EE 599: Fall 2018

Signal Processing and Control for Neural Movement

Predicting Tactile Decision

Final Project Report

Omer Solakli

Professor: Maryam Shanechi

Table of Contents

The Research

The Research is fundamentally to probe the neural dynamics in ALM which stands for Anterior Motor Cortex of adult mice and assessing its relationship with voluntary movement. The research is important because ALM is considered as analogue of the premotor cortex in human.

The research considers the EEG recordings of ALM neurons. They also use photo stimulation techniques such as silencing left or right hemisphere of ALM region, to assess how it correlates to the decision. The researchers conclude that unilateral silencing of ALM biases the movement. Hence, ALM control tactile decision making.

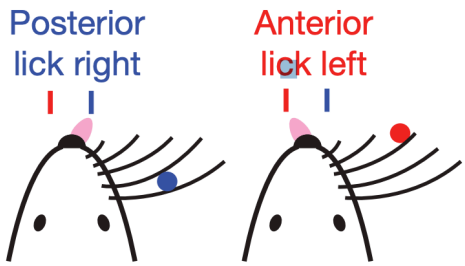

Figure 1: Overall Experiment

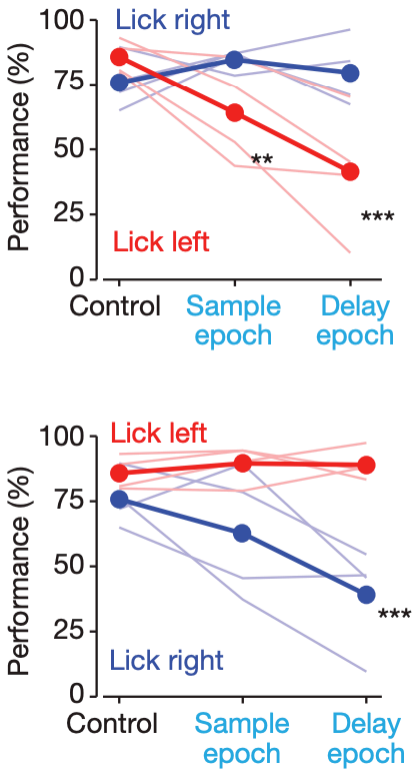

Figure 2: Performance on Photo stimulation - top: left stim bottom: right stim

Figure 1 depicts the setup. They put a pole on mice whisker and depending on the pole position (down or up), mice learns to lick right or left. Figure 2 depicts the performance of the photo stimulation trials, and it's clear that performance decreases with photo stimulation. Photo stimulation of left ALM increases the biasedness of licking right and vice versa.

The research also classifies ALM neurons in different classes and they conclude that "Pyramidal Putative Neurons," which corresponds to the neurons have a pyramidal spike shape, are the output of the ALM circuit. This suggest that they are the most important neurons on movement decision.

Experimental Setup

Mice are held still and measure the location of an object using their whiskers. For each trial a pole is introduced to the mice at the beginning of the Sample epoch. The mice are withheld from making a decision on delay epoch. This is where the prediction occurs, and the neural activity related to decision is high. After the delay epoch on response epoch the decision is made. The decision of the mice is recorded using a photodiode inside their mouth (Figure 3 and 4).

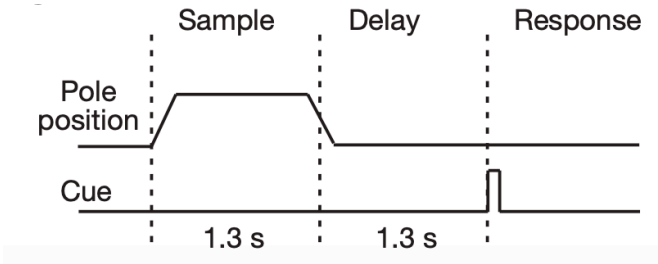

Figure 3: Epochs

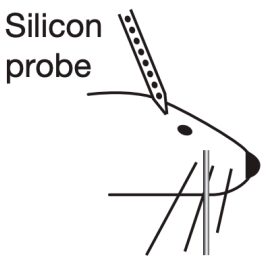

Figure 4: The Setup

Raster and PSTH for Each Neuron

First thing came up to my mind was to plot and see the PSTH and Raster activity for each neuron for each trial. Hence to visualise how they act for lickRight and lickLeft trials. Below I present all the raster and PSTH of 15 neurons. I actually spent lot of time on this part. I knew that I need to get used to the data. Even though it's presented on a structured way, the data had lots of dimensions and lots of different attributes that I didn't know since I have limited background on neuroscience.

Below is code snippet about how to extract the spikeTimes from the MATLAB cells.

```
for i=1:neuron_number %15 neurons in total in our case
    unit_tmp = obj.eventSeriesHash.value{i}; %for the ith neuron
    stable_trials_i = zeros(1,size(trialID,2)); %1x178 array in our case
    stable_trials_i(min(unit_tmp.eventTrials):max(unit_tmp.eventTrials)) = 1; %min 1 max 178
    spk_times_iCell = {};
    for i_trial = 1:size(trialID,2)
        % offset the spike times relative to trials start
        spk_times_iCell{i_trial} = unit_tmp.eventTimes(unit_tmp.eventTrials == i_trial)-t_TrialStart(i
    end
    PSTH_StartTime = -.52;
    PSTH_EndTime = 5.020;
    time = PSTH_StartTime:.001:PSTH_EndTime; %digitizing the time axis
    i_select_trial = find(hitR & ~stim & ~lickEarly & stable_trials_i & goodTrials); %finding the tria
    spk_times_select_trial_R = spk_times_iCell(i_select_trial);
    rasterR = [];
n_rep_R = size(spk_times_select_trial_R,2);
for i_rep = 1:n_rep_R
 raster_iRep_R = [zeros(size(spk_times_select_trial_R{1,i_rep},1),1)+i_rep spk_times_select_trial_R{1,
 rasterR = cat(1, rasterR, raster_iRep_R);
end
neuron_R{i}=rasterR;
i_select_trial = find(hitL & ~stim & ~lickEarly & stable_trials_i & goodTrials);
spk_times_select_trial_L = spk_times_iCell(i_select_trial);
rasterL = [];
n_rep_L = size(spk_times_select_trial_L,2);
for i_rep = 1:n_rep_L
    raster_iRep_L = [zeros(size(spk_times_select_trial_L{1,i_rep},1),1)+i_rep spk_times_select_trial_L
    rasterL = cat(1, rasterL, raster_iRep_L);
end
neuron_L{i}=rasterL;
```

Figure 5: Plotting raster and PSTH

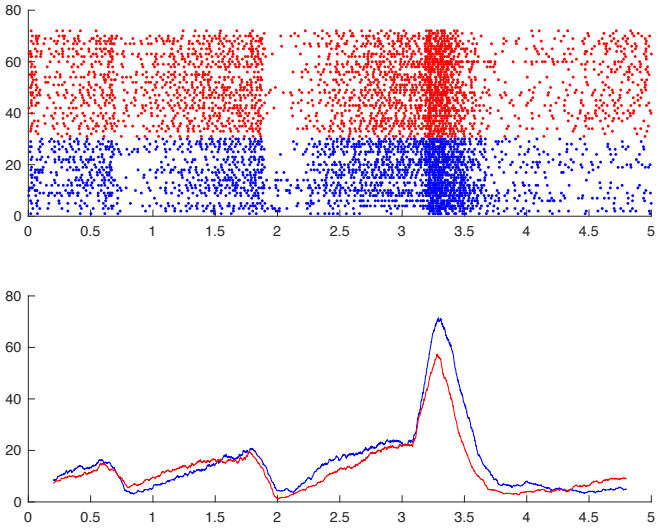

Figure 6: Neuron 1

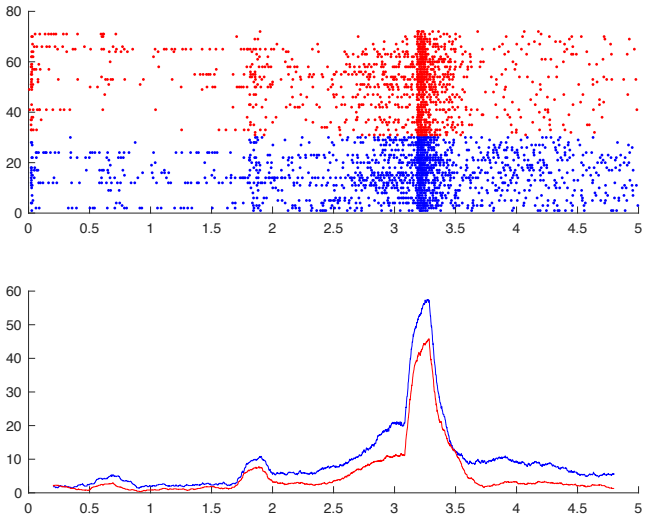

Figure 7: Neuron 2

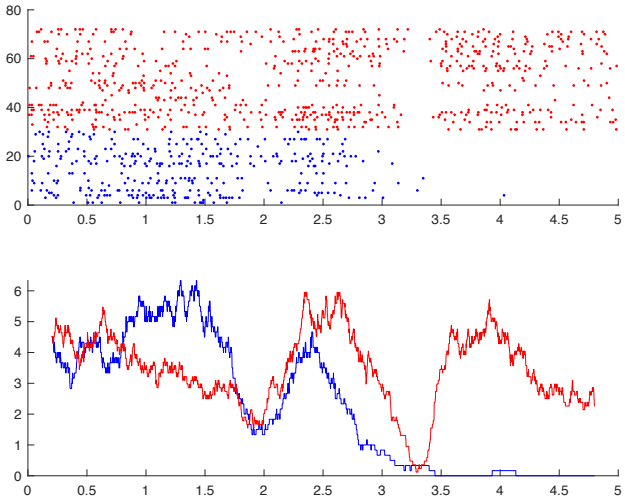

Figure 8: Neuron 3

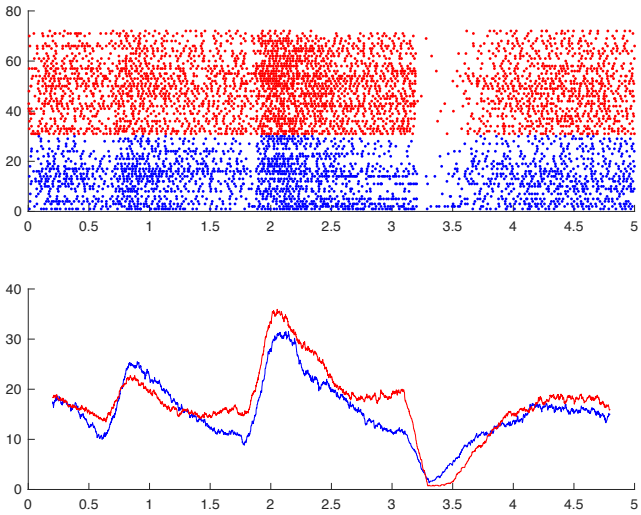

Figure 9: Neuron 4

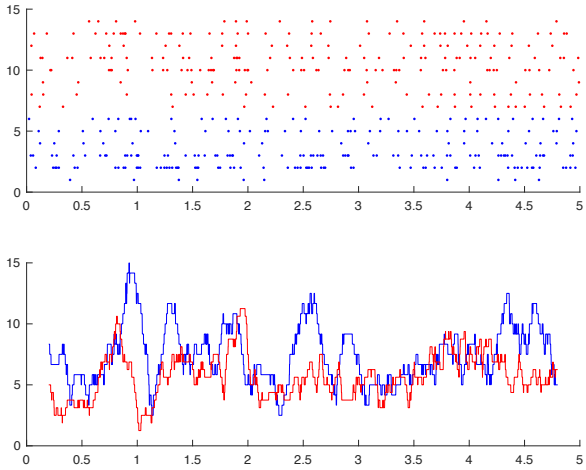

Figure 10: Neuron 5

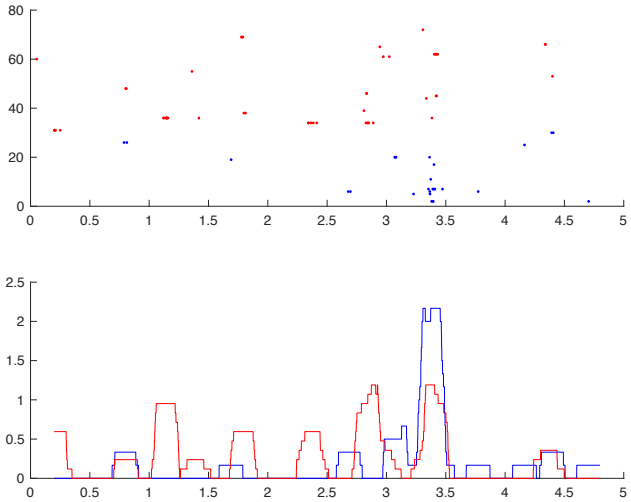

Figure 11: Neuron 6

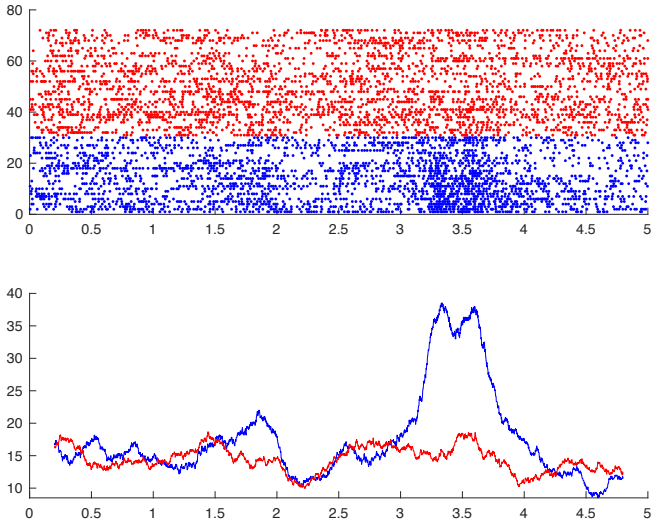

Figure 12: Neuron 7

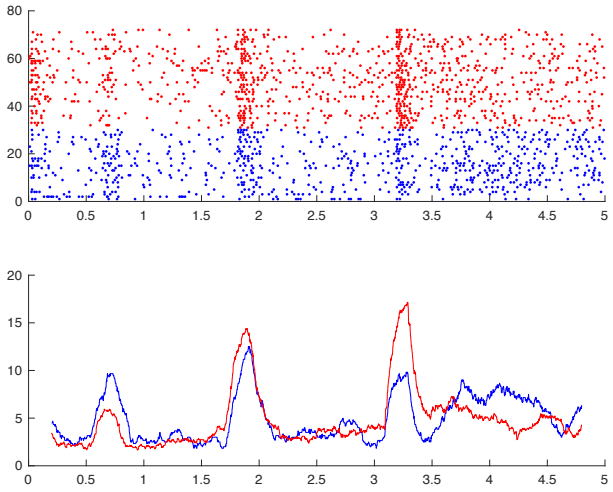

Figure 13: Neuron 8

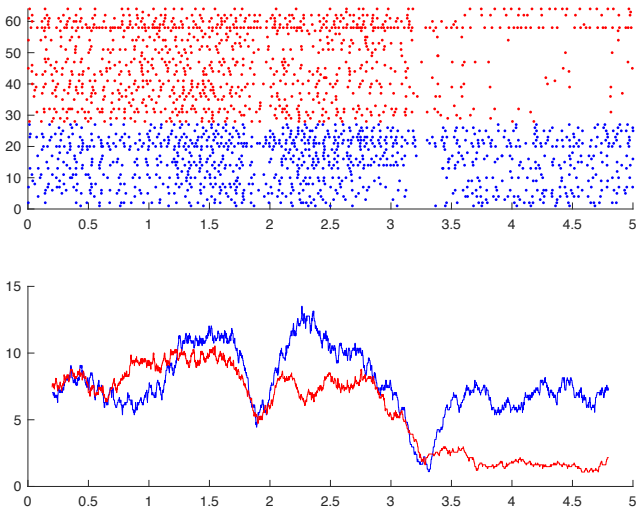

Figure 14: Neuron 9

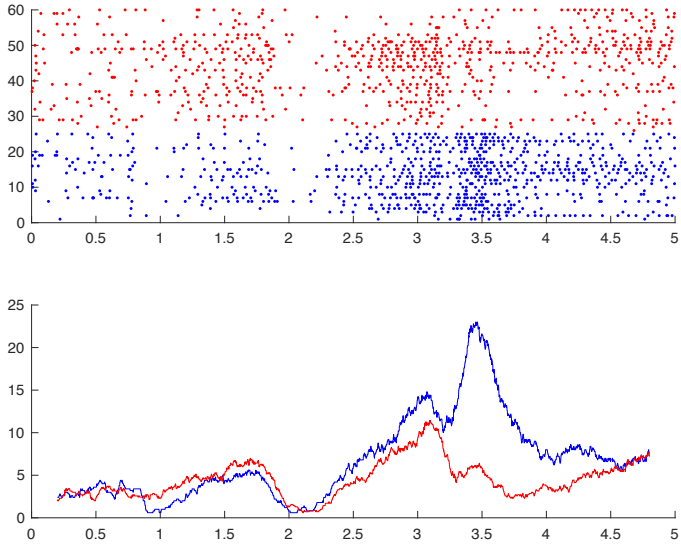

Figure 15: Neuron 10

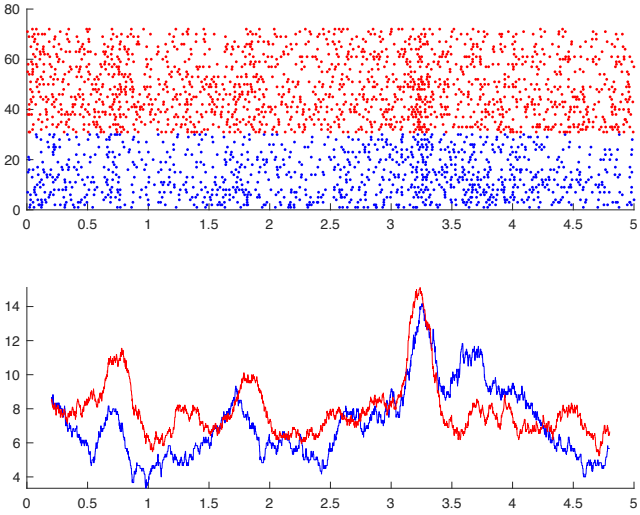

Figure 16: Neuron 11

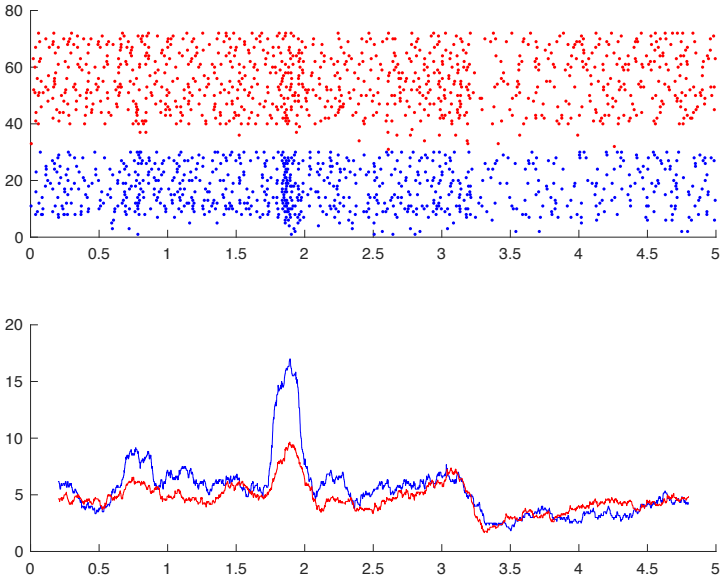

Figure 17: Neuron 12

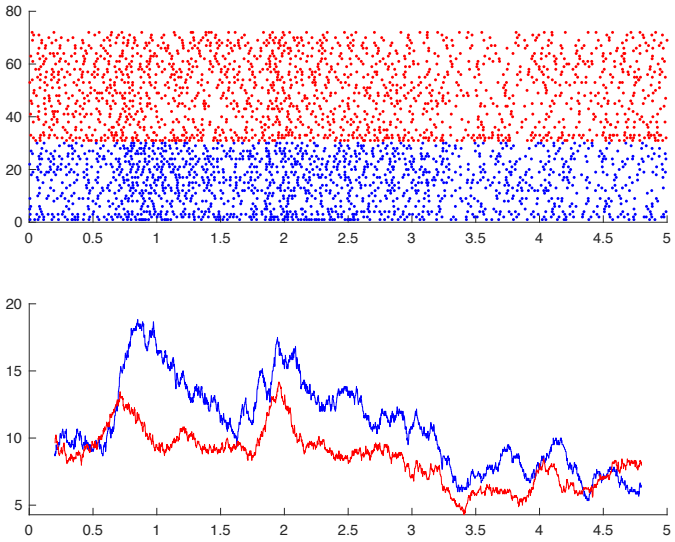

Figure 18: Neuron 13

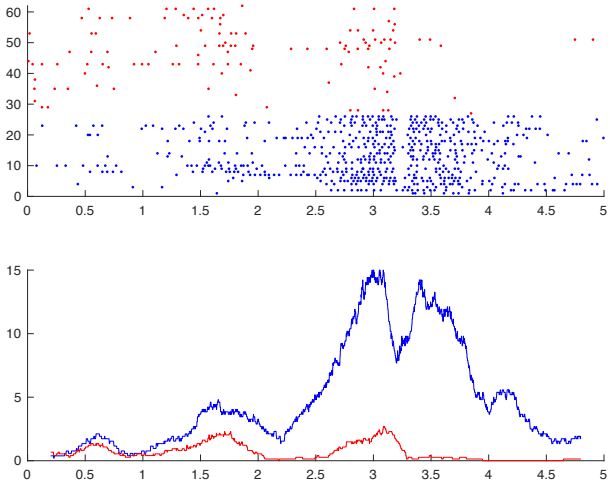

Figure 19: Neuron 14

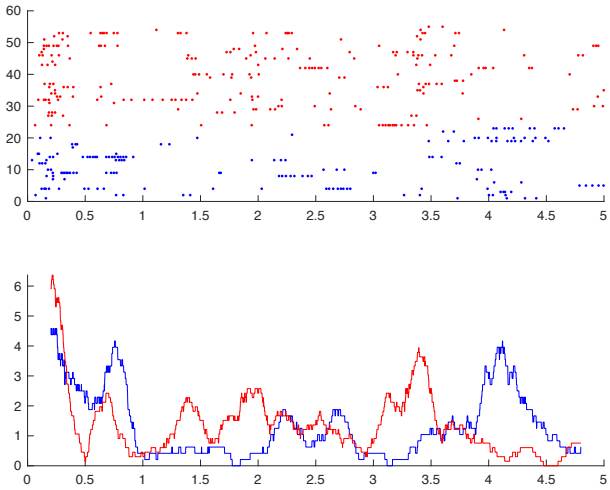

Figure 20: Neuron 15

Exploratory Data Analysis and Key Findings

Reading the paper and speaking with the Professor I learnt that the neurons tend to fire just before the end of the delay epoch (Figure 5). Thus, I analysed neural activity between 500ms before the end of the delay epoch and the end of the delay epoch (end of delay epoch is 3.17s in all trials).

Since the neural activity of different animals will be uncorrelated and very different, for this reason I decided to consider only 1 animal. Total #of trials from that animal is 178 when I disregarded the photo stimulation trials and disregarded the trials that are not good trials, defined by the paper. After that the #of trials is reduced to 72 in total (40 lickRight, 32 lickLeft). there were 15 measuring sites including 15 neurons.

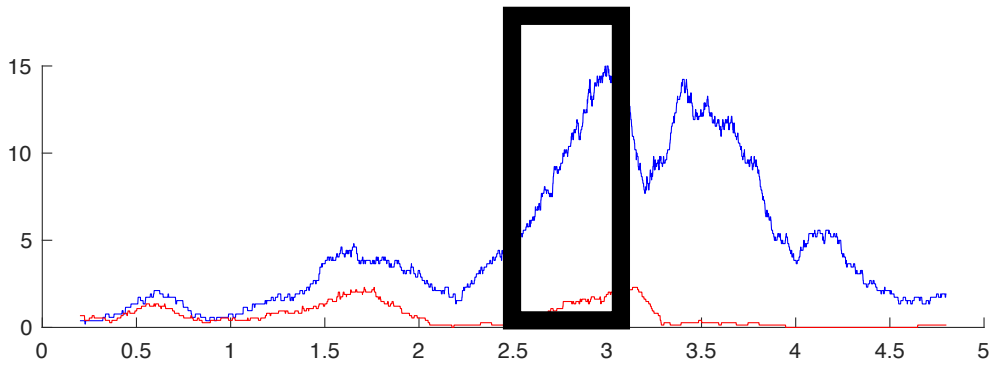

Figure 21: PSTH of a Pyramidal neuron

The Model

In this project, I am trying to predict whether a mouse is licked left or right just looking at it's spiking activity. To do that, I separate all of the lickRight and lickLeft trials extracting them from the MATLAB cells. Then I count the number of spikes for each right trial and for each left trial separately for each neuron. Since I already know whether a trial is resulted as lickLeft or lickRight for each neuron I can calculate the mean spike count across all lickLeft and lickRight trials. Thus, I have 2 different means for each neuron, lambdaRight and lambdaLeft.

For simple Poisson distribution, the maximum likelihood estimates of lambda is, it's sample mean. So that calculating the mean for each neuron gives me the MLE estimate of lambda for that neuron.

```
neuron_delay_1= neuron_1_times_L(:,2)<=3.17 & neuron_1_times_L(:,2)>2.67 ;
neuron_1_count_L=neuron_1_times_L(neuron_delay_1);

neuron1_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_1_count_L)
    for j=1:size(neuron_1_count_L)
    if neuron_1_count_L(j,1)==i
    neuron1_L(i)=neuron1_L(i)+1;
    end
    end
end
```

Figure 22: Counting the spikeTimes for a neuron

Given a test trial to be decoded, the decision rule is to calculate the likelihood of that trial. Using the mean lambdas, I found for each neuron and spike counts for the test trial I am able to calculate the probability of observed number of spikes for each neuron which is:

$$P(\mathrm{X} = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

Since I have 15 neurons, I assumed they are independently distributed. I am multiplying the of 15 probabilities to find the two likelihood for right and left lick. Then I get the logarithm to find the loglikelihood. If the loglikelihood of the right trial is bigger, the decision rule is to assess that the test trial is a right lick and vice versa.

In order to tackle down the time dependency of a firing activity of a neuron, I use leave-1 out cross validation to assess model fit quality. Thus, I leave out 1 trial train on the rest and try to decode the remaining trial using the others and leaving another trial on the next iteration.

```
for i=1:72
    j=randperm(72,1);
    X_test=X(j,:);
    X(j,:) = [];
    y_test(i)=X_test(16);
    spikesR=X(:,16)==1;
    Xr=X(spikesR,:);
    lambdasR=mean(Xr(:,1:15),1);
    spikesL=X(:,16)==0;
    Xl=X(spikesL,:);
    lambdasL=mean(Xl(:,1:15),1);
    Ll=poisspdf([X_test(:,1:15)],[lambdasL]);
    Ll=sum(log(Ll));
    disp(Ll);
    Lr=poisspdf([X_test(:,1:15)],[lambdasR]);
    Lr=sum(log(Lr));
    disp(Lr);
    if Lr>Ll
        X_predict(i)=1;
    else
        X_predict(i)=0;
    end
    X=[X_test;X];
end
```

Figure 23: Training and CV

The Results

Using leave 1 out cross-validation I was able to decode all 72 trials correctly given the others. Since there were no errors and not many data points this might be a sign that the model was working by chance. Thus, I altered the training labels randomly and fitted the model again. This yielded around 50% error which resulted that my initial model was correct.

To actually see whether the delay period is significant for decision making, I analysed the spike counts over all time and that yielded around 10% error. Hence, it can be stated that decision making process occurs during the delay period more significantly.

My project overlaps with the research paper and suggests that ALM neurons are important for tactile decision making and they have bilateral selectivity for decision behaviour prediction. Also, all of the neurons I work with are Pyramidal Putative Neurons, this also overlaps with the project result so that they are the output of the ALM circuit before the actual motor response occurs and they control directional licking.

In order to assess which neuron is able to predict better I tried to train on each neuron separately and find out that the neuron#12 is the most successful neuron to distinguish directional licking, whereas neuron#6 and neuron#11 are the worst.

```
error =

    38    29    34    29    36    40     2    39    40    28     0    48    21    15    26
```

Figure 24: Error of predicting after training for each neuron separately

Future Work and Remarks

In this project, I tried to analyse Spike Train data which I wasn't familiar with before. For me it was really exciting to actually try to analyse the data which has been used in a research paper. I spend most of my time reading the research and trying to get familiar with the data because there were so many dimensions and terms that I wasn't familiar with. For this kind of data simple Poisson model gave good results but in the future I'd like to develop more mathematically included models. One possible future addition for this project might be to analyse pairwise dependency of the neurons firing rate given a trial.

Bibliography

1. Nuo Li, Tsai-Wen Chen, Zengcai V. Guo, Charles R. Gerfen & Karel Svoboda. A motor cortex circuit for motor planning and movement. *Nature* 519, 51–56 (05 March 2015) PMID: 25731172 doi: 10.1038/nature14178

2. Nuo Li, Charles R Gerfen, Karel Svoboda (2014); Extracellular recordings from anterior lateral motor cortex (ALM) neurons of adult mice performing a tactile decision behavior. CRCNS.org. http://dx.doi.org/10.6080/K0MS3QNT

Code

```
trialID=obj.trialIds; %vector containing trialIds enumbered from 1 to
number of trials (178 in our case)

i_hitR=1; %indexing the trial types with respect to trialTypeStr object
i_hitL=2;
i_errR=3;
i_errL=4;
i_noLickR=5;
i_noLickL=6;
i_lickEarly=7;
i_stim=8; %trials under photostimulation

hitR=obj.trialTypeMat(1,:); %trialTypeMat is the matrix 8xnumberoftrials,
first row corresponds to lickRight trials (binary)
hitL=obj.trialTypeMat(2,:);
errR=obj.trialTypeMat(3,:);
errL=obj.trialTypeMat(4,:);
noLickR=obj.trialTypeMat(5,:);
noLickL=obj.trialTypeMat(6,:);
lickEarly=obj.trialTypeMat(7,:);
stim=obj.trialTypeMat(8,:);

t_TrialStart = obj.trialStartTimes; %start time of each trial
(1xnumberoftrials) vector


pole_in_Time = obj.trialPropertiesHash.value{1}; %start time of the sample
period relative to each trial
pole_out_Time = obj.trialPropertiesHash.value{2}; %end of sample period and
start of delay period
cue_Time = obj.trialPropertiesHash.value{3}; %end of delay period
stim_Type=obj.trialPropertiesHash.value{5}; %"0"--non-stimulation trials;
"1"--PT axonal stimulation; "2"--IT axonal stimulation; "NaN"--discard
(stimulation configuration for other purposes, should not analyze)

goodTrials = obj.trialPropertiesHash.value{4}'; %178x1 binary vector
neuron_number=size(obj.eventSeriesHash.value,2); % determining number of
neurons (or measuring sites)


neuron_R={};
neuron_L={};

for i=1:neuron_number %15 neurons in total in our case
    unit_tmp = obj.eventSeriesHash.value{i}; %for the ith neuron
    stable_trials_i = zeros(1,size(trialID,2)); %1x178 array in our case
    stable_trials_i(min(unit_tmp.eventTrials):max(unit_tmp.eventTrials)) =
1; %min 1 max 178
    spk_times_iCell = {};
    for i_trial = 1:size(trialID,2)
        % offset the spike times relative to trials start
        spk_times_iCell{i_trial} = unit_tmp.eventTimes(unit_tmp.eventTrials
== i_trial)-t_TrialStart(i_trial);
    end
```

```
    PSTH_StartTime = -.52;
    PSTH_EndTime = 5.020;
    time = PSTH_StartTime:.001:PSTH_EndTime; %digitizing the time axis
    i_select_trial = find(hitR & ~stim & ~lickEarly & stable_trials_i &
goodTrials); %finding the trials are not stim, not lickearly, stable, and
good
    spk_times_select_trial_R = spk_times_iCell(i_select_trial);
    rasterR = [];
n_rep_R = size(spk_times_select_trial_R,2);
for i_rep = 1:n_rep_R
 raster_iRep_R = [zeros(size(spk_times_select_trial_R{1,i_rep},1),1)+i_rep
spk_times_select_trial_R{1,i_rep}];
 rasterR = cat(1, rasterR, raster_iRep_R);
end
neuron_R{i}=rasterR;
i_select_trial = find(hitL & ~stim & ~lickEarly & stable_trials_i &
goodTrials);
spk_times_select_trial_L = spk_times_iCell(i_select_trial);
rasterL = [];
n_rep_L = size(spk_times_select_trial_L,2);
for i_rep = 1:n_rep_L
    raster_iRep_L =
[zeros(size(spk_times_select_trial_L{1,i_rep},1),1)+i_rep
spk_times_select_trial_L{1,i_rep}];
    rasterL = cat(1, rasterL, raster_iRep_L);
end
neuron_L{i}=rasterL;
end

neuron_1_times_L=neuron_L{1,1};
neuron_2_times_L=neuron_L{1,2};
neuron_3_times_L=neuron_L{1,3};
neuron_4_times_L=neuron_L{1,4};
neuron_5_times_L=neuron_L{1,5};
neuron_6_times_L=neuron_L{1,6};
neuron_7_times_L=neuron_L{1,7};
neuron_8_times_L=neuron_L{1,8};
neuron_9_times_L=neuron_L{1,9};
neuron_10_times_L=neuron_L{1,10};
neuron_11_times_L=neuron_L{1,11};
neuron_12_times_L=neuron_L{1,12};
neuron_13_times_L=neuron_L{1,13};
neuron_14_times_L=neuron_L{1,14};
neuron_15_times_L=neuron_L{1,15};

neuron_delay_1= neuron_1_times_L(:,2)<=35 & neuron_1_times_L(:,2)>0 ;
neuron_1_count_L=neuron_1_times_L(neuron_delay_1);

neuron1_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_1_count_L)
    for j=1:size(neuron_1_count_L)
    if neuron_1_count_L(j,1)==i
    neuron1_L(i)=neuron1_L(i)+1;
    end
    end
end




neuron_delay_2= neuron_2_times_L(:,2)<=5& neuron_2_times_L(:,2)>0 ;
neuron_2_count_L=neuron_2_times_L(neuron_delay_2);
```

```
neuron2_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_2_count_L)
    for j=1:size(neuron_2_count_L)
    if neuron_2_count_L(j,1)==i
    neuron2_L(i)=neuron2_L(i)+1;
    end
    end
end

neuron_delay_3= neuron_3_times_L(:,2)<=5 & neuron_3_times_L(:,2)>0 ;
neuron_3_count_L=neuron_3_times_L(neuron_delay_3);

neuron3_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_1_count_L)
    for j=1:size(neuron_3_count_L)
    if neuron_3_count_L(j,1)==i
    neuron3_L(i)=neuron3_L(i)+1;
    end
    end
end

neuron_delay_4= neuron_4_times_L(:,2)<=5 & neuron_4_times_L(:,2)>0 ;
neuron_4_count_L=neuron_4_times_L(neuron_delay_4);

neuron4_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_4_count_L)
    for j=1:size(neuron_4_count_L)
    if neuron_4_count_L(j,1)==i
    neuron4_L(i)=neuron4_L(i)+1;
    end
    end
end

neuron_delay_5= neuron_5_times_L(:,2)<=5 & neuron_5_times_L(:,2)>0 ;
neuron_5_count_L=neuron_5_times_L(neuron_delay_5);

neuron5_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_5_count_L)
    for j=1:size(neuron_5_count_L)
    if neuron_5_count_L(j,1)==i
    neuron5_L(i)=neuron5_L(i)+1;
    end
    end
end

neuron_delay_6= neuron_6_times_L(:,2)<=5 & neuron_6_times_L(:,2)>0;
neuron_6_count_L=neuron_6_times_L(neuron_delay_6);

neuron6_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_6_count_L)
    for j=1:size(neuron_6_count_L)
    if neuron_6_count_L(j,1)==i
    neuron6_L(i)=neuron6_L(i)+1;
    end
    end
end

neuron_delay_7= neuron_7_times_L(:,2)<=5 & neuron_7_times_L(:,2)>0 ;
neuron_7_count_L=neuron_7_times_L(neuron_delay_7);
```

```
neuron7_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_7_count_L)
    for j=1:size(neuron_7_count_L)
    if neuron_7_count_L(j,1)==i
    neuron7_L(i)=neuron7_L(i)+1;
    end
    end
end

neuron_delay_8= neuron_8_times_L(:,2)<=5 & neuron_8_times_L(:,2)>0 ;
neuron_8_count_L=neuron_8_times_L(neuron_delay_8);

neuron8_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_8_count_L)
    for j=1:size(neuron_8_count_L)
    if neuron_8_count_L(j,1)==i
    neuron8_L(i)=neuron8_L(i)+1;
    end
    end
end

neuron_delay_9= neuron_9_times_L(:,2)<=5 & neuron_9_times_L(:,2)>0 ;
neuron_9_count_L=neuron_9_times_L(neuron_delay_9);

neuron9_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_9_count_L)
    for j=1:size(neuron_9_count_L)
    if neuron_9_count_L(j,1)==i
    neuron9_L(i)=neuron9_L(i)+1;
    end
    end
end

neuron_delay_10= neuron_10_times_L(:,2)<=5& neuron_10_times_L(:,2)>0 ;
neuron_10_count_L=neuron_10_times_L(neuron_delay_10);

neuron10_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_10_count_L)
    for j=1:size(neuron_10_count_L)
    if neuron_10_count_L(j,1)==i
    neuron10_L(i)=neuron10_L(i)+1;
    end
    end
end

neuron_delay_11= neuron_11_times_L(:,2)<=5 & neuron_11_times_L(:,2)>0 ;
neuron_11_count_L=neuron_11_times_L(neuron_delay_11);

neuron11_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_11_count_L)
    for j=1:size(neuron_11_count_L)
    if neuron_11_count_L(j,1)==i
    neuron11_L(i)=neuron1_L(i)+1;
    end
    end
end

neuron_delay_12= neuron_12_times_L(:,2)<=5 & neuron_12_times_L(:,2)>0;
neuron_12_count_L=neuron_12_times_L(neuron_delay_12);

neuron12_L=zeros(1,max(neuron_1_count_L));
```

```
for i=1:max(neuron_12_count_L)
    for j=1:size(neuron_12_count_L)
    if neuron_12_count_L(j,1)==i
    neuron12_L(i)=neuron12_L(i)+1;
    end
    end
end

neuron_delay_13= neuron_13_times_L(:,2)<=5 & neuron_13_times_L(:,2)>0 ;
neuron_13_count_L=neuron_13_times_L(neuron_delay_13);

neuron13_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_13_count_L)
    for j=1:size(neuron_13_count_L)
    if neuron_13_count_L(j,1)==i
    neuron13_L(i)=neuron13_L(i)+1;
    end
    end
end

neuron_delay_14= neuron_14_times_L(:,2)<=5 & neuron_14_times_L(:,2)>0 ;
neuron_14_count_L=neuron_14_times_L(neuron_delay_14);

neuron14_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_14_count_L)
    for j=1:size(neuron_14_count_L)
    if neuron_14_count_L(j,1)==i
    neuron14_L(i)=neuron14_L(i)+1;
    end
    end
end

neuron_delay_15= neuron_15_times_L(:,2)<=5 & neuron_15_times_L(:,2)>0 ;
neuron_15_count_L=neuron_15_times_L(neuron_delay_15);

neuron15_L=zeros(1,max(neuron_1_count_L));
for i=1:max(neuron_15_count_L)
    for j=1:size(neuron_15_count_L)
    if neuron_15_count_L(j,1)==i
    neuron15_L(i)=neuron15_L(i)+1;
    end
    end
end


neuron_1_times_R=neuron_R{1,1};
neuron_2_times_R=neuron_R{1,2};
neuron_3_times_R=neuron_R{1,3};
neuron_4_times_R=neuron_R{1,4};
neuron_5_times_R=neuron_R{1,5};
neuron_6_times_R=neuron_R{1,6};
neuron_7_times_R=neuron_R{1,7};
neuron_8_times_R=neuron_R{1,8};
neuron_9_times_R=neuron_R{1,9};
neuron_10_times_R=neuron_R{1,10};
neuron_11_times_R=neuron_R{1,11};
neuron_12_times_R=neuron_R{1,12};
neuron_13_times_R=neuron_R{1,13};
neuron_14_times_R=neuron_R{1,14};
neuron_15_times_R=neuron_R{1,15};
```

```
neuron_delay_1= neuron_1_times_R(:,2)<=3.17 & neuron_1_times_R(:,2)>2.67 ;
neuron_1_count_R=neuron_1_times_R(neuron_delay_1);

neuron1_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_1_count_R)
    for j=1:size(neuron_1_count_R)
    if neuron_1_count_R(j,1)==i
    neuron1_R(i)=neuron1_R(i)+1;
    end
    end
end




neuron_delay_2= neuron_2_times_R(:,2)<=3.17 & neuron_2_times_R(:,2)>2.67 ;
neuron_2_count_R=neuron_2_times_R(neuron_delay_2);

neuron2_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_2_count_R)
    for j=1:size(neuron_2_count_R)
    if neuron_2_count_R(j,1)==i
    neuron2_R(i)=neuron2_R(i)+1;
    end
    end
end

neuron_delay_3= neuron_3_times_R(:,2)<=3.17 & neuron_3_times_R(:,2)>2.67 ;
neuron_3_count_R=neuron_3_times_R(neuron_delay_3);

neuron3_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_1_count_R)
    for j=1:size(neuron_3_count_R)
    if neuron_3_count_R(j,1)==i
    neuron3_R(i)=neuron3_R(i)+1;
    end
    end
end

neuron_delay_4= neuron_4_times_R(:,2)<=3.17 & neuron_4_times_R(:,2)>2.6 ;
neuron_4_count_R=neuron_4_times_R(neuron_delay_4);

neuron4_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_4_count_R)
    for j=1:size(neuron_4_count_R)
    if neuron_4_count_R(j,1)==i
    neuron4_R(i)=neuron4_R(i)+1;
    end
    end
end

neuron_delay_5= neuron_5_times_R(:,2)<=3.17 & neuron_5_times_R(:,2)>2.67 ;
neuron_5_count_R=neuron_5_times_R(neuron_delay_5);

neuron5_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_5_count_R)
    for j=1:size(neuron_5_count_R)
    if neuron_5_count_R(j,1)==i
    neuron5_R(i)=neuron5_R(i)+1;
    end
    end
```

```
end

neuron_delay_6= neuron_6_times_R(:,2)<=3.17 & neuron_6_times_R(:,2)>2.67 ;
neuron_6_count_R=neuron_6_times_R(neuron_delay_6);

neuron6_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_6_count_L)
    for j=1:size(neuron_6_count_R)
    if neuron_6_count_R(j,1)==i
    neuron6_R(i)=neuron6_R(i)+1;
    end
    end
end

neuron_delay_7= neuron_7_times_R(:,2)<=3.17 & neuron_7_times_R(:,2)>2.6 ;
neuron_7_count_R=neuron_7_times_R(neuron_delay_7);

neuron7_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_7_count_R)
    for j=1:size(neuron_7_count_R)
    if neuron_7_count_R(j,1)==i
    neuron7_L(i)=neuron7_L(i)+1;
    end
    end
end

neuron_delay_8= neuron_8_times_R(:,2)<=3.17 & neuron_8_times_R(:,2)>2.67 ;
neuron_8_count_R=neuron_8_times_R(neuron_delay_8);

neuron8_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_8_count_R)
    for j=1:size(neuron_8_count_R)
    if neuron_8_count_R(j,1)==i
    neuron8_R(i)=neuron8_R(i)+1;
    end
    end
end

neuron_delay_9= neuron_9_times_R(:,2)<=3.17 & neuron_9_times_R(:,2)>2.67 ;
neuron_9_count_R=neuron_9_times_R(neuron_delay_9);

neuron9_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_9_count_R)
    for j=1:size(neuron_9_count_R)
    if neuron_9_count_L(j,1)==i
    neuron9_R(i)=neuron9_R(i)+1;
    end
    end
end

neuron_delay_10= neuron_10_times_R(:,2)<=3.17 & neuron_10_times_R(:,2)>2.67
;
neuron_10_count_R=neuron_10_times_R(neuron_delay_10);

neuron10_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_10_count_R)
    for j=1:size(neuron_10_count_R)
    if neuron_10_count_R(j,1)==i
    neuron10_R(i)=neuron10_R(i)+1;
    end
    end
```

```
end

neuron_delay_11= neuron_11_times_R(:,2)<=3.17 & neuron_11_times_R(:,2)>2.67
;
neuron_11_count_R=neuron_11_times_R(neuron_delay_11);

neuron11_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_11_count_R)
    for j=1:size(neuron_11_count_L)
    if neuron_11_count_L(j,1)==i
    neuron11_L(i)=neuron1_L(i)+1;
    end
    end
end

neuron_delay_12= neuron_12_times_R(:,2)<=3.17 & neuron_12_times_R(:,2)>2.67
;
neuron_12_count_R=neuron_12_times_R(neuron_delay_12);

neuron12_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_12_count_R)
    for j=1:size(neuron_12_count_R)
    if neuron_12_count_R(j,1)==i
    neuron12_R(i)=neuron12_R(i)+1;
    end
    end
end

neuron_delay_13= neuron_13_times_R(:,2)<=3.17 & neuron_13_times_R(:,2)>2.67
;
neuron_13_count_R=neuron_13_times_R(neuron_delay_13);

neuron13_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_13_count_R)
    for j=1:size(neuron_13_count_R)
    if neuron_13_count_R(j,1)==i
    neuron13_R(i)=neuron13_R(i)+1;
    end
    end
end

neuron_delay_14= neuron_14_times_R(:,2)<=3.17 & neuron_14_times_R(:,2)>2.67
;
neuron_14_count_R=neuron_14_times_R(neuron_delay_14);

neuron14_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_14_count_R)
    for j=1:size(neuron_14_count_R)
    if neuron_14_count_R(j,1)==i
    neuron14_R(i)=neuron14_R(i)+1;
    end
    end
end

neuron_delay_15= neuron_15_times_R(:,2)<=3.17 & neuron_15_times_R(:,2)>2.67
;
neuron_15_count_R=neuron_15_times_R(neuron_delay_15);

neuron15_R=zeros(1,max(neuron_1_count_R));
for i=1:max(neuron_15_count_R)
    for j=1:size(neuron_15_count_R)
```

```
    if neuron_15_count_R(j,1)==i
    neuron15_L(i)=neuron15_L(i)+1;
    end
    end
end

%%sample mean is the unbiased estimator for MLE of Poisson


X_R=[neuron1_R' , neuron2_R',neuron3_R', neuron4_R' ,neuron5_R',
neuron6_R', neuron7_R', neuron8_R', neuron9_R' ,neuron10_R', neuron11_R',
neuron12_R', neuron13_R',neuron14_R', neuron15_R'];

X_L=[neuron1_L',neuron2_L',neuron3_L',neuron4_L',neuron5_L' ,neuron6_L',
neuron7_L',neuron8_L',neuron9_L',neuron10_L', neuron11_L', neuron12_L',
neuron13_L', neuron14_L', neuron15_L'];

X=[X_R;
   X_L];
y=[ones(30,1);zeros(42,1)];

X=[X,y];
X_predict=zeros(1,72);
y_test=zeros(1,72);
for i=1:72
  j=randperm(72,1);
  X_test=X(j,:);
  X(j,:) = [];
  y_test(i)=X_test(16);
  spikesR=X(:,16)==1;
  Xr=X(spikesR,:);
  lambdasR=mean(Xr(:,1:15),1);
  spikesL=X(:,16)==0;
  Xl=X(spikesL,:);
  lambdasL=mean(Xl(:,1:15),1);
  Ll=poisspdf([X_test(:,1:15)],[lambdasL]);
  Ll=sum(log(Ll));
  disp(Ll);
  Lr=poisspdf([X_test(:,1:15)],[lambdasR]);
  Lr=sum(log(Lr));
  disp(Lr);
  if Lr>Ll
      X_predict(i)=1;
  else
      X_predict(i)=0;
  end
  X=[X_test;X];
end
error=0;
for j=1:length(X_predict)
    if X_predict(j)~=y_test(j)
    error=error+1;
    end
end

p_err=error/72;

tf = isequal(X_predict,y_test);
tpye={};
```