Week 2: Measuring Mutual Information between Spike Trains

Maria Kesa

July 23, 2015

1 Calculating Mutual Information with the PyEntropy package

The aim of this week is to use the PyEntropy package to calculate the mutual information between spike trains. As in last week, the aim is purely technical– to establish the capability for calculating this quantity without any explicit semantics. The long term goal is to use Information Theory to comprehend the algorithms that the circuits are implementing as layed out in (1).
The PyEntropy package provides routines for calculating various information theoretic quantities. Currently, we are measuring the mutual information between two spike trains, however I am currently also investigating into how to measure mutual information between the activity of a population of neurons as in (2).

The PyEntropy accepts a sequence of 0's and 1's and makes a Discrete System object out of them. The representation of spike times in Brian is event-based meaning that only the times of the spikes are recorded. I wrote a function to convert this representation into a sequence of 0's and 1's using a number of bins that can be defined by the user.

The function is given below:

```
def generate_empty_bins(nr_of_bins,nr_of_neurons):
    spike_list=[]
    for neuron in range(0,nr_of_neurons):
        bin_list=[]
        for time_bin in range(0,nr_of_bins):
            bin_list.append(0)
        spike_list.append(bin_list)
    return spike_list

def discretize_spikes(nr_of_bins,time_interval,spikes,nr_of_neurons):
    time_bin=float(time_interval)/nr_of_bins
    time_bin_state=time_bin*second
    state=0
    spike_list=generate_empty_bins(nr_of_bins,nr_of_neurons)
    for spike in spikes:
        spike_index=spike[0]
        #This line makes the code work, but I have to look into Brian more in
```

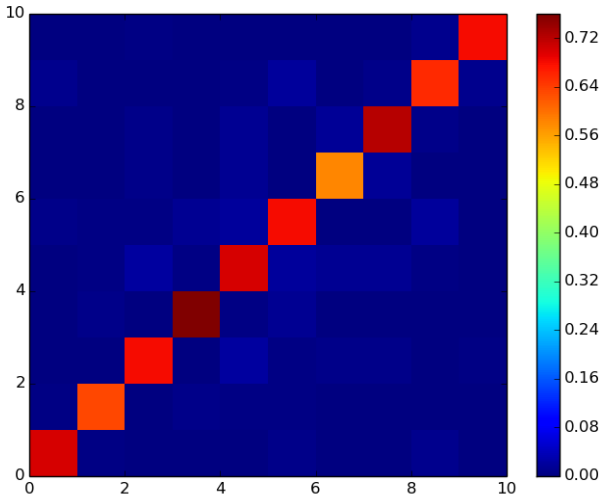
```
        #depth, something seems to be wrong with the timing
        spike_time=spike[1]
        if spike_time>time_bin_state:
            time_bin_state=time_bin_state+time_bin*second
            state+=1
            spike_list[spike_index][state]=1
        else:
            spike_list[spike_index][state]=1
    return spike_list
```

The setup of the simulation was the same as in the previous report. It contained a group of neurons that received excitatory input with Poisson statistics through AMPA-R synapses.

The mutual information is calculated in the following way:

```
def calculate_mutual_information(X,Y,n):
    s=pyentropy.systems.DiscreteSystem(X,(1,n), Y,(1,n))
    s.calculate_entropies(method='plugin', calc=['HX', 'HXY'])
    return s.I()
```

For the 10 neurons in the simulation, simulated for 0.5 seconds, the heat map of mutual informations is given below.



The mutual information is low for Poisson spiking neurons.

2 References

(1) Wibral, M., Lizier, J., Priesemann, V. "Bits from brains for biologically inspired computing", Frontiers in Robotics and AI 2015

(2) Quiroga, R., Panzeri, S., "Extracting information from neuronal populations: information theory and decoding approaches", Nature Reviews Neuroscience, 2009