preceding a spike with the code in ‘spike triggered average’.

**Results:** For an unknown stimulus, the four neurons will generate responses r1, r2, r3 and r4. I calculated the population vector for these neurons: Each neuron adds in a component in its preferred direction, with a weight given by its firing rate divided by its maximum average firing rate, which approximates the projection of the wind direction, onto that preferred direction. Because of the preferred directions of these neurons are at 90 degrees to each other, only two neurons will fire for a single stimulus. Thus, we can compute the direction using the formula below:

np.degrees(np.arctan(np.mean(data1['r2'])/ max(np.mean(data['neuron1'], axis = 0))/np.mean(data1['r1'])\*a1)\* max(np.mean(data['neuron2'], axis = 0)))

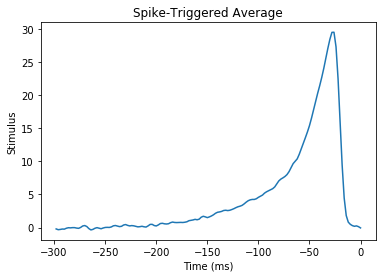


To determine if the neurons are Poisson, I plotted the mean-variance curve for each of the neuron:



Three of the neurons have a Fano factor of 1. So these three are Poisson, while the other one is not.

To learn about the spike-triggering stimulus of H1 neuron from fly, I plotted the following curve:



From the curve, we learn that this is a typical leaky integrate and fire neuron.

**Discussion and conclusion:** The work presented here provides a way to decode the responses of neurons and find the stimulus. However, this simplified model is based on several assumptions and is not accurate enough to reveal the stimulus from the real world. First, we simplified the nervous system by using only four interneurons whose preferred direction are at 90 degrees and whose tuning curves are cosine curves. In the future, an optimal decoder constructed from higher order interneurons should be used to more accurately estimate the stimulus. Second, we use mean firing rate over 10s, so we cannot calculate stimulus in a short temporal interval. For future studies that deal with short temporal interval of sensory inputs, we should decode based on spike trains: each spike input causes an increment in the decoder output according to the preferred stimulus [1]. Overall, even though this proposed model is able to decode simple sensory neuron responses, much more complex work need to be performed to accurately model sensory neural circuitry in the reality.

PS: The idea of this project comes from the Coursera course Computational Neuroscience. I regenerated the ‘tuning’ data set with Matlab and come up with most of the Python code to solve the problems.

References

1. Chaisanguanthum and Lisberger, 2011 A neurally efficient implementation of sensory population decoding J. Neurosci., 31 (2011), pp. 4868–4877
2. Jazayeri and Movshon, 2006 Optimal representation of sensory information by neural populations Nat. Neurosci., 9 (2006), pp. 690–696
3. Jacobs G A, Miller J P and Aldworth Z 2008 Computational mechanisms of mechanosensory processing in the cricket J. Exp. Biol. 211 1819–28
4. Theunissen and Miller, 1991 Representation of Sensory Information in the Cricket Cereal Sensory System. II. Information Theoretic Calculation of System Accuracy and Optimal Tuning-Curve Widths of Four Primary Interneurons. J. Neurophysiol., 66 (1991), pp. 1690–1703