BE 521: Homework 7

p300 Speller

Spring 2020

34 points

Due: 3/31/2020 11:59 PM

Objective: Spell letters using neurosignals

Andrew Clark
Collaborators: Alex Silva, Vishal Tien

P300 Speller

In this homework, you will work with data from a P300-based brain computer interface called BCI2000 (Schalk et al. 2004) that allows people to spell words by focusing their attention on a particular letter displayed on the screen. In each trial the user focused on a letter, and when that letter's row or column is flashed, the user's brain elicits a P300 evoked response. By analyzing whether a P300 signal was produced, across the flashes of many different rows or columns, the computer can determine the letter that the person is focusing on.
Figure 1 shows the letter matrix from one trial of this task.

Data Organization

The data for this homework is stored in `I521_A0008_D001` on the IEEG Portal. The EEG in this dataset were recorded during 85 intended letter spellings. For each letter spelling, 12 row/columns were flashed 15 times in random order ($12 \times 15 = 180$ iterations). The EEG was recorded with a sampling rate of 240 Hz on a 64-channel scalp EEG.

The annotations for this dataset are organized in two layers as follows:

- `TargetLetter` annotation layer indicates the target letter (annotation.description) on which the user was focusing during the recorded EEG segment (annotation.start/annotation.stop). This layer is also provided as TargetLetterAnnots.mat.
- `Stim` annotation layer indicates the row/column that is being flashed (annotation.description) and whether the target letter is contained in that flash (annotation.type). The recorded EEG during that flash is (annotation.start/annotation.stop). Note that this annotation layer is provided as StimAnnots.mat. It is NOT on the portal.

Figure 1: The letter matrix for the P300 speller with the third row illuminated. If the user were focusing on any of the letters in the third row (M, N, O, P, Q, or R), their brain would emit a P300 response. Otherwise it would not.

Figure 2: The row/column indices of the letter matrix, as encoded in the **Stim** annotation layer (annotation.description) matrix.

Figure 3: The scalp EEG 64-channel layout.

Hints: There are many annotations in this dataset and getting them all may take 5-10 minutes. Once you retrieve the annotations once, save them for faster loading in the future. Also, use `{ }` to gather variables across structs for easier manipulation (e.g. `strcmp({annotations.type},'1')`)

Topographic EEG Maps

You can make topographic plots using the provided `topoplotEEG` function. This function needs an "electrode file." and can be called like

```
topoplotEEG(data,'eloc64.txt','gridscale',150)
```

where `data` is the value to plot for each channel. This function plots the electrodes according to the map in Figure 3.

1 Exploring the data

In this section you will explore some basic properties of the data in `I521_A0008_D001`.

1. For channel 11 (Cz), plot the mean EEG for the target and non-target stimuli separately, (i.e. rows/-columns including and not-including the desired character, respectively), on the same set of axes. Label your x-axis in milliseconds. (3 pts)

```
%create session
session = IEEGSession('I521_A0008_D001', 'andrewc', '/Users/andrewclark/Downloads/ieeg_password.bin' );
```

```
Warning: Objects of edu/upenn/cis/db/mefview/services/TimeSeriesDetails class
exist — not clearing java
IEEGSETUP: Found log4j on Java classpath.
URL: https://www.ieeg.org/services
Client user: andrewc
Client password: ****
```

```
%session.data

nr = ceil((session.data.rawChannels(11).get_tsdetails.getEndTime)/1e6*session.data.sampleRate);
channel_11_Cz = session.data.getvalues(1:nr,11);
```

```
durationInUSec_11 = session.data(1).rawChannels(11).get_tsdetails.getDuration;
durationInSec_11 = durationInUSec_11 / 1e6;
```

```
%we need to match when stim and target are the same
non_target_indexes=[];
target_indexes=[];
for i=1:length(Stim)
    if strcmp(Stim(i).type,'0') == 1 %perform the string compare
        non_target_indexes=[non_target_indexes i];
    else
        target_indexes=[target_indexes i];
    end

end
```

```
%overlay each target and non target over each other and do the mean of each column

sample_rate=session.data.sampleRate;% sample rate equals 240 Hz

%time * sample rate (sec * 1/sec)

non_target_mat=zeros(length(non_target_indexes),240);

%pull out the desired data for the non-target data
for i=1:length(non_target_indexes)
    %start of the window in the index space
    window_start=ceil((Stim(non_target_indexes(i)).start)*(1e-6)*(240));

    %stop of the window in index space
    window_stop=ceil((Stim(non_target_indexes(i)).stop)*(1e-6)*(240));
```

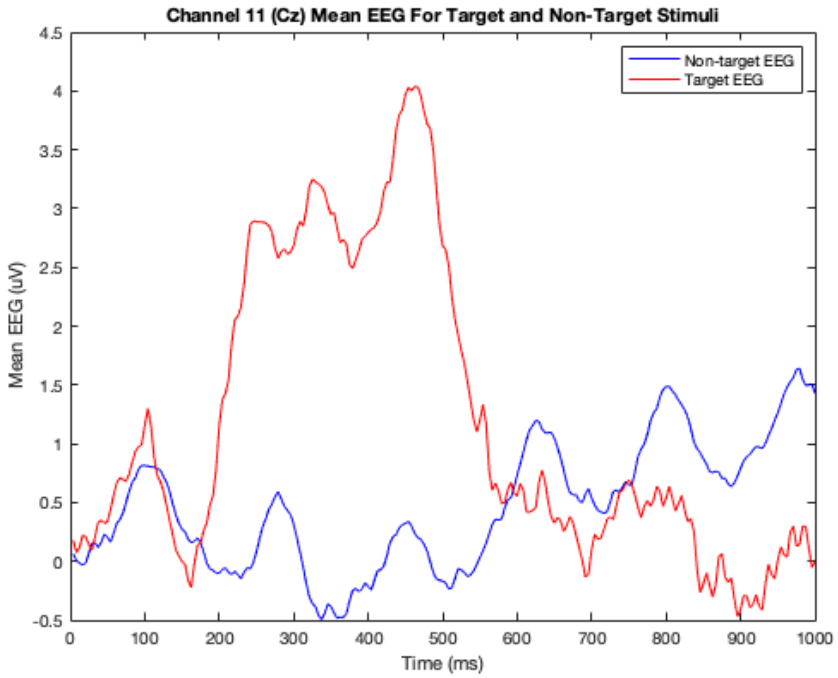
```
    %calculate mean at each interval
    non_target_mat(i,1:240)=channel_11_Cz(window_start:window_stop);
    clear window_start
    clear window_stop
end
```

```
%pull out the desired data for the target data
target_mat=zeros(length(target_indexes),240);

for j=1:length(target_indexes)
    %start of the window in the index space
    window_start_targ=ceil((Stim(target_indexes(j)).start)*(1e-6)*(240));

    %stop of the window in index space
    window_stop_targ=ceil((Stim(target_indexes(j)).stop)*(1e-6)*(240));
    %calculate mean at each interval
    target_mat(j,1:240)=channel_11_Cz(window_start_targ:window_stop_targ);
    clear window_start_targ
    clear window_stop_targ
end
```

```
%take means of each column in both the target and non target matrices
means_non_target=zeros(1,240);
means_target=zeros(1,240);

for i=1:240
    means_non_target(1,i)=mean(non_target_mat(:,i));
    means_target(1,i)=mean(target_mat(:,i));
end
```

```
%create time vector
time_ms=(1/sample_rate:(1/sample_rate):1)*1000; %the length of each trial is one second
```

```
%create figure
figure
plot(time_ms,means_non_target,'b')
hold on
plot(time_ms,means_target,'r')
legend('Non-target EEG','Target EEG')
xlabel('Time (ms)')
ylabel('Mean EEG (uV)')
title('Channel 11 (Cz) Mean EEG For Target and Non-Target Stimuli')
```



2. Repeat the previous questions for channel 23 (Fpz). (1 pts)

```
%read in channel 23 data
nr = ceil((session.data.rawChannels(23).get_tsdetails.getEndTime)/1e6*session.data.sampleRate);
channel_23 = session.data.getvalues(1:nr,23);
```

```
durationInUSec_23 = session.data(1).rawChannels(23).get_tsdetails.getDuration;
durationInSec_23 = durationInUSec_23 / 1e6;
```

```
%we need to match when stim and target are the same
non_target_indexes=[];
target_indexes=[];
for i=1:length(Stim)
    if strcmp(Stim(i).type,'0') == 1 %perform the string compare
        non_target_indexes=[non_target_indexes i];
    else
        target_indexes=[target_indexes i];
    end

end
```

```
%overlay each target and non target over each other and do the mean of each column

sample_rate=session.data.sampleRate;% sample rate equals 240 Hz

%time * sample rate (sec * 1/sec)

non_target_mat=zeros(length(non_target_indexes),240);
```

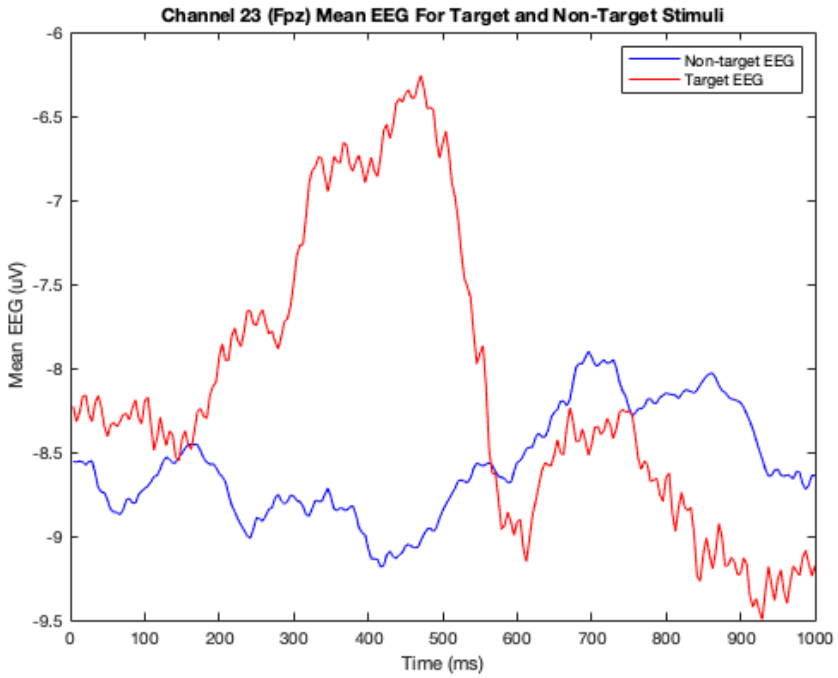
```
%pull out the desired data for the non-target data
for i=1:length(non_target_indexes)
    %start of the window in the index space
    window_start=ceil((Stim(non_target_indexes(i)).start)*(1e-6)*(240));

    %stop of the window in index space
    window_stop=ceil((Stim(non_target_indexes(i)).stop)*(1e-6)*(240));
    %calculate mean at each interval
    non_target_mat(i,1:240)=channel_23(window_start:window_stop);
    clear window_start
    clear window_stop
end
```

```
%pull out the desired data for the target data
target_mat=zeros(length(target_indexes),240);

for j=1:length(target_indexes)
    %start of the window in the index space
    window_start_targ=ceil((Stim(target_indexes(j)).start)*(1e-6)*(240));

    %stop of the window in index space
    window_stop_targ=ceil((Stim(target_indexes(j)).stop)*(1e-6)*(240));
    %calculate mean at each interval
    target_mat(j,1:240)=channel_23(window_start_targ:window_stop_targ);
    clear window_start_targ
    clear window_stop_targ
end
```

```
%take means of each column in both the target and non target matrices
means_non_target=zeros(1,240);
means_target=zeros(1,240);

for i=1:240
    means_non_target(1,i)=mean(non_target_mat(:,i));
    means_target(1,i)=mean(target_mat(:,i));
end
```

```
%create time vector
time_ms=(1/sample_rate:(1/sample_rate):1)*1000; %the length of each trial is one second
```

```
%create figure
figure
plot(time_ms,means_non_target,'b')
hold on
plot(time_ms,means_target,'r')
legend('Non-target EEG','Target EEG')
xlabel('Time (ms)')
ylabel('Mean EEG (uV)')
title('Channel 23 (Fpz) Mean EEG For Target and Non-Target Stimuli')
```



3. Which of the two previous channels looks best for distinguishing between target and non-target stimuli? Which time points look best? Explain in a few sentences. (2 pts)

   Of the two previous channels, Channel 11 (Cz) looks best for distinguishing between target and non-target stimuli. This is because the absolute value of the difference between the mean non-target and mean target EEG on Channel 11 is larger compared to the difference between the mean non-target EEG and mean target EEG on channel 23. The time points that look best for distinguishing between target and non-target stimuli are between 280ms and 500 ms, because there is the greatest difference between mean target EEG and mean non-target EEG during this time period. A greater different in between the mean target EEG and the mean non-target EEG will make it easier to accurately distinguish between the target and non-target signals.

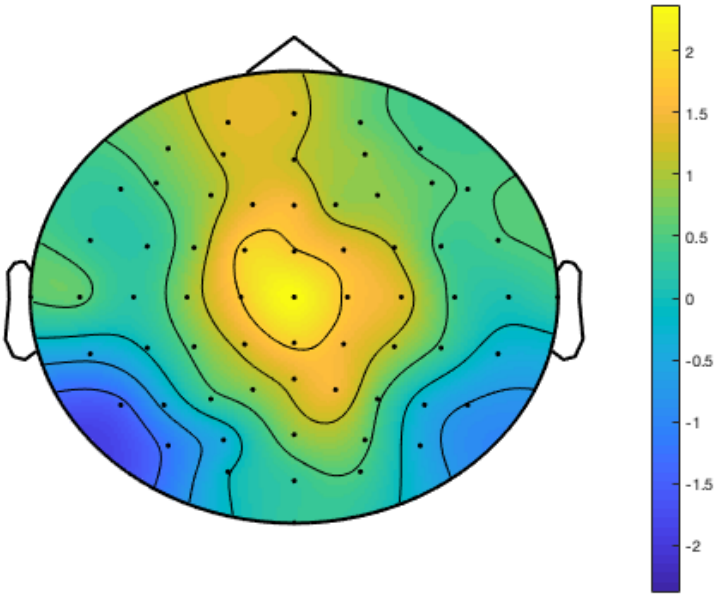
4. Compute the mean difference between the target and non-target stimuli for each channel at timepoint 300 ms averaged across all row/column flashes. Visualize these values using the `topoplotEEG` function. Include a colorbar. (3 pts)

```
%compute mean difference between target and non-target stimuli for each channel at timepoint 300 ms.
%each trial is 1000 ms long.


%all_channels_data=session.data.getvalues(1:nr,1:length(session.data.rawChannels))


%we have already broken up the Stim into target and non-target:
non_target_indexes=[];
target_indexes=[];
for i=1:length(Stim)
    if strcmp(Stim(i).type,'0') == 1 %perform the string compare
        non_target_indexes=[non_target_indexes i];
    else
        target_indexes=[target_indexes i];
    end
```

```
end

%next we need to pull out the data for each trial at the 300ms timepoint
index=240*(300/1000); % this is the 72nd point in each trial

all_channels_target_mat=zeros(1,length(session.data.rawChannels));% this is 1 x 64

%we need to take the mean of all the target data at 300 ms for each trial

%first lets pull out the 300 ms timepoint in each trial for each channel
for i=1:length(session.data.rawChannels)
    channel=session.data.getvalues(1:nr,i);
    target_mat=zeros(length(target_indexes),1);
    for j=1:length(target_indexes)
        %need to use the getvalues method to request data.
        %three_hun_ms_pts=[];
        target_mat(j,1)=channel(ceil((Stim(target_indexes(j)).start)*(1e-6)*(240)+index)-1);
        %take mean of the target_mat column vector and input it into the
        all_channels_target_mat(1,i)=mean(target_mat);

    end
    clear channel
    clear target_mat
end
```

```
all_channels_non_target_mat=zeros(1,length(session.data.rawChannels));% this is 1 x 64


for i=1:length(session.data.rawChannels)
    channel=session.data.getvalues(1:nr,i);
    non_target_mat=zeros(length(non_target_indexes),1);
    for j=1:length(non_target_indexes)
        non_target_mat(j,1)=channel(ceil((Stim(non_target_indexes(j)).start)*(1e-6)*(240)+index)-1);
        %take mean of the target_mat column vector and input it into the
        all_channels_non_target_mat(1,i)=mean(non_target_mat);

    end
    clear channel
    clear non_target_mat
end
```

```
%calculate differences
difference_mat=(all_channels_target_mat-all_channels_non_target_mat);
figure
topoplotEEG(difference_mat,'eloc64.txt','gridscale',150)
colorbar
```



5. How do the red and blue parts of this plot correspond to the plots from above? (2 pts)

   The red parts of this plot correspond to a positive difference (in uV) between the mean target EEG and mean non-target EEG's (those signals are shown above in the plots for parts 1.1 and 1.2 for channels 11 and 23). The blue parts of this plot correspond to a negative difference (in uV) between the mean target EEG and mean non-target EEG (those signals are shown above in the plots for parts 1.1 and 1.2 above for channels 11 and 23).

2 Using Individual P300s in Prediction

Hopefully the Question 1.4 convinced you that the Cz channel is a reasonably good channel to use in separating target from non-target stimuli in the P300. For the rest of the homework, you will work exclusively with this channel.

1. Explain a potential advantage to using just one channel other than the obvious speed of calculation advantage. Explain one disadvantage. (3 pts)

   One potential advantage of using just one channel other than the obvious speed of calculation is that there may be differences in the quality of the physical hardware of each of the channels or how well each channel is adhered to the subject, so picking one that is very good representation (due to a hardware problem, placement on the patient, etc.) of the behavior of the EEG signals collected may yield better results (since sub-optimal channels are not used and thus cannot distort the dataset). One disadvantage of using just one channel is that if the channel used is not a good representation of the EEG signals that should be collected it could distort the dataset alot. Additionally, it could be the case that there are channels besides the one chosen that have features not neccessarily picked up by the channel chosen, and as such using just one channel could lead to missing out on collected information.

2. One simple way of identifying a P300 in a single trial (which we'll call the *p300 score*) is to take the mean EEG from 250 to 450 ms and then subtract from it the mean EEG from 600 to 700 ms. What is the *p300 score* for epoch (letter) 10, iteration 11 at electrode Cz? (3 pts)

```
%only working with channel\_11\_Cz here
epoch=TargetLetter(10);% this is the epoch we are being asked to look at
Stim_start_index=180*9;
Stim_end_index=180*10;
%narrow data down to the epoch we would like
epoch_data=channel_11_Cz((Stim(Stim_start_index).stop*(240/1e6)):(Stim(Stim_end_index).stop*(240/1e6)));

%now we need to narrow down to the 11th flash within epoch_data
%we want Stim(1631).start to Stim(1631).stop => this will be 240 data
%points long (1 second long)
flash_data=channel_11_Cz(ceil((Stim(1631).start*(240/1e6))):ceil((Stim(1631).stop*(240/1e6))));

%now use flash data to get the
p_300_score=mean(flash_data(((250/1000)*240):((450/1000)*240)))-mean(flash_data(((600/1000)*240):((700/1000)
```

```
p_300_score =

    1.4511
```

As computed by the code above the *p300 score* for epoch 10, iteration 11 at electrode Cz is equal to 1.4511 uV.

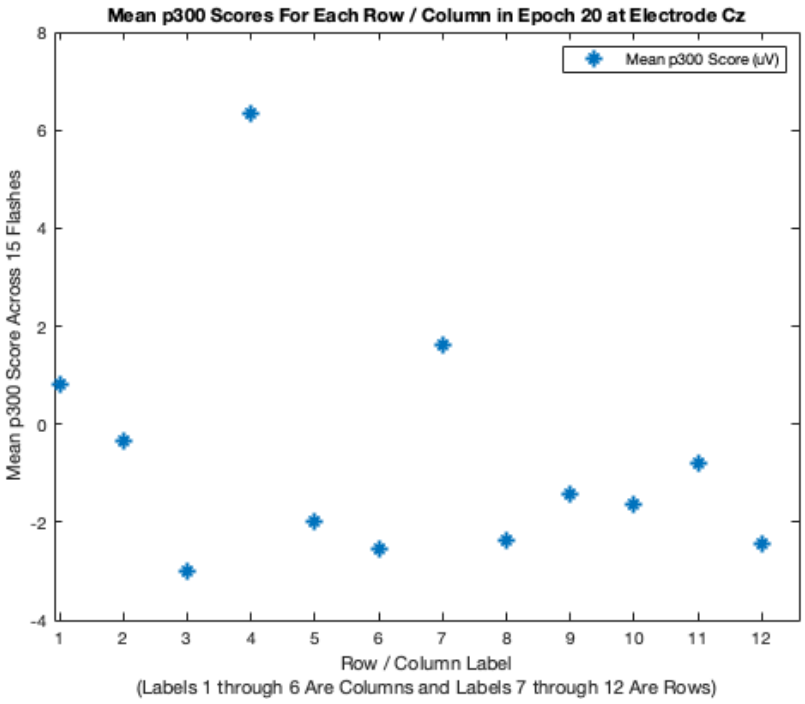
3. Plot the *p300 scores* for each row/column in epoch 20 at electrode Cz. (3 pts)

```
%plot p 300 score for each row/column in epoch 20 at electrode Cz
target_letter=TargetLetter(20).description; % is 'D'

epoch_20_start_idx=(180*19+1);%.start of this. this equals 3421
epoch_20_stop_idx=(180*20);% .stop of this. this equlas 3600

epoch_20_Cz_data=channel_11_Cz(ceil((Stim(epoch_20_start_idx).start*(240/1e6))):ceil((Stim(epoch_20_stop_idx

%initialize the p_300 score matrix. we will take the mean of each column in order to obtain the p_300 score
p_300_epoch_20_mat=zeros(15,12);

%we will compute 180 p_300 scores and fill in the matrix with them

 %compute p 300 score. we need to take the mean eeg from 600 to 700
        %ms and subtract the mean eeg from 250 to 450 ms for each trial in
        %the epoch data. but we need to place these scores correctly in our
        %matrix (fill each row in the matrix with scores
        for j=1:15
            %just compute the first row in the matrix firs
            clear trial
            p_300_list=zeros(1,12);
            for i=1:12

                trial=channel_11_Cz(ceil((Stim(epoch_20_start_idx).start*(240/1e6))):ceil((Stim(epoch_20_sta

                if strcmp(Stim(epoch_20_start_idx).description,'1')==1
                    %channel_11_Cz(ceil((Stim(1631).start*(240/1e6))):ceil((Stim(1631).stop*(240/1e6))));
                    p_300_list(1)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240)
                    %repeat for each of the twelve different rows / columns
                elseif strcmp(Stim(epoch_20_start_idx).description,'2')==1
                    p_300_list(2)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240)

                elseif strcmp(Stim(epoch_20_start_idx).description,'3')==1
                    p_300_list(3)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240)

                elseif strcmp(Stim(epoch_20_start_idx).description,'4')==1
```

```
                    p_300_list(4)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240)

                elseif strcmp(Stim(epoch_20_start_idx).description,'5')==1
                    p_300_list(5)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240)

                elseif strcmp(Stim(epoch_20_start_idx).description,'6')==1
                    p_300_list(6)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240)

                elseif strcmp(Stim(epoch_20_start_idx).description,'7')==1
                    p_300_list(7)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240)

                elseif strcmp(Stim(epoch_20_start_idx).description,'8')==1
                    p_300_list(8)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240)

                elseif strcmp(Stim(epoch_20_start_idx).description,'9')==1
                    p_300_list(9)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240)

                elseif strcmp(Stim(epoch_20_start_idx).description,'10')==1
                    p_300_list(10)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240

                elseif strcmp(Stim(epoch_20_start_idx).description,'11')==1
                    p_300_list(11)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240

                elseif strcmp(Stim(epoch_20_start_idx).description,'12')==1
                    p_300_list(12)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240
                end
                epoch_20_start_idx=epoch_20_start_idx+1;
            end
            p_300_epoch_20_mat(j,1:12)=p_300_list;
        end
```

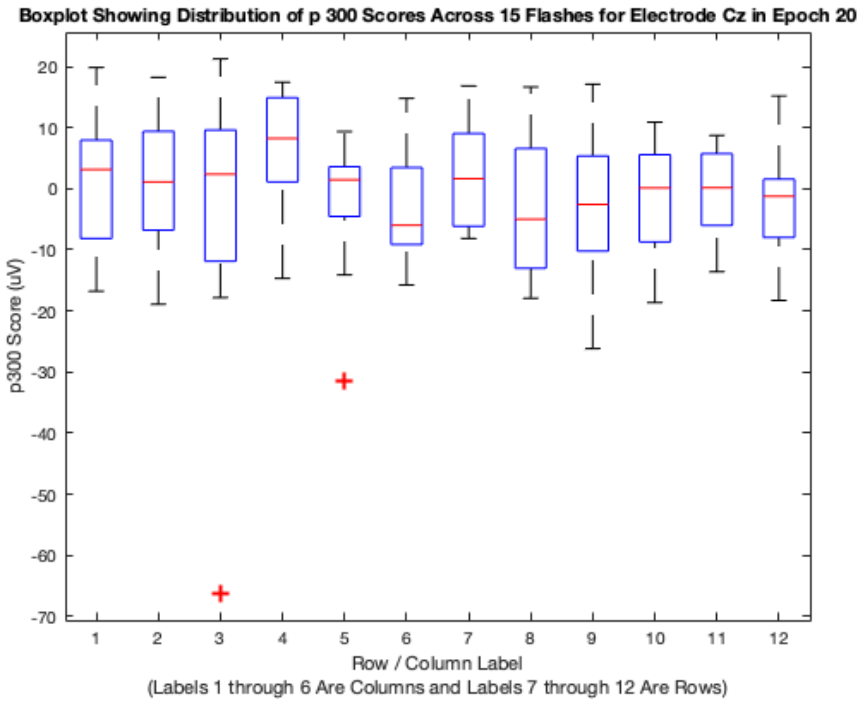
```
%create figure with mean of each column
mean_p_300_list=[];
for k=1:12
    mean_p_300_list(k)=mean(p_300_epoch_20_mat(:,k));
end
```

```
%create mean plot
x=1:12:144;

figure
plot(x,mean_p_300_list,'*')% gives D as the highest mean score which is good
legend('Mean p300 Score (uV)')
xlabel({'Row / Column Label'; '(Labels 1 through 6 Are Columns and Labels 7 through 12 Are Rows)'})
xticks(1:12:144)
xticklabels({'1','2','3','4','5','6','7','8','9','10','11','12'})
ylabel('Mean p300 Score Across 15 Flashes')
title('Mean p300 Scores For Each Row / Column in Epoch 20 at Electrode Cz')
```



Plot showing the mean Mean p300 Scores (in uV) For Each Row / Column in Epoch 20 at Electrode Cz.

```
%create box plot

%create grouping variables. one for each column
g1=repmat({'1'},1,1);
g2=repmat({'2'},1,1);
g3=repmat({'3'},1,1);
g4=repmat({'4'},1,1);
g5=repmat({'5'},1,1);
g6=repmat({'6'},1,1);
g7=repmat({'7'},1,1);
g8=repmat({'8'},1,1);
g9=repmat({'9'},1,1);
g10=repmat({'10'},1,1);
g11=repmat({'11'},1,1);
g12=repmat({'12'},1,1);
g=[g1;g2;g3;g4;g5;g6;g7;g8;g9;g10;g11;g12];

figure
boxplot(p_300_epoch_20_mat,g)
xlabel({'Row / Column Label'; '(Labels 1 through 6 Are Columns and Labels 7 through 12 Are Rows)'})
ylabel('p300 Score (uV)')
title('Boxplot Showing Distribution of p 300 Scores Across 15 Flashes for Electrode Cz in Epoch 20')
```



Box plot showing distribution and median p300 Scores (in uV) For Each Row / Column in Epoch 20 at Electrode Cz (TA Piazza posts said it was fine to plot either the mean p300 scores or do a box plot. Both are shown here).

4. Based on your previous answer for epoch 20, what letter do you predict the person saw? Is this prediction correct? (2 pts)

   As shown in the plot showing the mean p300 scores for all rows / columns, the column with the highest mean p300 score was labeled 4 and the row with the highest mean p300 score was labeled 7, so based on my previous answer for epoch 20 I predict the letter the person saw was 'D'. This prediction is correct.

5. Using this *p300 score*, predict (and print out) the letter viewed at every epoch. What was you prediction accuracy? (2 pts)

```
%create matrix for each epoch. Then do the mean of each column and save to list and then find the
%max of the first 6 elements in the list and the max of the last 6 elements
%in the list and save the positions of those maxes to a matrix. iterate
%through matrix and print out letters with conditionals
epoch_20_start_idx=1;

%initalize matrix for storing max average p 300 scores. first column is the
% the row index. second column is the column index
max_p_300_labels=zeros(85,3);


for y=1:length(TargetLetter)%iterate through each epoch
    p_300_epoch_20_mat=zeros(15,12); %initialize p_300 matrix for each epoch

    %we will compute 180 p_300 scores and fill in the matrix with them

    %compute p 300 score. we need to take the mean eeg from 600 to 700
    %ms and subtract the mean eeg from 250 to 450 ms for each trial in
```

```
%the epoch data. but we need to place these scores correctly in our
%matrix (fill each row in the matrix with scores

for j=1:15%iterate through each flash
    %just compute the first row in the matrix firs
    clear trial
    p_300_list=zeros(1,12);
    for i=1:12 %iterate through each row / column in each flash

        %load in the individual flash (trial) data
        trial=channel_11_Cz(ceil((Stim(epoch_20_start_idx).start*(240/1e6))):ceil((Stim(epoch_20_start_i

        %compute the p300 score after sorting into appropriate place in
        %the p 300 list
        if strcmp(Stim(epoch_20_start_idx).description,'1')==1
            %channel_11_Cz(ceil((Stim(1631).start*(240/1e6))):ceil((Stim(1631).stop*(240/1e6))));
            p_300_list(1)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7
            %repeat for each of the twelve different rows / columns
        elseif strcmp(Stim(epoch_20_start_idx).description,'2')==1
            p_300_list(2)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7

        elseif strcmp(Stim(epoch_20_start_idx).description,'3')==1
            p_300_list(3)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7

        elseif strcmp(Stim(epoch_20_start_idx).description,'4')==1
            p_300_list(4)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7

        elseif strcmp(Stim(epoch_20_start_idx).description,'5')==1
            p_300_list(5)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7

        elseif strcmp(Stim(epoch_20_start_idx).description,'6')==1
            p_300_list(6)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7

        elseif strcmp(Stim(epoch_20_start_idx).description,'7')==1
            p_300_list(7)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7

        elseif strcmp(Stim(epoch_20_start_idx).description,'8')==1
            p_300_list(8)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7

        elseif strcmp(Stim(epoch_20_start_idx).description,'9')==1
            p_300_list(9)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7

        elseif strcmp(Stim(epoch_20_start_idx).description,'10')==1
            p_300_list(10)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((

        elseif strcmp(Stim(epoch_20_start_idx).description,'11')==1
            p_300_list(11)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((

        elseif strcmp(Stim(epoch_20_start_idx).description,'12')==1
            p_300_list(12)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((
        end
        epoch_20_start_idx=epoch_20_start_idx+1;
    end
    %fill in each row in the p_300 epoch matrix with the the p 300 scores
    p_300_epoch_20_mat(j,1:12)=p_300_list;
end
%compute mean
mean_p_300_list=[];
for k=1:12
    mean_p_300_list(k)=mean(p_300_epoch_20_mat(:,k));
end
%find the 2 maximums of the list and store them
max_row=0;
max_col=0;
max_col=max(mean_p_300_list(1:6));
```

```
    max_row=max(mean_p_300_list(7:12));
    max_p_300_labels(y,1)=find(mean_p_300_list==max_row);%first column in matrix has the rows
    max_p_300_labels(y,2)=find(mean_p_300_list==max_col);%second column in matrix has the columns
end
```

```
%create labels cell array
labels=cell(1,85);

for i=1:length(max_p_300_labels)
    if max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'A';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'B';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 3
        labels{i}= 'C';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 4
        labels{i}= 'D';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 5
         labels{i}= 'E';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 6
        labels{i}= 'F';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'G';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'H';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 3
        labels{i}= 'I';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 4
        labels{i}= 'J';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 5
        labels{i}= 'K';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 6
        labels{i}= 'L';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'M';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'N';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 3
        labels{i}= 'O';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 4
        labels{i}= 'P';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 5
        labels{i}= 'Q';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 6
        labels{i}= 'R';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'S';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'T';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 3
        labels{i}= 'U';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 4
        labels{i}= 'V';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 5
        labels{i}= 'W';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 6
        labels{i}= 'X';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'Y';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'Z';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 3
        labels{i}= '1';
```

```
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 4
        labels{i}= '2';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 5
        labels{i}= '3';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 6
        labels{i}= '4';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 1
        labels{i}= '5';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 2
        labels{i}= '6';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 3
        labels{i}= '7';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 4
        labels{i}= '8';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 5
        labels{i}= '9';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 6
        labels{i}= '_';
    end
end

%Now calculate the accuray
num_correct=0;
for i=1:length(labels)
    if strcmp(TargetLetter(i).description,labels{i}) ==1
        num_correct=num_correct+1;
    end
end

%print out the predicted letters
for i=1:length(labels)
    disp(labels(i))
end

%comput prediction accuracy
prediction_accuracy=num_correct/length(labels)*100
```

```
    'A'

    'P'

    'V'

    '5'

    'U'

    'F'

    'O'

    'H'

    'I'

    'O'

    'P'

    'B'

    'R'
```

```
'J'
'M'
'F'
'C'
'E'
'D'
'-'
'C'
'T'
'W'
'I'
'D'
'B'
'P'
'C'
'O'
'A'
'W'
'R'
'5'
'I'
'U'
'K'
'O'
'L'
'3'
'O'
'O'
'E'
'W'
'E'
'6'
```

```
'T'
'G'
'K'
'4'
'R'
'Z'
'3'
'O'
'-'
'H'
'Y'
'6'
'O'
'-'
'E'
'X'
'E'
'K'
'V'
'L'
'N'
'X'
'T'
'K'
'K'
'P'
'I'
'H'
'N'
'T'
'F'
'X'
'X'
```

```
    'L'
    'O'
    'T'
    'B'
    'Q'
    'R'

prediction_accuracy =
   34.1176
```

As calculated by the code above the prediction accuracy is equal to 34.1176 percent. The predicted letters viewed at each epoch are also shown above (in the same order as the epochs).

3 Automating the Learning

In Section 2, you used a fairly manual method for predicting the letter. Here, you will have free rein to use put any and all learning techniques to try to improve your testing accuracy.

1. Play around with some ideas for improving/generalizing the prediction paradigm used in the letter prediction. Use the first 50 letter epochs as the training set and the later 35 for validation. Here, you are welcome to hard-code in whatever parameters you like/determine to be optimal. What is the optimal validation accuracy you get? Note: don't worry too much about accuracy, we are more interested in your thought process. (4 pts)

Train the model

```
p300_score_mat_ch_11=zeros(180*50, 3); %set up the matrix

%area feature function
A = @(x) sum(x);
%caculate p300 scores and area
for i=1:180*50 %only train on first
    trial=channel_11_Cz(ceil((Stim(i).start*(240/1e6))):ceil((Stim(i).stop*(240/1e6))));%this is 240 datapoi
    p300_score_mat_ch_11(i,1)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240):((7
    %compute area feature over 250 ms to 450 ms
    p300_score_mat_ch_11(i,2)=A(trial(((250/1000)*240):((450/1000)*240)));

end

for i=1:(180*50)
    if strcmp((Stim(i).type),'1')==1
        p300_score_mat_ch_11(i,3)=2;%2 is target
    else
        p300_score_mat_ch_11(i,3)=1; % 1 is non target
    end

end

%normalize both features
```

```
%for i=1:180*50
p300_score_mat_ch_11(:,1)=(p300_score_mat_ch_11(:,1)-mean(p300_score_mat_ch_11(:,1)))/std(p300_score_mat_ch_
p300_score_mat_ch_11(:,2)=(p300_score_mat_ch_11(:,2)-mean(p300_score_mat_ch_11(:,2)))/std(p300_score_mat_ch_
%end

%train logistic regression classifier to get probabilities of each stim
B=mnrfit(p300_score_mat_ch_11(:,1:2),p300_score_mat_ch_11(:,3));

%svm_mdl=fitcsvm(p300_score_mat_ch_11(:,1:2),p300_score_mat_ch_11(:,3),'KernelFunction','RBF');

%pihat=mnrval(B,p300_score_mat_ch_11(:,1:2)); %first column is 1 probability second column is 2 probability

% pihat_percentage=predict(svm_mdl, p300_score_mat_ch_11_test(:,1:2));
%pihat_percentage=pihat*100;
```

Test the Model

```
p300_score_mat_ch_11_test=zeros(180*35, 3); %set up the matrix
%index=1;

%area feature function
A = @(x) sum(x);
%caculate p300 scores and area
for i=((50*180)+1):length(Stim) %compute testing features
    trial=channel_11_Cz(ceil((Stim(i).start*(240/1e6))):ceil((Stim(i).stop*(240/1e6))));%this is 240 datapoi
    p300_score_mat_ch_11_test(i,1)=mean(trial(((250/1000)*240):((450/1000)*240)))-mean(trial(((600/1000)*240
    %compute area feature over 250 ms to 450 ms
    p300_score_mat_ch_11_test(i,2)=A(trial(((250/1000)*240):((450/1000)*240)));

end

p300_score_mat_ch_11_test=p300_score_mat_ch_11_test(9001:15300,:);

%normalize the p_300 scores and area
p300_score_mat_ch_11_test(:,1)=(p300_score_mat_ch_11_test(:,1)-mean(p300_score_mat_ch_11_test(:,1)))/std(p30
p300_score_mat_ch_11_test(:,2)=(p300_score_mat_ch_11_test(:,2)-mean(p300_score_mat_ch_11_test(:,2)))/std(p30
```

```
pihat=mnrval(B,p300_score_mat_ch_11_test(:,1:2)); %first column is 1 probability second column is 2 probabil

pihat_percentage=pihat*100; %we will use column two of this matrix to do the letter prediction
```

```
%label data by row and column
test_pihat_percentage=[pihat_percentage(:,2), zeros(length(pihat_percentage),1)];
```

```
%assign numbers to the third column
for i=9001:length(Stim)
    test_pihat_percentage(i-9000,2)=str2num(Stim(i).description);
end

i=1;
epoch=test_pihat_percentage(1:180,1);

index_mat=zeros(15,12);
for i=1:length(test_pihat_percentage)
    if test_pihat_percentage(i,2) ==1
```

```
        index_mat(i,1)=i;
    elseif test_pihat_percentage(i,2) ==2
        index_mat(i,2)=i;
    elseif test_pihat_percentage(i,2) ==3
        index_mat(i,3)=i;
    elseif test_pihat_percentage(i,2) ==4
        index_mat(i,4)=i;
    elseif test_pihat_percentage(i,2) ==5
        index_mat(i,5)=i;
    elseif test_pihat_percentage(i,2) ==6
        index_mat(i,6)=i;
    elseif test_pihat_percentage(i,2) ==7
        index_mat(i,7)=i;
    elseif test_pihat_percentage(i,2) ==8
        index_mat(i,8)=i;
    elseif test_pihat_percentage(i,2) ==9
        index_mat(i,9)=i;
    elseif test_pihat_percentage(i,2) ==10
        index_mat(i,10)=i;
    elseif test_pihat_percentage(i,2) ==11
        index_mat(i,11)=i;
    elseif test_pihat_percentage(i,2) ==12
        index_mat(i,12)=i;
    end
    %index_mat=nonzeros(index_mat);

end

%test=index_mat(index_mat~=0)
actual_index_mat=zeros(525,12);
for i=1:12
    col=index_mat(:,i);
    actual_index_mat(:,i)=col(col~=0);
end

prob_mat_final=zeros(size(actual_index_mat));
per=test_pihat_percentage(:,1);
prob_mat_final=per(actual_index_mat);
```

```
%now calculate average probabilities for each 15 row increment in the
%prob_mat_final

positions_mat=zeros(35,2);
starter_idx=1;
ender_idx=15;
for i=1:35
    prob_mat_to_avg=prob_mat_final(starter_idx:ender_idx,:);

    %compute mean of columns
    mean_p_300_col_list=[];
    for k=1:6%columns are 1 to 6
        mean_p_300_col_list(k)=mean(prob_mat_to_avg(:,k));
    end

%compute mean of the rows
    mean_p_300_row_list=[];
    for k=7:12%columns are 1 to 6
        mean_p_300_row_list(k)=mean(prob_mat_to_avg(:,k));
    end
%find the 2 maximums of the list and store them
%max_row=0;
%max_col=0;
    max_col=max(mean_p_300_col_list(1:6));
```

```
    max_row=max(mean_p_300_row_list(7:12));
    positions_mat(i,1)=find(mean_p_300_row_list==max_row);%first column in matrix has the rows
    positions_mat(i,2)=find(mean_p_300_col_list==max_col);%second column is rows
    starter_idx=starter_idx+15;
    ender_idx=ender_idx+15;
end
```

```
max_p_300_labels=positions_mat;
```

```
%now predict letter
labels=cell(1,35);

for i=1:length(max_p_300_labels)
    if max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'A';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'B';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 3
        labels{i}= 'C';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 4
        labels{i}= 'D';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 5
         labels{i}= 'E';
    elseif max_p_300_labels(i,1) == 7 &&  max_p_300_labels(i,2) == 6
        labels{i}= 'F';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'G';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'H';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 3
        labels{i}= 'I';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 4
        labels{i}= 'J';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 5
        labels{i}= 'K';
    elseif max_p_300_labels(i,1) == 8 &&  max_p_300_labels(i,2) == 6
        labels{i}= 'L';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'M';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'N';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 3
        labels{i}= 'O';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 4
        labels{i}= 'P';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 5
        labels{i}= 'Q';
    elseif max_p_300_labels(i,1) == 9 &&  max_p_300_labels(i,2) == 6
        labels{i}= 'R';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'S';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'T';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 3
        labels{i}= 'U';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 4
        labels{i}= 'V';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 5
        labels{i}= 'W';
    elseif max_p_300_labels(i,1) == 10 &&  max_p_300_labels(i,2) == 6
        labels{i}= 'X';
```

```
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 1
        labels{i}= 'Y';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 2
        labels{i}= 'Z';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 3
        labels{i}= '1';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 4
        labels{i}= '2';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 5
        labels{i}= '3';
    elseif max_p_300_labels(i,1) == 11 &&  max_p_300_labels(i,2) == 6
        labels{i}= '4';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 1
        labels{i}= '5';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 2
        labels{i}= '6';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 3
        labels{i}= '7';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 4
        labels{i}= '8';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 5
        labels{i}= '9';
    elseif max_p_300_labels(i,1) == 12 &&  max_p_300_labels(i,2) == 6
        labels{i}= '_';
    end
end


%Now calculate the accuray
num_correct=0;
for j=51:85
    if strcmp(TargetLetter(j).description,labels{j-50})==1
        num_correct=num_correct+1;
    end
end

%print out the predicted letters
for i=1:length(labels)
    disp(labels(i))
end

%compute prediction accuracy
prediction_accuracy=num_correct/length(labels)*100
```

```
    'Z'

    '3'

    'M'

    'L'

    'H'

    'Y'

    '6'

    'Q'

    '_'
```

```
    'W'
    'X'
    'E'
    '9'
    'V'
    'L'
    'R'
    'X'
    'X'
    'K'
    'K'
    'P'
    'I'
    'H'
    'N'
    'H'
    'F'
    'X'
    'X'
    'I'
    'O'
    'T'
    'H'
    'Q'
    'L'

prediction_accuracy =

    40
```

As computed by the code above the optimal validation accuracy obtained is 40 percent. The predicted letters are also outputted above in the same order as the order of the epochs.

2. Describe your algorithm in detail. Also describe what you tried that didn't work. (6 pts)

   The algorithm above has a prediction accuracy of 40 percent in the testing dataset, which is an improvement over the 34.1176 percent that was achieved in part 2.5. The algorithm works by first computing two features, p300 score and area, for each and every flash in the training dataset (there are 9000 flashes total in the training dataset). The p300 score for each flash is defined the same as in

part 2.2, i.e. the mean EEG from 250 to 450 ms is subtracted from the mean EEG from 600 to 700 ms, and the area feature is defined simply as the sum of the datapoints in the flash from 250 to 450 ms. The algorithm then labels each of these pairs of features as either target or non-target (based on the Stim annotations.type) and then normalizes the features. Next, a multinomial logistic regression classifer is trained using the two features and the assigned labels. After the training is complete, the same two normalized features described above are computed for the testing dataset (which consists of the last 35 epochs). The created logistic regression model is then run on the testing data, yielding a probability that each of the flashes in the testing data was either on-target or off-target. The 6300 on-target probabilities (one for each flash in the last 35 epochs) are then sorted into 35 15 x 12 matrices, with each matrix representing a single epoch, and the mean of each of the 12 columns in this matrix are taken (these means represent the average probabilites that a respective row or column is on-target for a single epoch). The position of the maximum mean computed for each row and column is then calculated, yielding a (row, column) tuple for each of the 35 epochs. This (row,column) tuple is then mapped to a letter using conditionals. These predicted letters (which are also shown in the code above) are then compared to the letters provided by the TargetLetter.descriptions in the annotation file and a prediction accuracy is computed.

In terms of what did not work as well, I tried the same method as above but with the area feature defined not as the sum of all points over an interval, but as the absolute value of the sum of all points over an interval and it did not work as well, with a prediction accuracy of only 31.4286 percent in the testing data. Additionally, I tried just using the p300 feature to train the logistic regression classifier and this did did not work as well as the final method used, with a prediction accuracy of 34.2857 percent in the testing dataset. I also tried just using the area feature (as defined in the optimal method) to train the logistic regression model and this also did not work as well as both features used with a a prediction accuracy of 37.1429 percent in the testing dataset. I also tried adding another feature, the area between 600 and 700 ms to the optimal method, however that also yielded a prediction accuracy of 40 percent (it encoded almost the exact same information). I also tried zero crossings about the mean (instead of area) with p300 score and a logistic regression but that only yielded a prediction accuracy of 37.1429 percent. Using either line length or energy instead of area with a logistic regression yielded a prediction accuracy of only 20 percent (for each of these). Additionally, I tried to use an SVM model with an RBF kernel function to classify the data into target and non-target spaces (with the intent of sorting each of the 1s and 2s outputted into 35 15 x 12 matrices and taking the highest average of each of the columns, which is the same strategy as the optimal one), however every data point was classified as a 1 (symbolizing non-target) so this model did not work.