July 2021

دانشگاه تربیت مدرّس

. In the name of GOD .

# Neuroscience
# project_part2_final

Presented to

Dr. Bahmani

Submitted by

Poorya Aghaomidi

9961391001

Q1 - a

1.1.1. File name :

1.a

1.1.2. Figures :

____

```
IM_main =

    0.5721
```

1.1.3. Explanation :

در این سوال می خواهیم mutual information را محاسبه کنیم. برای این کار ابتدا trial های مربوط به هر location را شناسایی کرده و سپس تعداد اسپایک های رخ داده در بازه 1000 تا 2000 میلی ثانیه در trial های مربوط به هر location را محاسبه کرده تا ماتریس spike count مربوط به هر stimulus را داشته باشیم.

پس از اینکه به ازای هر location یک آرایه از spike count ها ساختیم، باید احتمال رخ دادن response به شرط stimulus را حساب کنیم. برای این کار ابتدا بازه های هیستوگرام را تعریف می کنیم به طوری که تعداد بین ها برای همه stimulus ها یکسان فرض شده است. پس از آن مقادیر حاصل از هیستوگرام را بر تعداد اعضای آرایه تقسیم می کنیم تا احتمال به دست آید. پس از آن با قرار دادن احتمال در فرمول آنتروپی، آنتروپی مربوط به آن stimulus را بدست می آوریم که با میانگین گیری آنتروپی همه stimulus ها با توجه به احتمال رخ دادن هرکدام، آنتروپی نویز بدست می آید.

برای محاسبه آنتروپی کل دو راه داریم، یکی این که با ضرب متناظر احتمال r به شرط s در احتمال رخ دادن آن stimulus و جمع نتایج بدست آمده برای همه stimulus ها با هم، احتمال r را محاسبه می کنیم. در ادامه با قرار دادن این احتمال در فرمول آنتروپی، آنتروپی کل بدست می آید. و روش دیگر که در کد اعمال شده این است که با زیر هم قرار دادن trial ها بدون توجه به stimulus آن ها، هیستوگرام آن را کشیده و بر تعداد اعضا تقسیم کرده تا احتمال به دست آید. سپس با قرار دادن احتمال در فرمول آنتروپی، آنتروپی کل را محاسبه می کنیم. اگر از این مقدار آنتروپی نویز را کم کنیم به mutual information می رسیم.

برای اطمینان از مقادیر بدست آمده shuffle انجام شده که توضیحات آن در قسمت c بیان شده است.

## 1.1.4. Codes :

```
%% initialize
locations = unique(Event.mgs.codes(:,5));
neuron = spike{4, 2} ;

L51 = find(Event.mgs.codes(:,5) == locations(1));
L52 = find(Event.mgs.codes(:,5) == locations(2));
L53 = find(Event.mgs.codes(:,5) == locations(3));
L54 = find(Event.mgs.codes(:,5) == locations(4));
L55 = find(Event.mgs.codes(:,5) == locations(5));
L56 = find(Event.mgs.codes(:,5) == locations(6));

for i=1:numel(L51)
    spike_count_51(1,i) = numel(find(neuron.mgs(L51(i),1000:2000) == 1));
end

for i=1:numel(L52)
    spike_count_52(1,i) = numel(find(neuron.mgs(L52(i),1000:2000) == 1));
end

for i=1:numel(L53)
    spike_count_53(1,i) = numel(find(neuron.mgs(L53(i),1000:2000) == 1));
end

for i=1:numel(L54)
    spike_count_54(1,i) = numel(find(neuron.mgs(L54(i),1000:2000) == 1));
end

for i=1:numel(L55)
    spike_count_55(1,i) = numel(find(neuron.mgs(L55(i),1000:2000) == 1));
end

for i=1:numel(L56)
    spike_count_56(1,i) = numel(find(neuron.mgs(L56(i),1000:2000) == 1));
end

%%  H noise
% 51
edge = [ min(spike_count_51) : (max(spike_count_51)-min(spike_count_51))/3:
max(spike_count_51) ] ;
S_51_Prob = histogram(sort(spike_count_51),edge).Values ./ numel(spike_count_51) ;
S_51_Prob(S_51_Prob == 0) = [] ;
entropy_51 = - sum(S_51_Prob .* log2(S_51_Prob)) ;

% 52
edge = [ min(spike_count_52) : (max(spike_count_52)-min(spike_count_52))/3:
max(spike_count_52) ] ;
S_52_Prob = histogram(sort(spike_count_52),edge).Values ./ numel(spike_count_52) ;
S_52_Prob(S_52_Prob == 0) = [] ;
entropy_52 = - sum(S_52_Prob .* log2(S_52_Prob)) ;

% 53
edge = [ 5 : (36-5)/3 : 36 ] ;
S_53_Prob = histogram(sort(spike_count_53),edge).Values ./ numel(spike_count_53) ;
S_53_Prob(S_53_Prob == 0) = [] ;
entropy_53 = - sum(S_53_Prob .* log2(S_53_Prob)) ;

% 54
edge = [ min(spike_count_54) : (max(spike_count_54)-min(spike_count_54))/3:
max(spike_count_54) ] ;
S_54_Prob = histogram(sort(spike_count_54),edge).Values ./ numel(spike_count_54) ;
S_54_Prob(S_54_Prob == 0) = [] ;
entropy_54 = - sum(S_54_Prob .* log2(S_54_Prob)) ;

% 55
edge = [ min(spike_count_55) : (max(spike_count_55)-min(spike_count_55))/3:
max(spike_count_55) ] ;
S_55_Prob = histogram(sort(spike_count_55),edge).Values ./ numel(spike_count_55) ;
S_55_Prob(S_55_Prob == 0) = [] ;
entropy_55 = - sum(S_55_Prob .* log2(S_55_Prob)) ;
```

```
% 56
edge = [ min(spike_count_56) : (max(spike_count_56)-min(spike_count_56))/3:
max(spike_count_56) ] ;
S_56_Prob = histogram(sort(spike_count_56),edge).Values ./ numel(spike_count_56) ;
S_56_Prob(S_56_Prob == 0) = [] ;
entropy_56 = - sum(S_56_Prob .* log2(S_56_Prob)) ;

total = numel(L51) + numel(L52) + numel(L53) + numel(L54) + numel(L55) + numel(L56) ;
H_noise = (entropy_51*numel(L51) + entropy_52*numel(L52) + entropy_53*numel(L53)  ...
         + entropy_54*numel(L54) + entropy_55*numel(L55) + entropy_56*numel(L56)) / total
;


%% Mutual Information
L = [L51;L52;L53;L54;L55;L56];
L = sort(L);
for i=1:numel(L)
    spike_count(1,i) = numel(find(neuron.mgs(L(i),1000:2000) == 1));
end
edge = [ min(spike_count) : (max(spike_count)-min(spike_count))/8: max(spike_count) ] ;
R_prob = histogram(spike_count,edge).Values ./ numel(spike_count) ;
H = - sum(R_prob .* log2(R_prob)) ;
close all

IM_main = H - H_noise

save('IM_main')
```

Q1 - b

1.2.1. File name :

1.b

1.2.2. Figures :

```
IM_main =

    0.5477
```

### 1.2.3. Explanation :

در این قسمت همان مراحل قبلی را تکرار می کنیم ولی به جای 6 تا location، فقط 51 و 54 را درنظر می گیریم و H پاسخ را هم فقط برای trial های این دو location محاسبه می کنیم.

### 1.2.4. Codes :

```
%% initialize
locations = unique(Event.mgs.codes(:,5));
neuron = spike{4, 2} ;

L51 = find(Event.mgs.codes(:,5) == locations(1));
L52 = find(Event.mgs.codes(:,5) == locations(2));
L53 = find(Event.mgs.codes(:,5) == locations(3));
L54 = find(Event.mgs.codes(:,5) == locations(4));
L55 = find(Event.mgs.codes(:,5) == locations(5));
L56 = find(Event.mgs.codes(:,5) == locations(6));

for i=1:numel(L51)
    spike_count_51(1,i) = numel(find(neuron.mgs(L51(i),1000:2000) == 1));
end


for i=1:numel(L54)
    spike_count_54(1,i) = numel(find(neuron.mgs(L54(i),1000:2000) == 1));
end


%%  H noise
% IN
edge = [ min(spike_count_51) : (max(spike_count_51)-min(spike_count_51))/3:
max(spike_count_51) ] ;
S_IN_Prob = histogram(spike_count_51,edge).Values ./ numel(spike_count_51) ;
S_IN_Prob(S_IN_Prob == 0) = [] ;
entropy_IN = - sum(S_IN_Prob .* log2(S_IN_Prob)) ;

% OUT
edge = [ min(spike_count_54) : (max(spike_count_54)-min(spike_count_54))/3:
max(spike_count_54) ] ;
S_OUT_Prob = histogram(spike_count_54,edge).Values ./ numel(spike_count_54) ;
S_OUT_Prob(S_OUT_Prob == 0) = [] ;
entropy_OUT = - sum(S_OUT_Prob .* log2(S_OUT_Prob)) ;

total = numel(L51) + numel(L54) ;
H_noise = entropy_IN * (numel(L51)/total) + entropy_OUT * (numel(L54)/total) ;


%% Mutual Information
L = [L51;L54];
L = sort(L);
for i=1:numel(L)
    spike_count(1,i) = numel(find(neuron.mgs(L(i),1000:2000) == 1));
end
edge = [ min(spike_count) : (max(spike_count)-min(spike_count))/8: max(spike_count) ] ;
R_prob = histogram(spike_count,edge).Values ./ numel(spike_count) ;
H = - sum(R_prob .* log2(R_prob)) ;
close all

IM_main = H - H_noise

save('IM_main')
```

Q1 - c

1.3.1. File name :

———

1.2.2. Figures :

———

1.3.3. Explanation :

در این قسمت می خواهیم ارتباط بین response و stimulus را بررسی کنیم :

قسمت a : در قسمت a ما mutual information را برای هر 6 تا location محاسبه کردیم که عدد 0.57 به دست آمد. برای بررسی صحت این عدد نیاز داریم که آن را با یک حالت که ارتباط بین response و stimulus از بین رفته مقایسه کنیم تا ببینیم ارتباط معنا داری بین این دو حالت وجود دارد یا خیر. برای این کار یک کد دیگر نوشتیم که ابتدا stimulus مربوط به trial ها را به صورت shuffle جا به جا می کند و محاسبات mutual information را دوباره انجام می دهد. اگر این کار را 100 بار تکرار کنیم توزیعی از mutual information ها خواهیم داشت. این توزیع را می توانیم با mutual information اصلی مقایسه آماری کنیم تا ببینیم آیا ارتباط معناداری بین response و stimulus وجود دارد یا خیر. لازمه این موضوع محاسبه p value می باشد. پس از محاسبه آن نتایج زیر حاصل می شود که نشان می دهد فرض صفر رد شده و ارتباط معنادار وجود دارد. پس نتایج ما همانطور که انتظار می رفت اتفاق افتاد.

```
p =

   4.1394e-18


h =

  logical

   1
```

کد a :

```
%% initialize
locations = unique(Event.mgs.codes(:,5));
neuron = spike{4, 2} ;

L51 = find(Event.mgs.codes(:,5) == locations(1));
L52 = find(Event.mgs.codes(:,5) == locations(2));
L53 = find(Event.mgs.codes(:,5) == locations(3));
L54 = find(Event.mgs.codes(:,5) == locations(4));
L55 = find(Event.mgs.codes(:,5) == locations(5));
L56 = find(Event.mgs.codes(:,5) == locations(6));

for k = 1:100
%% test
spikes = neuron.mgs ;
x = randperm(numel(spikes)) ;
spikes = reshape(spikes(x),size(spikes)) ;

%% spike count
for i=1:numel(L51)
    spike_count_51(1,i) = numel(find(spikes(L51(i),1000:2000) == 1));
end

for i=1:numel(L52)
    spike_count_52(1,i) = numel(find(spikes(L52(i),1000:2000) == 1));
end

for i=1:numel(L53)
    spike_count_53(1,i) = numel(find(spikes(L53(i),1000:2000) == 1));
end

for i=1:numel(L54)
    spike_count_54(1,i) = numel(find(spikes(L54(i),1000:2000) == 1));
end

for i=1:numel(L55)
    spike_count_55(1,i) = numel(find(spikes(L55(i),1000:2000) == 1));
end

for i=1:numel(L56)
    spike_count_56(1,i) = numel(find(spikes(L56(i),1000:2000) == 1));
end

%%  H noise
% 51
edge = [ min(spike_count_51) : (max(spike_count_51)-min(spike_count_51))/3:
max(spike_count_51) ] ;
S_51_Prob = histogram(sort(spike_count_51),edge).Values ./ numel(spike_count_51) ;
S_51_Prob(S_51_Prob == 0) = [] ;
entropy_51 = - sum(S_51_Prob .* log2(S_51_Prob)) ;

% 52
edge = [ min(spike_count_52) : (max(spike_count_52)-min(spike_count_52))/3:
max(spike_count_52) ] ;
S_52_Prob = histogram(sort(spike_count_52),edge).Values ./ numel(spike_count_52) ;
S_52_Prob(S_52_Prob == 0) = [] ;
entropy_52 = - sum(S_52_Prob .* log2(S_52_Prob)) ;

% 53
edge = [ 5 : (36-5)/3 : 36 ] ;
S_53_Prob = histogram(sort(spike_count_53),edge).Values ./ numel(spike_count_53) ;
S_53_Prob(S_53_Prob == 0) = [] ;
entropy_53 = - sum(S_53_Prob .* log2(S_53_Prob)) ;

% 54
edge = [ min(spike_count_54) : (max(spike_count_54)-min(spike_count_54))/3:
max(spike_count_54) ] ;
S_54_Prob = histogram(sort(spike_count_54),edge).Values ./ numel(spike_count_54) ;
S_54_Prob(S_54_Prob == 0) = [] ;
entropy_54 = - sum(S_54_Prob .* log2(S_54_Prob)) ;
```

```
% 55
edge = [ min(spike_count_55) : (max(spike_count_55)-min(spike_count_55))/3:
max(spike_count_55) ] ;
S_55_Prob = histogram(sort(spike_count_55),edge).Values ./ numel(spike_count_55) ;
S_55_Prob(S_55_Prob == 0) = [] ;
entropy_55 = - sum(S_55_Prob .* log2(S_55_Prob)) ;

% 56
edge = [ min(spike_count_56) : (max(spike_count_56)-min(spike_count_56))/3:
max(spike_count_56) ] ;
S_56_Prob = histogram(sort(spike_count_56),edge).Values ./ numel(spike_count_56) ;
S_56_Prob(S_56_Prob == 0) = [] ;
entropy_56 = - sum(S_56_Prob .* log2(S_56_Prob)) ;

total = numel(L51) + numel(L52) + numel(L53) + numel(L54) + numel(L55) + numel(L56) ;
H_noise = (entropy_51*numel(L51) + entropy_52*numel(L52) + entropy_53*numel(L53)  ...
         + entropy_54*numel(L54) + entropy_55*numel(L55) + entropy_56*numel(L56)) / total ;


%% Mutual Information
L = [L51;L52;L53;L54;L55;L56];
L = sort(L);
for i=1:numel(L)
    spike_count(1,i) = numel(find(neuron.mgs(L(i),1000:2000) == 1));
end
edge = [ min(spike_count) : (max(spike_count)-min(spike_count))/8: max(spike_count) ] ;
R_prob = histogram(spike_count,edge).Values ./ numel(spike_count) ;
H = - sum(R_prob .* log2(R_prob)) ;
close all


IM_shuffle(1,k) = H - H_noise ;

end

%% compare
load('IM_main')
IM_main = IM_main*ones(1,100) ;

[p,h] = signrank(IM_main,IM_shuffle)
```

قسمت b : در قسمت a ما mutual information را برای دو تا location محاسبه کردیم که عدد 0.54 به دست آمد. مانند قبل، برای بررسی صحت این عدد نیاز داریم که آن را با یک حالت که ارتباط بین response و stimulus از بین رفته مقایسه کنیم تا ببینیم ارتباط معنا داری بین این دو حالت وجود دارد یا خیر. مراحل را مانند قبل انجام دادیم و p value را محاسبه کردیم. این عدد دوباره نشان می دهد که ارتباط بین پاسخ و محرک وجو دارد که همین انتظار را هم داشتیم.

```
p =

   4.2663e-18


h =

  logical

   1
```

کد b :

```
%% initialize
locations = unique(Event.mgs.codes(:,5));
neuron = spike{4, 2} ;

L51 = find(Event.mgs.codes(:,5) == locations(1));
L52 = find(Event.mgs.codes(:,5) == locations(2));
L53 = find(Event.mgs.codes(:,5) == locations(3));
L54 = find(Event.mgs.codes(:,5) == locations(4));
L55 = find(Event.mgs.codes(:,5) == locations(5));
L56 = find(Event.mgs.codes(:,5) == locations(6));

for k = 1:100
%% test
spikes = neuron.mgs ;
x = randperm(numel(spikes)) ;
spikes = reshape(spikes(x),size(spikes)) ;

%% spike count
for i=1:numel(L51)
    spike_count_51(1,i) = numel(find(spikes(L51(i),1000:2000) == 1));
end

for i=1:numel(L54)
    spike_count_54(1,i) = numel(find(spikes(L54(i),1000:2000) == 1));
end

%%  H noise
% IN
edge = [ min(spike_count_51) : (max(spike_count_51)-min(spike_count_51))/3:
max(spike_count_51) ] ;
S_IN_Prob = histogram(spike_count_51,edge).Values ./ numel(spike_count_51) ;
S_IN_Prob(S_IN_Prob == 0) = [] ;
entropy_IN = - sum(S_IN_Prob .* log2(S_IN_Prob)) ;

% OUT
edge = [ min(spike_count_54) : (max(spike_count_54)-min(spike_count_54))/3:
max(spike_count_54) ] ;
S_OUT_Prob = histogram(spike_count_54,edge).Values ./ numel(spike_count_54) ;
S_OUT_Prob(S_OUT_Prob == 0) = [] ;
entropy_OUT = - sum(S_OUT_Prob .* log2(S_OUT_Prob)) ;

total = numel(L51) + numel(L54) ;
H_noise = entropy_IN * (numel(L51)/total) + entropy_OUT * (numel(L54)/total) ;


%% Mutual Information by trails
L = [L51;L54];
L = sort(L);
for i=1:numel(L)
    spike_count(1,i) = numel(find(neuron.mgs(L(i),1000:2000) == 1));
end
edge = [ min(spike_count) : (max(spike_count)-min(spike_count))/8: max(spike_count) ] ;
R_prob = histogram(spike_count,edge).Values ./ numel(spike_count) ;
H = - sum(R_prob .* log2(R_prob)) ;
close all

IM_shuffle(1,k) = H - H_noise ;

end

%% compare
load('IM_main')
IM_main = IM_main*ones(1,100) ;

[p,h] = signrank(IM_main,IM_shuffle)
```

Q3

3.1. File name :

3

3.2. Figures :

```
neuron_number =

     3


p_value_0_1000 =

    0.4799


p_value_1000_2000 =

   5.7610e-09


p_value_2500_3000 =

    0.4448
```

```
neuron_number =

     5


p_value_0_1000 =

    0.8152


p_value_1000_2000 =

   2.3057e-44


p_value_2500_3000 =

    0.0030
```

3.3. Explanation :

در این سوال هدف مقایسه دو موقعیت in و out در سه بازه زمانی می باشد. برای این کار ابتدا مانند قبل trial های مربوط به هر location را جدا می کنیم. سپس برای هر location 3 آرایه می سازیم که در هر کدام spike count مربوط به هر کدام از سه بازه محاسبه می شوند. از بین شش location شماره 51 و 54 را برای مقایسه انتخاب می کنیم. ابتدا آن ها را بصورت متناظر از هم کم می کنیم تا ماتریس اختلاف spike count تشکیل شود. سپس میانگین و واریانس هر یک را حساب کرده و t را محاسبه می کنیم. سپس با توجه به علامت t مقدار p value را به دست می آوریم. این کار را برای هر 3 بازه انجام می دهیم.

این کد را برای چند نورون اجرا کردم و مقادیر متفاوتی بدست آمد که این مقادیر با انتظار ما هم خوانی داشتند. در بیشتر این نتایج p value مربوط به بازه 0 تا 1000 میلی ثانیه بیشتر از 0.05 بود که نشان می دهد تفاوت معنا داری بین دو حالت وجود ندارد، زیرا در فاز fixation هنوز stimulus نداریم و منطقی است که دو موقعیت in و out تفاوت معنا دار نداشته باشند. مقادیر p value برای بازه 1000 تا 2000 میلی ثانیه خیلی کمتر از 0.05 بود که نشان می دهد در این بازه که visual نام دارد و در آن stimulus رخ می دهد، تفاوت معنا داری بین دو condition داریم. و در نهایت نتایج بدست آمده برای p value در بازه 2500 تا 3000 میلی ثانیه را بررسی می کنیم. همان طور که در درس خواندیم این مقادیر برای انواع نورون ها متفاوت است و ممکن است که از 0.05 بیشتر یا کمتر شود.

به عنوان مثال مقادیر را برای نورون 5 و 3 آوردیم.

## 3. 4. Codes :

```
%% initialize
locations = unique(Event.mgs.codes(:,5));
neuron_number = 7
neuron = spike{neuron_number, 1} ;

for i = 1:numel(locations)
    L{i} = find(Event.mgs.codes(:,5) == locations(i));
end

for j = 1:numel(locations)
    for i = 1:numel(L{j})
        spike_1{j}(1,i) = numel(find(neuron.mgs(L{j}(i,1),1:1000) == 1));
        spike_2{j}(1,i) = numel(find(neuron.mgs(L{j}(i,1),1000:2000) == 1));
        spike_3{j}(1,i) = numel(find(neuron.mgs(L{j}(i,1),2500:3000) == 1));
    end
end


%% 0 to 1000
in_condition = spike_1{1} ;
out_condition = spike_1{4} ;

len = min(numel(in_condition),numel(out_condition)) ;
diff(1,1:len) = in_condition(1,1:len) - out_condition(1,1:len) ;
mu = mean(diff) ;
va = sqrt(var(diff)) ;
t = (mu*sqrt(len)) / va ;

if t > 0
    p_value_0_1000 = 2*normcdf(-t)
elseif t < 0
    p_value_0_1000 = 2*normcdf(t)
end
%% 1000 to 2000
in_condition = spike_2{1} ;
out_condition = spike_2{4} ;

len = min(numel(in_condition),numel(out_condition)) ;
diff(1,1:len) = in_condition(1,1:len) - out_condition(1,1:len) ;
mu = mean(diff) ;
va = sqrt(var(diff)) ;
t = (mu*sqrt(len)) / va ;

if t > 0
    p_value_1000_2000 = 2*normcdf(-t)
elseif t < 0
    p_value_1000_2000 = 2*normcdf(t)
end
%% 2500 to 3000
in_condition = spike_3{1} ;
out_condition = spike_3{4} ;

len = min(numel(in_condition),numel(out_condition)) ;
diff(1,1:len) = in_condition(1,1:len) - out_condition(1,1:len) ;
mu = mean(diff) ;
va = sqrt(var(diff)) ;
t = (mu*sqrt(len)) / va ;

if t > 0
    p_value_2500_3000 = 2*normcdf(-t)
elseif t < 0
    p_value_2500_3000 = 2*normcdf(t)
end
```

3.3. Alternative :

در کد دیگر p value را به کمک کد signrank اجرا کردم که کد قبلی را تایید کرد و نتایج مورد انتظار را نتیجه داد :

```
neuron_number =

     4


p =                 p =                 p =

    0.0540              3.9563e-10          0.1235


h =                 h =                 h =

  logical             logical             logical

    0                   1                   0
```