BE 521: Homework 7

p300 Speller

Spring 2020

34 points

Due: 3/31/2020 11:59 PM

Objective: Spell letters using neurosignals

John Talley
Collaborators: Joseph Iwasyk

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)

Load Data

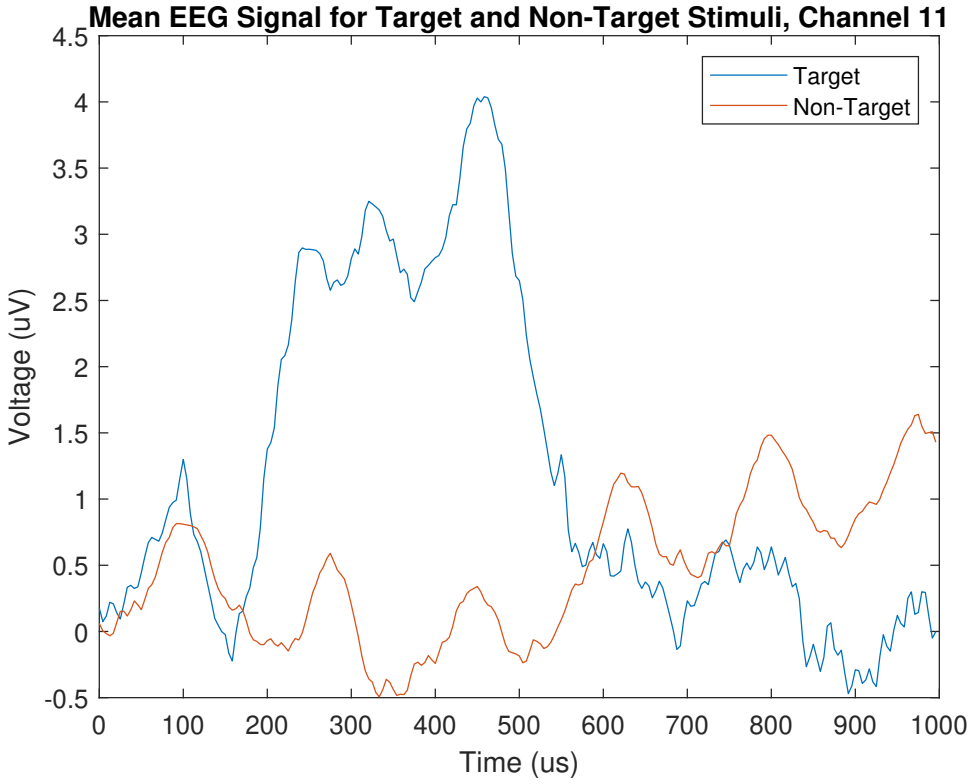
```
dataset_ID = 'I521_A0008_D001';
ieeg_id = 'jtalley';
ieeg_pw = 'jta_ieeglogin.bin';

session = IEEGSession(dataset_ID,ieeg_id,ieeg_pw);

load('StimAnnots.mat');
load('TargetLetterAnnots.mat');


durInUSec = session.data.rawChannels(1).get_tsdetails.getDuration;
durInSec = durInUSec/1e6;

Fs  = session.data.sampleRate;
```

```
IEEGSETUP: Found log4j on Java classpath.
URL: https://www.ieeg.org/services
Client user: jtalley
Client password: ****
```

Channel 11

```
channel_11 = getvalues(session.data,1:durInSec*Fs + 2,11);

stim_type = strcmp({Stim.type}, '1');
% Stim(1).start - Stim(1).end = 1000000
% each stimulation is 1 second long, 240 samples

targetEEG11 = zeros(ceil(durInSec), Fs);
nonTargetEEG11 = zeros(ceil(durInSec), Fs);

targetIdx = find(stim_type == 1);
nonTargetIdx = find(stim_type == 0);

for i = 1:durInSec
    if stim_type(i) == 1
        targetEEG11(i,:) = channel_11(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
    else
        nonTargetEEG11(i,:) = channel_11(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
    end
end

targetEEG11 = targetEEG11(targetIdx,:);
targetEEGAvg11 = mean(targetEEG11);
```

```
nonTargetEEG11 = nonTargetEEG11(nonTargetIdx,:);
nonTargetEEGAvg11 = mean(nonTargetEEG11);


t = 1/Fs:1000/Fs:1000;

figure;
plot(t, targetEEGAvg11);
hold on;
plot(t, nonTargetEEGAvg11);
xlabel('Time (us)');
ylabel('Voltage (uV)');
title('Mean EEG Signal for Target and Non-Target Stimuli, Channel 11');
legend('Target', 'Non-Target');
```

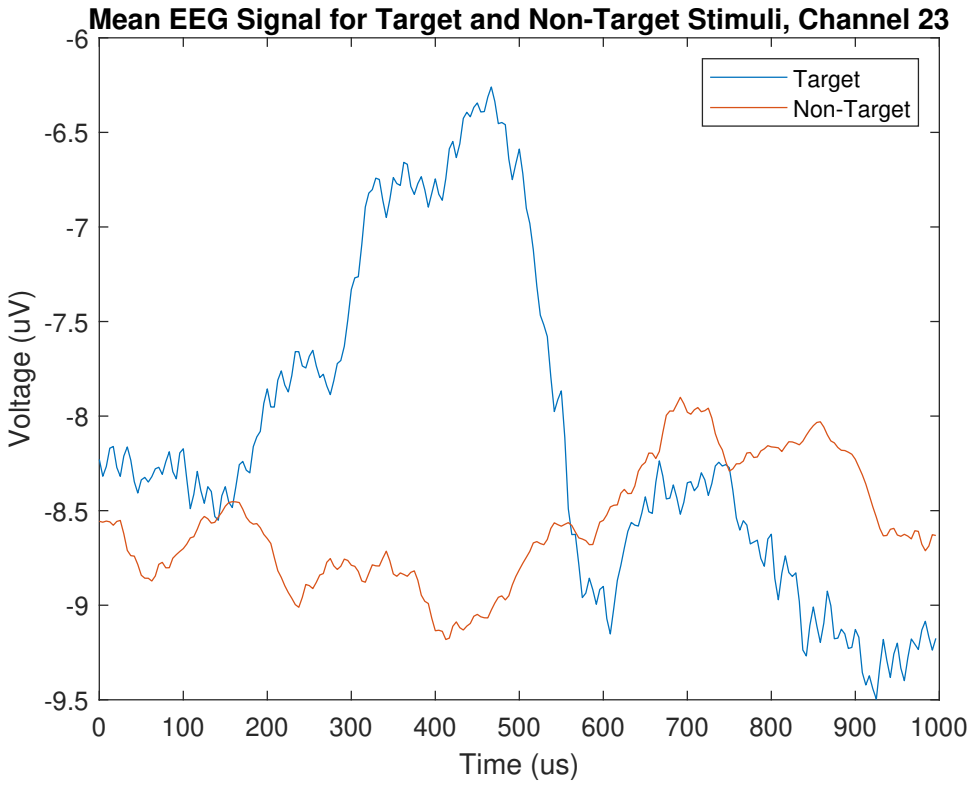


2. Repeat the previous questions for channel 23 (Fpz). (1 pts)

Channel 23

```
channel_23 = getvalues(session.data,1:durInSec*Fs + 2,23);

stim_type = strcmp({Stim.type}, '1');
% Stim(1).start - Stim(1).end = 1000000
% each stimulation is 1 second long, 240 samples

targetEEG23 = zeros(ceil(durInSec), Fs);
nonTargetEEG23 = zeros(ceil(durInSec), Fs);

targetIdx = find(stim_type == 1);
```

```
nonTargetIdx = find(stim_type == 0);

for i = 1:durInSec
    if stim_type(i) == 1
        targetEEG23(i,:) = channel_23(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
    else
        nonTargetEEG23(i,:) = channel_23(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
    end
end

targetEEG23 = targetEEG23(targetIdx,:);
targetEEGAvg23 = mean(targetEEG23);

nonTargetEEG23 = nonTargetEEG23(nonTargetIdx,:);
nonTargetEEGAvg23 = mean(nonTargetEEG23);


t = 1/Fs:1000/Fs:1000;

figure;
plot(t, targetEEGAvg23);
hold on;
plot(t, nonTargetEEGAvg23);
xlabel('Time (us)');
ylabel('Voltage (uV)');
title('Mean EEG Signal for Target and Non-Target Stimuli, Channel 23');
legend('Target', 'Non-Target');
```



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)

   Channel 11 looks better than Channel 23 for distinguishing between target and non-target stimuli. At the peak of the mean of the target stimuli EEG signal, there is a ˜4.0 uV max potential difference

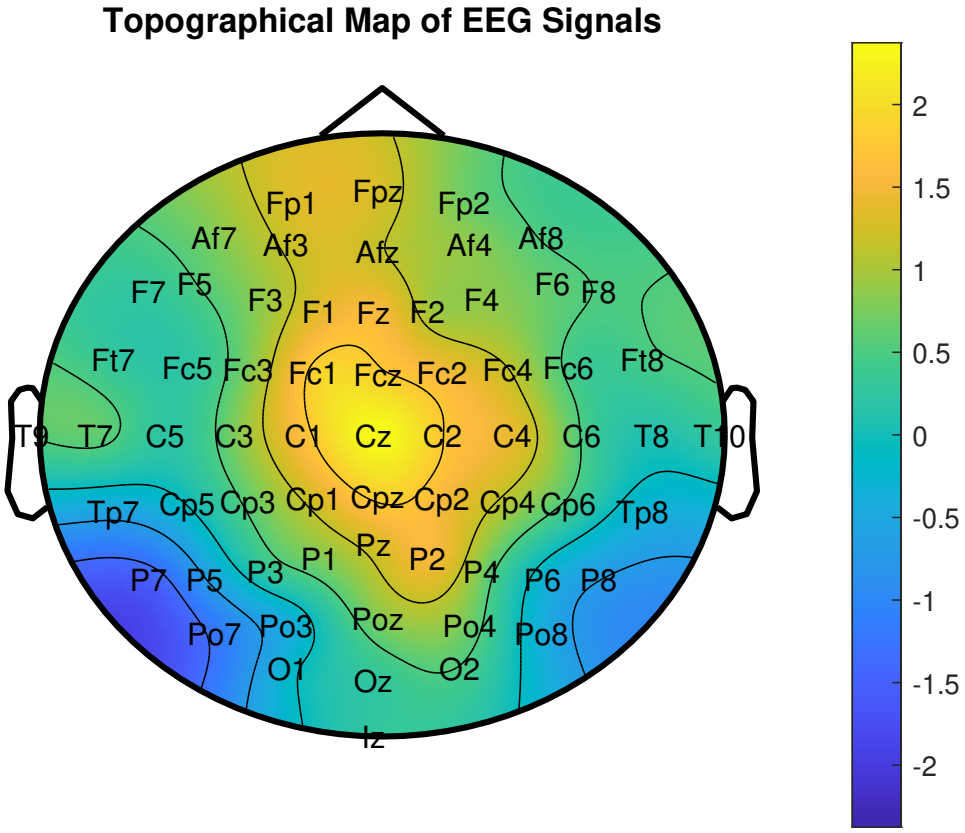
between and target and non-target (noise), occurring at about 450 ms. In Channel 23, the peak potential difference is ~2.75 uV, occurring at around 470 ms. Thus, Channel 11 displays a better mean signal to noise ratio, which would make it easier to differentiate between target and non-target stimuli EEG responses.

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)

Grab All Channels and store in Cell Array

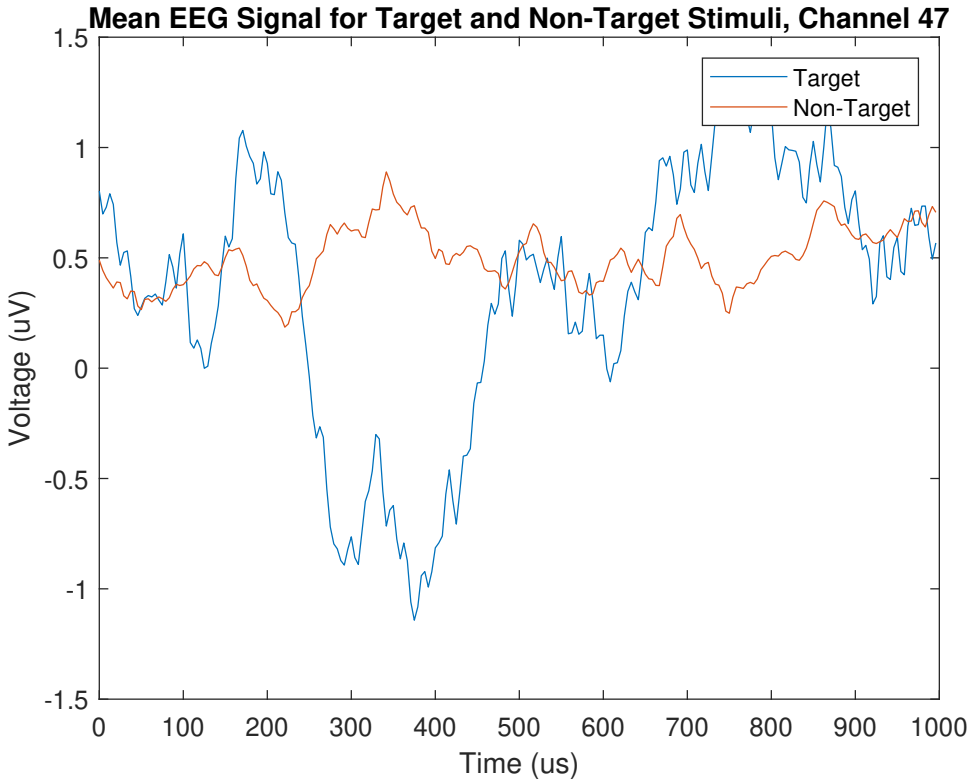
```
channelMeanDiffs = zeros(1,64);

for i = 1:64
    EEG{i} = getvalues(session.data,1:durInSec*Fs + 2,i);
end
```

```
for j = 1:64
    targetEEG = zeros(ceil(durInSec), Fs);
    nonTargetEEG = zeros(ceil(durInSec), Fs);
    for i = 1:durInSec
        if stim_type(i) == 1
            targetEEG(i,:) = EEG{j}(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
        else
            nonTargetEEG(i,:) = EEG{j}(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
        end
    end

    targetEEG = targetEEG(targetIdx,:);
    targetEEGAvg = mean(targetEEG);

    nonTargetEEG = nonTargetEEG(nonTargetIdx,:);
    nonTargetEEGAvg = mean(nonTargetEEG);

    channelMeanDiffs(j) = (targetEEGAvg(300/1000*Fs) - nonTargetEEGAvg(300/1000*Fs));
end

figure;
topoplotEEG(channelMeanDiffs,'eloc64.txt','gridscale',150,'electrodes','labels');
colorbar;
title('Topographical Map of EEG Signals');
```



5. How do the yellow and blue parts of this plot correspond to the plots from above? (2 pts)
   Example Blue region: Ch 47

```
targetEEG47 = zeros(ceil(durInSec), Fs);
nonTargetEEG47 = zeros(ceil(durInSec), Fs);
for i = 1:durInSec
    if stim_type(i) == 1
        targetEEG47(i,:) = EEG{47}(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
    else
        nonTargetEEG47(i,:) = EEG{47}(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
    end
end

targetEEG47 = targetEEG47(targetIdx,:);
targetEEGAvg47 = mean(targetEEG47);

nonTargetEEG47 = nonTargetEEG47(nonTargetIdx,:);
nonTargetEEGAvg47 = mean(nonTargetEEG47);

figure;
plot(t, targetEEGAvg47);
hold on;
plot(t, nonTargetEEGAvg47);
xlabel('Time (us)');
ylabel('Voltage (uV)');
title('Mean EEG Signal for Target and Non-Target Stimuli, Channel 47');
legend('Target', 'Non-Target');

% Example Yellow region: Ch 11
```



The yellow regions represent those where there is a high positive discriminability between the mean potential difference in target stimuli vs. non-target stimuli in that channel (at 300 ms). In the blue regions, there is a large, but negative, mean potential difference in the EEG signal at this timepoint. In the green region, there is little discriminability with a small measured potential difference between the target and non-target stimuli at 300 ms. In practice, both the yellow and blue regions could serve well for distinguishing between target and non-target stimuli.

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 to using just one channel is in increasing the usability/adoptability of a potential EEG recording device for a user, and reducing production / end user costs of the device. This is especially true if the region of the brain that is modulated by the stimuli is already. One disadvantage is that in the event of a failure of the electrode or a noisy signal on a particular trial, there would be no set of electrodes and corresponding EEG signals to reference the one channel against, reducing the accuracy of the device in the long run.

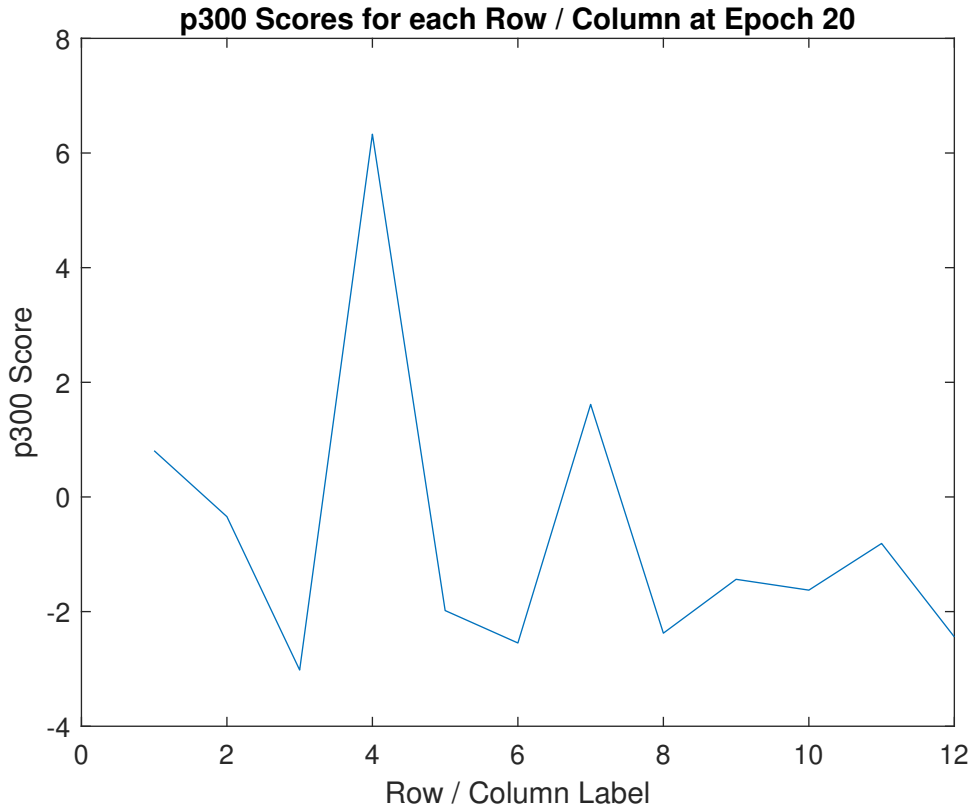
2. One simple way of identifying a P300 in a single trial (which we'll call the *p300 score*) is to take the mean EEG from 250 to 450 ms and then subtract from it the mean EEG from 600 to 700 ms. What is the *p300 score* for epoch (letter) 10, iteration 11 at electrode Cz? (3 pts)

```
iters = 12*15;
epoch = 9;

trial = EEG{11}(ceil(Stim(epoch*iters+11).start*Fs/1e6):ceil(Stim(epoch*iters+11).stop*Fs/1e6));

p300Score = mean(trial(0.25*Fs:0.45*Fs)) - mean(trial(0.6*Fs:0.7*Fs));

% p300 Score for epoch 10, iteration 11 at Cz is 1.4511
```

p300 Score epoch 10, iteration 11: 1.45

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

```
epoch = 19;
p300_scores = zeros(1,180);
row_column = zeros(1,180);

for i = 1:180
    trial = EEG{11}(ceil(Stim(epoch*iters+i).start*Fs/1e6):ceil(Stim(epoch*iters+i).stop*Fs/1e6));
    p300_scores(i) = mean(trial(0.25*Fs:0.45*Fs)) - mean(trial(0.6*Fs:0.7*Fs));
    row_column(i) = str2num(Stim(epoch*iters+i).description);
end

p300_scores_20 = zeros(1,12);
for j = 1:12
    idx = find(row_column == j);
    p300_scores_20(j) = mean(p300_scores(idx));
end

rc = 1:1:12;

figure;
plot(rc, p300_scores_20);
xlabel('Row / Column Label');
ylabel('p300 Score');
title('p300 Scores for each Row / Column at Epoch 20');
```



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

   Based on the largest peaks in p300 score occurring at 7 (row) and 4 (column), we can use the grid provided in figure 2 to predict that the person saw the letter 'D'.

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

```
epoch_scores = {};
p300_scores = {};
row_column = {};
predicted_row = zeros(1,85);
predicted_col = zeros(1,85);

for epoch = 0:84
    p300_scores{epoch+1} = zeros(1,180);
    row_column{epoch+1} = zeros(1,180);

    for i = 1:180
        trial = EEG{11}(ceil(Stim(epoch*iters+i).start*Fs/1e6):floor(Stim(epoch*iters+i).stop*Fs/1e6)); % ea
        p300_scores{epoch+1}(i) = mean(trial(0.25*Fs:0.45*Fs)) - mean(trial(0.6*Fs:0.7*Fs)); % p300 score fo
        row_column{epoch+1}(i) = str2num(Stim(epoch*iters+i).description); %
    end

    p300_scores_epoch = zeros(1,12);
    for j = 1:12
        idx = find(row_column{epoch+1} == j);
        p300_scores_epoch(j) = mean(p300_scores{epoch+1}(idx));
    end
    epoch_scores{epoch+1} = p300_scores_epoch;
```

```
    [~, predicted_col(epoch+1)] = max(epoch_scores{epoch+1}(1:6));
    [~, predicted_row(epoch+1)] = max(epoch_scores{epoch+1}(7:12));
end

grid = {'A', 'B', 'C', 'D', 'E', 'F';
    'G', 'H', 'I', 'J', 'K', 'L';
    'M', 'N', 'O', 'P', 'Q', 'R';
    'S', 'T', 'U', 'V', 'W' 'X';
    'Y', 'Z', '1', '2', '3', '4';
    '5', '6', '7', '8', '9', '_'};

predictions = {};
result = zeros(1,85);

for n = 1:85
    predictions{n} = grid{predicted_row(n), predicted_col(n)};
    result(n) = (predictions{n} == TargetLetter(n).description);
end

pred_accuracy = 100*sum(result)/85;

disp(predictions)

% Prediction accuracy is 34.12 percent
```

Prediction accuracy is 34.12 percent

```
    '4'    'A'    'P'    'V'    '5'    'U'    'F'    'O'    'H'    'I'    'O'

  Columns 12 through 22

    'P'    'B'    'R'    'J'    'M'    'F'    'C'    'E'    'D'    '_'    'C'

  Columns 23 through 33

    'T'    'W'    'I'    'D'    'B'    'P'    'C'    'O'    'A'    'W'    'R'

  Columns 34 through 44

    '5'    'I'    'U'    'K'    'O'    'L'    '3'    'O'    'O'    'E'    'W'

  Columns 45 through 55

    'E'    '6'    'T'    'G'    'K'    '4'    'R'    'Z'    '3'    'O'    '_'

  Columns 56 through 66

    'H'    'Y'    '6'    'O'    '_'    'E'    'X'    'E'    'K'    'V'    'L'

  Columns 67 through 77

    'N'    'X'    'T'    'K'    'K'    'P'    'I'    'H'    'N'    'T'    'F'

  Columns 78 through 85

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

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)

```
% let's consider using some of the ML algorithms we have exploredd earlier
% in this course

% calculate Line Length, Area, Energy, and Zx for 250-450 ms and 600-700 ms
% and use these values to get new p300 scores based on those features

% use logistic, k-nn, and svm classifiers to predict (trained on first 50,
% tested on next 35) and evaluate which has the best prediction accuracy

LLFn = @(x) sum(abs(diff(x)));
Area = @(x) sum(abs(x));
Energy = @(x) sum(x.^2);
ZX = @(x) sum((x(1:length(x)-1) > mean(x) & x(2:length(x)) < mean(x)) | (x(1:length(x)-1) < mean(x) & x(2:le

% epoch_scores_p300mod = {};
% predict_row_p300mod = zeros(1,85);
% predict_col_p300mod = zeros(1,85);
%
% for epoch = 0:49 % train with first 50 epochs
%     p300_scores_p300mod = zeros(1,180);
%     p300_scores_LL = zeros(1,180);
%     p300_scores_Area = zeros(1,180);
%     p300_scores_Energy = zeros(1,180);
%     p300_scores_ZX = zeros(1,180);
%     row_column = zeros(1,180);
%
%     for i = 1:180
%         trial = EEG{11}(ceil(Stim(epoch*iters+i).start*Fs/1e6):floor(Stim(epoch*iters+i).stop*Fs/1e6)); %
%         p300_scores_p300mod(i) = mean(trial(0.25*Fs:0.45*Fs)) - mean(trial(0.6*Fs:0.7*Fs)); % p300 score f
%         p300_scores_LL(i) = LLFn(trial(0.25*Fs:0.45*Fs)) - LLFn(trial(0.6*Fs:0.7*Fs));
%         p300_scores_Area(i) = Area(trial(0.25*Fs:0.45*Fs)) - Area(trial(0.6*Fs:0.7*Fs));
%         p300_scores_Energy(i) = Energy(trial(0.25*Fs:0.45*Fs)) - Energy(trial(0.6*Fs:0.7*Fs));
%         p300_scores_ZX(i) = ZX(trial(0.25*Fs:0.45*Fs)) - ZX(trial(0.6*Fs:0.7*Fs));
%         row_column(i) = str2num(Stim(epoch*iters+i).description);
%     end
%
%     p300_scores_epoch_p300mod = zeros(1,12);%sorted by epoch
%     p300_scores_epoch_LL = zeros(1,12);
%     p300_scores_epoch_Area = zeros(1,12);
%     p300_scores_epoch_Energy = zeros(1,12);
%     p300_scores_epoch_ZX = zeros(1,12);
%
%     for j = 1:12
%         idx = find(row_column == j);
%         p300_scores_epoch_p300mod(j) = mean(p300_scores_p300mod(idx));
%         p300_scores_epoch_LL(j) = mean(p300_scores_LL(idx));
%         p300_scores_epoch_Area(j) = mean(p300_scores_Area(idx));
%         p300_scores_epoch_Energy(j) = mean(p300_scores_Energy(idx));
%         p300_scores_epoch_ZX(j) = mean(p300_scores_ZX(idx));
%     end
%
```

```
%     epoch_scores_p300mod{epoch+1} = p300_scores_epoch_p300mod;
%     [~, predict_col_p300mod(epoch+1)] = max(epoch_scores_p300mod{epoch+1}(1:6));
%     [~, predict_row_p300mod(epoch+1)] = max(epoch_scores_p300mod{epoch+1}(7:12));
%
%     epoch_scores_LL{epoch+1} = p300_scores_epoch_LL;
%     [~, predict_col_LL(epoch+1)] = max(epoch_scores_LL{epoch+1}(1:6));
%     [~, predict_row_LL(epoch+1)] = max(epoch_scores_LL{epoch+1}(7:12));
%
%     epoch_scores_Area{epoch+1} = p300_scores_epoch_Area;
%     [~, predict_col_Area(epoch+1)] = max(epoch_scores_Area{epoch+1}(1:6));
%     [~, predict_row_Area(epoch+1)] = max(epoch_scores_Area{epoch+1}(7:12));
%
%     epoch_scores_Energy{epoch+1} = p300_scores_epoch_Energy;
%     [~, predict_col_Energy(epoch+1)] = max(epoch_scores_Energy{epoch+1}(1:6));
%     [~, predict_row_Energy(epoch+1)] = max(epoch_scores_Energy{epoch+1}(7:12));
%
%     epoch_scores_ZX{epoch+1} = p300_scores_epoch_ZX;
%     [~, predict_col_ZX(epoch+1)] = max(epoch_scores_ZX{epoch+1}(1:6));
%     [~, predict_row_ZX(epoch+1)] = max(epoch_scores_ZX{epoch+1}(7:12));
%
% end
%
%
% predictions_p300mod = {};
% result_p300mod = zeros(1,85);
%
% predictions_LL = {};
% result_LL = zeros(1,85);
%
% predictions_Area = {};
% result_Area = zeros(1,85);
%
% predictions_Energy = {};
% result_Energy = zeros(1,85);
%
% predictions_ZX = {};
% result_ZX = zeros(1,85);
%
% for n = 1:85
%     predictions_p300mod{n} = grid{predict_row_p300mod(n), predict_col_p300mod(n)};
%     result_p300mod(n) = (predictions_p300mod{n} == TargetLetter(n).description);
%
%     predictions_LL{n} = grid{predict_row_LL(n), predict_col_LL(n)};
%     result_LL(n) = (predictions_LL{n} == TargetLetter(n).description);
%
%     predictions_Area{n} = grid{predict_row_Area(n), predict_col_Area(n)};
%     result_Area(n) = (predictions_Area{n} == TargetLetter(n).description);
%
%     predictions_Energy{n} = grid{predict_row_Energy(n), predict_col_Energy(n)};
%     result_Energy(n) = (predictions_Energy{n} == TargetLetter(n).description);
%
%     predictions_ZX{n} = grid{predict_row_ZX(n), predict_col_ZX(n)};
%     result_ZX(n) = (predictions_ZX{n} == TargetLetter(n).description);
% end
%
% pred_accuracy_p300mod = 100*sum(result_p300mod)/85;
% pred_accuracy_LL = 100*sum(result_LL)/85;
% pred_accuracy_Area = 100*sum(result_Area)/85;
% pred_accuracy_Energy = 100*sum(result_Energy)/85;
% pred_accuracy_ZX = 100*sum(result_ZX)/85;


% DIDNT WORK: Calculating p300 scores based on features of the 250:450 ms
% and 600:700 ms windows including Line Length, Area, Energy, and ZX.  Did
% not implement training vs testing set though.
```

```
% Next step: figure out how to integrate knowledge about targetEEG and
% nonTargetEEG into this model
```

```
% first epoch, first iteration, channel 11, all features

% 30 target flashes per 180 iterations in each epoch
%
% for epoch = 0:49
%     p300_scores_p300mod =  zeros(1,100);
%     line_length = zeros(1,180);
%     area = zeros(1,180);
%     energy = zeros(1,180);
%     zx = zeros(1,180);
%     row_column = zeros(1,180);
%     for i = 1:180
%         trial = EEG{11}(ceil(Stim(epoch*iters+i).start*Fs/1e6):floor(Stim(epoch*iters+i).stop*Fs/1e6)); %
%         p300_scores_p300mod(i) = mean(trial(0.25*Fs:0.45*Fs)) - mean(trial(0.6*Fs:0.7*Fs));
%         line_length(i) = LLFn(trial);
%         area(i) = Area(trial);
%         energy(i) = Energy(trial);
%         zx(i) = ZX(trial);
%         row_column(i) = str2num(Stim(epoch*iters+i).description);
%     end
%
%     epoch_p300mod = zeros(1,12);
%     epoch_LL = zeros(1,12);
%     epoch_Area = zeros(1,12);
%     epoch_Energy = zeros(1,12);
%     epoch_ZX = zeros(1,12);
%
%     for j = 1:12
%         idx = find(row_column == j);
%         epoch_p300mod(j) = mean(p300_scores_p300mod(idx));
%         epoch_LL(j) = mean(line_length(idx));
%         epoch_Area(j) = mean(area(idx));
%         epoch_Energy(j) = mean(energy(idx));
%         epoch_ZX(j) = mean(zx(idx));
%     end
%
%     epoch_scores_p300mod{epoch+1} = epoch_p300mod;
%     [~, predict_col_p300mod(epoch+1)] = max(epoch_scores_p300mod{epoch+1}(1:6));
%     [~, predict_row_p300mod(epoch+1)] = max(epoch_scores_p300mod{epoch+1}(7:12));
%
%     epoch_scores_LL{epoch+1} = epoch_LL;
%     [~, predict_col_LL(epoch+1)] = max(epoch_scores_LL{epoch+1}(1:6));
%     [~, predict_row_LL(epoch+1)] = max(epoch_scores_LL{epoch+1}(7:12));
%
%     epoch_scores_Area{epoch+1} = epoch_Area;
%     [~, predict_col_Area(epoch+1)] = max(epoch_scores_Area{epoch+1}(1:6));
%     [~, predict_row_Area(epoch+1)] = max(epoch_scores_Area{epoch+1}(7:12));
%
%     epoch_scores_Energy{epoch+1} = epoch_Energy;
%     [~, predict_col_Energy(epoch+1)] = max(epoch_scores_Energy{epoch+1}(1:6));
%     [~, predict_row_Energy(epoch+1)] = max(epoch_scores_Energy{epoch+1}(7:12));
%
%     epoch_scores_ZX{epoch+1} = epoch_ZX;
%     [~, predict_col_ZX(epoch+1)] = max(epoch_scores_ZX{epoch+1}(1:6));
%     [~, predict_row_ZX(epoch+1)] = max(epoch_scores_ZX{epoch+1}(7:12));
% end
%
% for n = 1:50
%     predictions_p300mod{n} = grid{predict_row_p300mod(n), predict_col_p300mod(n)};
```

```
%     result_p300mod(n) = (predictions_p300mod{n} == TargetLetter(n).description);
%
%     predictions_LL{n} = grid{predict_row_LL(n), predict_col_LL(n)};
%     result_LL(n) = (predictions_LL{n} == TargetLetter(n).description);
%
%     predictions_Area{n} = grid{predict_row_Area(n), predict_col_Area(n)};
%     result_Area(n) = (predictions_Area{n} == TargetLetter(n).description);
%
%     predictions_Energy{n} = grid{predict_row_Energy(n), predict_col_Energy(n)};
%     result_Energy(n) = (predictions_Energy{n} == TargetLetter(n).description);
%
%     predictions_ZX{n} = grid{predict_row_ZX(n), predict_col_ZX(n)};
%     result_ZX(n) = (predictions_ZX{n} == TargetLetter(n).description);
% end
%
% pred_accuracy_p300mod = 100*sum(result_p300mod)/50;
% pred_accuracy_LL = 100*sum(result_LL)/50;
% pred_accuracy_Area = 100*sum(result_Area)/50;
% pred_accuracy_Energy = 100*sum(result_Energy)/50;
% pred_accuracy_ZX = 100*sum(result_ZX)/50;
```

```
% For each epoch (letter spelling) we have 30 target stimuli and 150
% non-target stimuli.  These 30 target stimuli represetn 15 flashes for the
% target letter row and 15 flashes for the target letter column, each of
% which can be found using Stim(i).description

% so, for every flash in the dataset, we know both the row/column that was
% shown and whether it is targert or not (1 or 0).

class = zeros(15300,1); % Target or Non-Target Stimuli

targetEEG = zeros(ceil(durInSec), Fs);
nonTargetEEG = zeros(ceil(durInSec), Fs);

targetIdx = find(stim_type == 1);
nonTargetIdx = find(stim_type == 0);

for i = 1:durInSec
    if stim_type(i) == 1
        targetEEG(i,:) = EEG{11}(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
        class(i) = 1;
    else
        nonTargetEEG(i,:) = EEG{11}(ceil(Stim(i).start*Fs/1e6):ceil(Stim(i).stop*Fs/1e6));
        class(i) = 2;
    end
end

% try looking at only 200:500 ms and calculating all features, including
% signal mean, line length, area, energy, and Z-crossings.  This is the
% temporal region that appears exhbiit the greatest difference between
% target and non-target signal potential.

% LOOKED AT 200:500 ms FOR ALL EPOCHS, MEAN YIELDED A
% 43.53% accuracy

%trial_Energy{epoch}(i)
for epoch = 0:49
    n = 1;
    m = 1;

    trial_p300mod{epoch+1} = zeros(1,180);
    trial_LL{epoch+1} = zeros(1,180);
    trial_Area{epoch+1} = zeros(1,180);
```

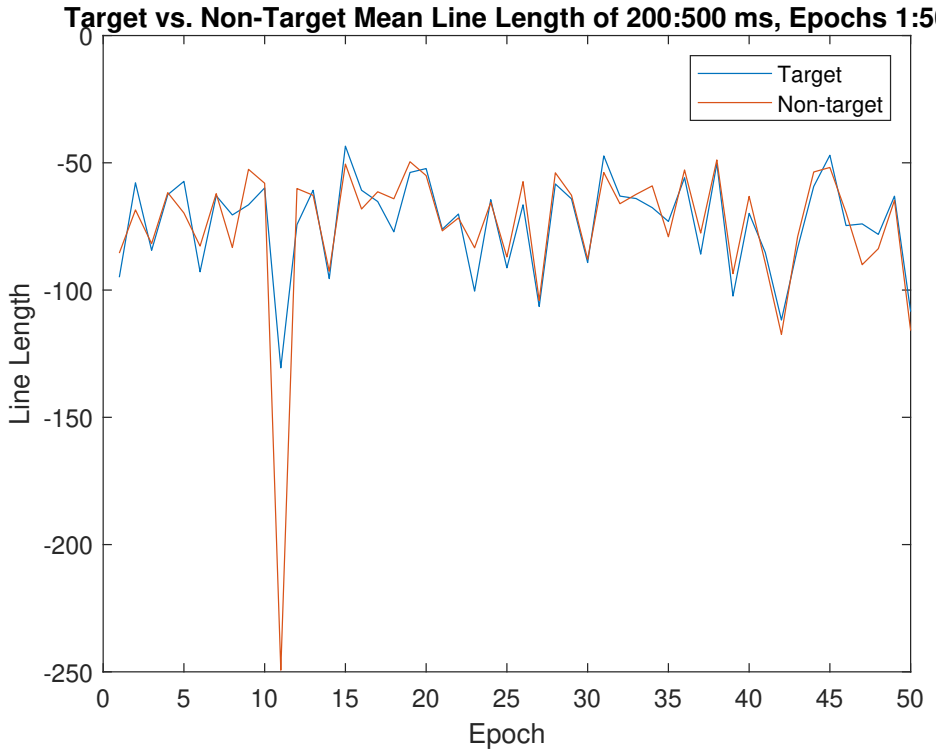
```
    trial_Energy{epoch+1} = zeros(1,180);
    trial_ZX{epoch+1} = zeros(1,180);
    row_col{epoch+1} = zeros(1,180);

    target_trial_p300mod{epoch+1} = zeros(1,180);
    target_trial_LL{epoch+1} = zeros(1,180);
    target_trial_Area{epoch+1} = zeros(1,180);
    target_trial_Energy{epoch+1} = zeros(1,180);
    target_trial_ZX{epoch+1} = zeros(1,180);

    nontarget_trial_p300mod{epoch+1} = zeros(1,180);
    nontarget_trial_LL{epoch+1} = zeros(1,180);
    nontarget_trial_Area{epoch+1} = zeros(1,180);
    nontarget_trial_Energy{epoch+1} = zeros(1,180);
    nontarget_trial_ZX{epoch+1} = zeros(1,180);

    for i = 1:180
        trial = EEG{11}(ceil(Stim(epoch*iters+i).start*Fs/1e6):floor(Stim(epoch*iters+i).stop*Fs/1e6));
        p300mod = trial(0.6*Fs:1.0*Fs); % REGION WHERE NON-TARGET > TARGET
        trial = trial(0.2*Fs:0.5*Fs);
        trial_p300mod{epoch+1}(i) = mean(trial) - mean(p300mod); % MODIFIED p300 SCORING
        trial_LL{epoch+1}(i) = LLFn(trial) - LLFn(p300mod); % LINE LENGTH SCORING
        trial_Area{epoch+1}(i) = Area(trial) - Area(p300mod); % AREA SCORING
        trial_Energy{epoch+1}(i) = Energy(trial) - Energy(p300mod); % ENERGY SCORING
        trial_ZX{epoch+1}(i) = ZX(trial) - ZX(p300mod); % ZX SCORING
        row_col{epoch+1}(i) = str2num(Stim(epoch*iters+i).description); % FLASHED ROW OR COL
        if class(epoch*iters + i) == 1 % SEPARATE TARGET FROM NON-TARGET
            target_trial_p300mod{epoch+1}(n) = trial_p300mod{epoch+1}(i);
            target_trial_LL{epoch+1}(n) = trial_LL{epoch+1}(i);
            target_trial_Area{epoch+1}(n) = trial_Area{epoch+1}(i);
            target_trial_Energy{epoch+1}(n) = trial_Energy{epoch+1}(i);
            target_trial_ZX{epoch+1}(n) = trial_ZX{epoch+1}(i);
            n = n+1;
        else
            nontarget_trial_p300mod{epoch+1}(n) = trial_p300mod{epoch+1}(i);
            nontarget_trial_LL{epoch+1}(n) = trial_LL{epoch+1}(i);
            nontarget_trial_Area{epoch+1}(n) = trial_Area{epoch+1}(i);
            nontarget_trial_Energy{epoch+1}(n) = trial_Energy{epoch+1}(i);
            nontarget_trial_ZX{epoch+1}(n) = trial_ZX{epoch+1}(i);
            n = n+1;
        end
    end
    % AVERAGES OF FEATURES ACROSS ENTIRE EPOCH FOR TARGET VS. NON-TARGET
    target_trial_p300mod_avg(epoch+1) = sum(target_trial_p300mod{epoch+1})/30;
    target_trial_LL_avg(epoch+1) = sum(target_trial_LL{epoch+1})/30;
    target_trial_Area_avg(epoch+1) = sum(target_trial_Area{epoch+1})/30;
    target_trial_Energy_avg(epoch+1) = sum(target_trial_Energy{epoch+1})/30;
    target_trial_ZX_avg(epoch+1) = sum(target_trial_ZX{epoch+1})/30;

    nontarget_trial_p300mod_avg(epoch+1) = sum(nontarget_trial_p300mod{epoch+1})/150;
    nontarget_trial_LL_avg(epoch+1) = sum(nontarget_trial_LL{epoch+1})/150;
    nontarget_trial_Area_avg(epoch+1) = sum(nontarget_trial_Area{epoch+1})/150;
    nontarget_trial_Energy_avg(epoch+1) = sum(nontarget_trial_Energy{epoch+1})/150;
    nontarget_trial_ZX_avg(epoch+1) = sum(nontarget_trial_ZX{epoch+1})/150;

    scores_epoch_p300mod = zeros(1,12);%sorted by epoch
    scores_epoch_LL = zeros(1,12);
    scores_epoch_Area = zeros(1,12);
    scores_epoch_Energy = zeros(1,12);
    scores_epoch_ZX = zeros(1,12);

    for j = 1:12
        idx = find(row_col{epoch+1} == j);
        scores_epoch_p300mod(j) = mean(trial_p300mod{epoch+1}(idx));
        scores_epoch_LL(j) = mean(trial_LL{epoch+1}(idx));
```

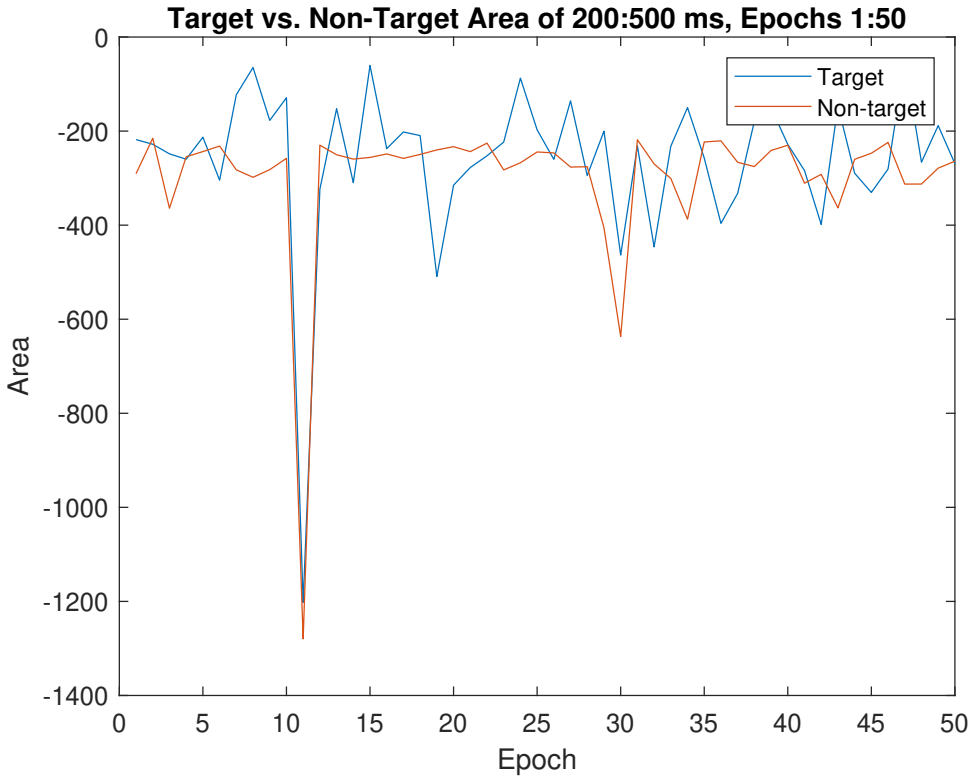
```
        scores_epoch_Area(j) = mean(trial_Area{epoch+1}(idx));
        scores_epoch_Energy(j) = mean(trial_Energy{epoch+1}(idx));
        scores_epoch_ZX(j) = mean(trial_ZX{epoch+1}(idx));
    end

    epoch_scores_p300mod{epoch+1} = scores_epoch_p300mod;
    [~, predict_col_p300mod(epoch+1)] = max(scores_epoch_p300mod(1:6));
    [~, predict_row_p300mod(epoch+1)] = max(scores_epoch_p300mod(7:12));

    epoch_scores_LL{epoch+1} = scores_epoch_LL;
    [~, predict_col_LL(epoch+1)] = max(scores_epoch_LL(1:6));
    [~, predict_row_LL(epoch+1)] = max(scores_epoch_LL(7:12));

    epoch_scores_Area{epoch+1} = scores_epoch_Area;
    [~, predict_col_Area(epoch+1)] = max(scores_epoch_Area(1:6));
    [~, predict_row_Area(epoch+1)] = max(scores_epoch_Area(7:12));

    epoch_scores_Energy{epoch+1} = scores_epoch_Energy;
    [~, predict_col_Energy(epoch+1)] = max(scores_epoch_Energy(1:6));
    [~, predict_row_Energy(epoch+1)] = max(scores_epoch_Energy(7:12));

    epoch_scores_ZX{epoch+1} = scores_epoch_ZX;
    [~, predict_col_ZX(epoch+1)] = max(scores_epoch_ZX(1:6));
    [~, predict_row_ZX(epoch+1)] = max(scores_epoch_ZX(7:12));
end

for n = 1:50
    predictions_p300mod{n} = grid{predict_row_p300mod(n), predict_col_p300mod(n)};
    result_p300mod(n) = (predictions_p300mod{n} == TargetLetter(n).description);

    predictions_LL{n} = grid{predict_row_LL(n), predict_col_LL(n)};
    result_LL(n) = (predictions_LL{n} == TargetLetter(n).description);

    predictions_Area{n} = grid{predict_row_Area(n), predict_col_Area(n)};
    result_Area(n) = (predictions_Area{n} == TargetLetter(n).description);

    predictions_Energy{n} = grid{predict_row_Energy(n), predict_col_Energy(n)};
    result_Energy(n) = (predictions_Energy{n} == TargetLetter(n).description);

    predictions_ZX{n} = grid{predict_row_ZX(n), predict_col_ZX(n)};
    result_ZX(n) = (predictions_ZX{n} == TargetLetter(n).description);
end

pred_accuracy_p300mod = 100*sum(result_p300mod)/50; % PREDICTION ACCURACY: 48 percent
pred_accuracy_LL = 100*sum(result_LL)/50; % PREDICTION ACCURACY: 4 percent
pred_accuracy_Area = 100*sum(result_Area)/50; % PREDICTION ACCURACY: 6 percent
pred_accuracy_Energy = 100*sum(result_Energy)/50; % PREDICTION ACCURACY: 6 percent
pred_accuracy_ZX = 100*sum(result_ZX)/50; % PREDICTION ACCURACY: 2 percent
```

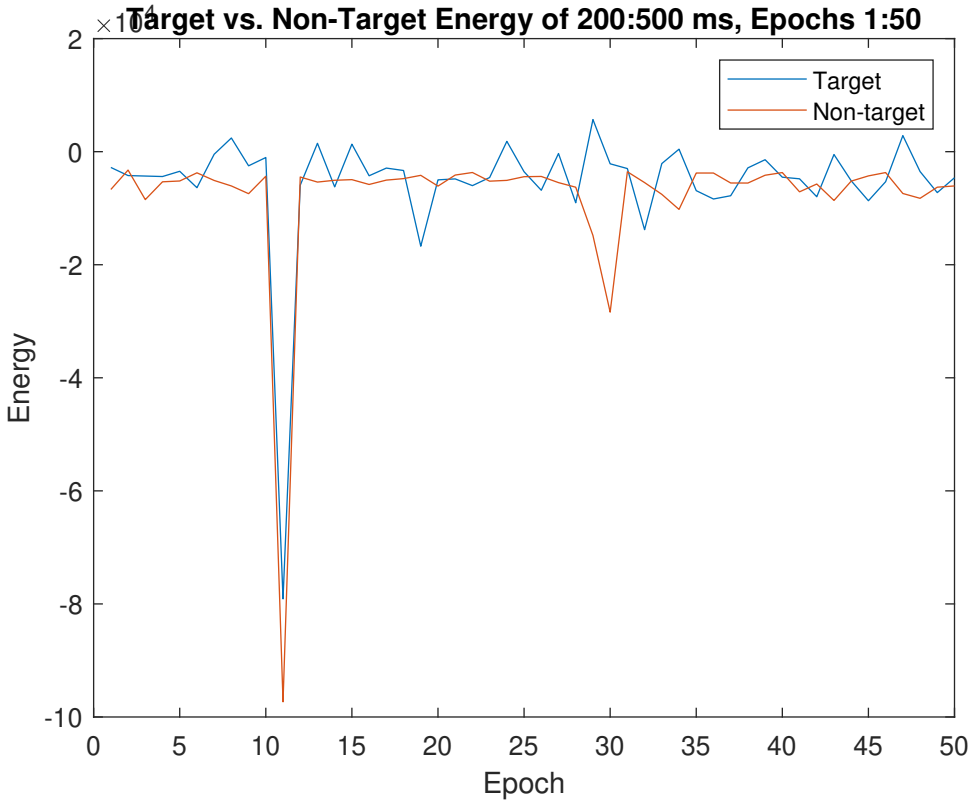
```
% PLOT MEAN OF THE FEATURES SEGMENTED BY TARGET VS NON-TARGET IN 200-500 ms
% WINDOW TO DETERMINE BEST FEATURE TO TRAIN WITH

figure;
plot(1:50, target_trial_p300mod_avg);
hold on;
plot(1:50, nontarget_trial_p300mod_avg);
xlabel('Epoch');
ylabel('Modified p300 Score)');
legend('Target', 'Non-target');
title('Target vs. Non-Target Mean Modified p300 Score, Epochs 1:50');

figure;
plot(1:50, target_trial_LL_avg);
```

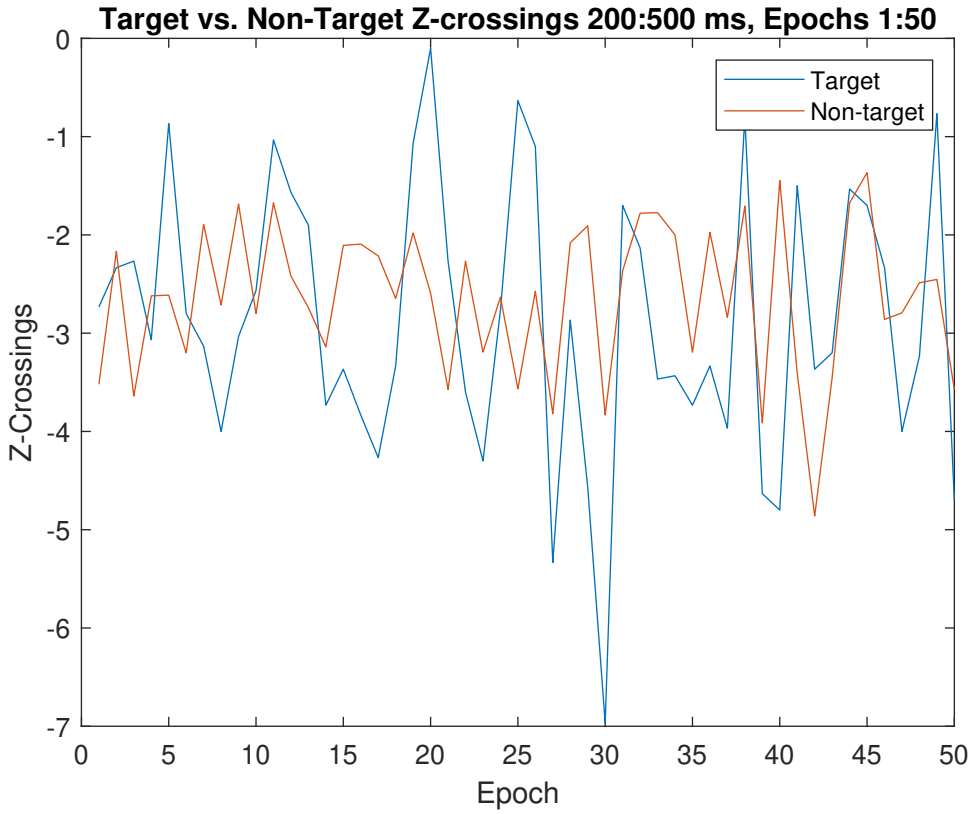
```
hold on;
plot(1:50, nontarget_trial_LL_avg);
xlabel('Epoch');
ylabel('Line Length');
legend('Target', 'Non-target');
title('Target vs. Non-Target Mean Line Length of 200:500 ms, Epochs 1:50');

figure;
plot(1:50, target_trial_Area_avg);
hold on;
plot(1:50, nontarget_trial_Area_avg);
xlabel('Epoch');
ylabel('Area');
legend('Target', 'Non-target');
title('Target vs. Non-Target Area of 200:500 ms, Epochs 1:50');

figure;
plot(1:50, target_trial_Energy_avg);
hold on;
plot(1:50, nontarget_trial_Energy_avg);
xlabel('Epoch');
ylabel('Energy');
legend('Target', 'Non-target');
title('Target vs. Non-Target Energy of 200:500 ms, Epochs 1:50');

figure;
plot(1:50, target_trial_ZX_avg);
hold on;
plot(1:50, nontarget_trial_ZX_avg);
xlabel('Epoch');
ylabel('Z-Crossings');
legend('Target', 'Non-target');
title('Target vs. Non-Target Z-crossings 200:500 ms, Epochs 1:50');


% look at each signal / extracted feature and train target vs non-target.
% Based on this, determine if a signal is target or non-target, and pick
% out row/column according to whether or not it is predicted to be a target
% stimuli.

% FROM THESE PLOTS, MEAN APPEARS TO BE THE ONLY FEATURE THAT IS USEFUL IN
% DISTINGUISHING TARGET VS NON-TARGET STIMULI BASED ON THE EEG
```


Target vs. Non-Target Mean Modified p300 Score, Epochs 1:50
Target
Non-target
Modified p300 Score)
Epoch
8
6
4
2
0
-2
-4
-6
-8
-10
-12
0
5
10
15
20
25
30
35
40
45
50


Target vs. Non-Target Mean Line Length of 200:500 ms, Epochs 1:50
Target
Non-target
Line Length
Epoch
0
-50
-100
-150
-200
-250
0
5
10
15
20
25
30
35
40
45
50


Target vs. Non-Target Area of 200:500 ms, Epochs 1:50
Target
Non-target
Area
Epoch
0
-200
-400
-600
-800
-1000
-1200
-1400
0
5
10
15
20
25
30
35
40
45
50


Target vs. Non-Target Energy of 200:500 ms, Epochs 1:50
×10^4
Target
Non-target
Energy
Epoch
2
0
-2
-4
-6
-8
-10
0
5
10
15
20
25
30
35
40
45
50



TRY TRAINING BASED ON MODIFIED p300 SCORING

```
% GRAB TRAINING VALUES
for epoch = 0:49
    for i = 1:180
        train_p300mod(epoch*iters + i) = trial_p300mod{epoch+1}(i);
        train_p300modZ(epoch*iters + i) = (trial_p300mod{epoch+1}(i) - mean(trial_p300mod{epoch+1}))./std(tr
% Z-scored area
    end
end

% GRAB TEST VALUES (RAN ONCE, VALUES STORED AFTERWARDS)
% for epoch = 50:84
%     for i = 1:180
%         test_p300mod((epoch-50)*iters + i) = trial_p300mod{epoch+1}(i);
%         test_p300modZ((epoch-50)*iters + i) = (trial_p300mod{epoch+1}(i) - mean(trial_p300mod{epoch+1}))./
% Z-scored area
%     end
% end


train_p300mod = reshape(train_p300mod, [9000 1]);
train_p300modZ = reshape(train_p300modZ, [9000 1]);

% TRY LOGISTIC REGRESSION CLASSIFIER

test_p300mod = reshape(test_p300mod, [6300 1]);
test_p300modZ = reshape(test_p300modZ, [6300 1]);
```

```
train_class = class(1:9000);
%
B = mnrfit(train_p300modZ,train_class); % coefficients
% training error (percentage);
probs = mnrval(B,train_p300modZ);
%
[~,Ytrain] = max(probs,[],2);
training_error = 100*((length(train_class) - sum(Ytrain == train_class))/length(train_class));
% 16.83 percent -> assumes all (except a very small percentage) are
% Non-Target -> incorrect /16 or 16.67 percent of the time
% Predicted that 16.83 - 16.67 = 0.16 percent were false positive targets
% or false negative non-targets

% TRY RUNNING ON REMAINING 35 EPOCHS TO DETERMINE TARGET VS NONTARGET

test_class = class(9001:15300);
probabilities = mnrval(B,test_p300modZ);

[~,Ytest] = max(probabilities,[],2);
test_error = 100*((length(test_class) - sum(Ytest == test_class))/length(test_class));
% Test error: 16.78%

% 2 different methods:
% 1. try training like in HW 4 using three different classifiers based on features
% 2. try using log(likelihood) method from HW5

% TRY K-NN

knn_mdl = fitcknn(train_p300modZ,train_class);
knn_train = predict(knn_mdl,train_p300modZ);
knn_train_error = 100*((length(train_class) - sum(knn_train == train_class))/length(train_class));
% 0 percent training error

knn_test = predict(knn_mdl,test_p300modZ);
knn_test_error = 100*((length(test_class) - sum(knn_test == test_class))/length(test_class));
% 28.22 percent test error

% TRY SVM

SVM_model = fitcsvm(train_p300modZ,train_class,'KernelFunction','RBF');
SVM_train = predict(SVM_model, train_p300modZ);
SVM_train_error = 100*((length(train_class) - sum(SVM_train == train_class))/length(train_class));
% 16.61 percent SVM training error

SVM_test = predict(SVM_model, test_p300modZ);
SVM_test_error = 100*((length(test_class) - sum(SVM_test == test_class))/length(test_class));
% 16.70 percent SVM testing error
```

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

   I first tried re-calculating p300 scores based on line length, area, energy, and z-crossings in each of the 250:450 ms and 600:700 ms windows for each iteration (or flash)). This yielded prediction accuracies that did not exceed 5 percent, and it makes sense because I was calculating features over unequal time windows and taking the difference. I next tried calculating the features over the whole signal for each iteration within each epoch, and then predicted based on the row and column that had the highest average feature value within each epoch (across all 180 iterations). The highest prediction accuracies for these methods were 6 percent for Area and Energy features, still not beating the original p300 score values.

   I next tried adjusting the window used to calculate the p300 scores, and found the highest prediction accuracy on 50 training epochs of 48 percent when using a window of 200:500 ms and subtracting the window of 600:1000 ms from it. When calculating the features on each of these, the prediction

accuracies still did not exceed 6 percent. To figure out which of the 5 features would work best, I visualized the average of each feature within each epoch, segmented by Target vs. Non-Target stimuli to see if there was any trend (ex. "Target stimuli have a significantly higher area in each epoch than Non-Target stimuli") that I may be able to use to train a classifier. As shown, the only feature that appears to do a decent job of distinguishing between target and non-target stimuli is the p300 score over the modified windows. Thus, I proceeded by attempting to train the model using the modified p300 scores calculated. In attempting to train these classifiers on the first 50 epochs to determine whether a p300mod score (p300mod Z-score, for normalized attempts) was associated with a target or non-target. In each model, the testing and training errors hovered around 16.67 percent. Upon further inspection, each model simply predicted all to be non-target, with a few false target classification and missed non-target classifications.

Thus, the best prediction accuracy I was able to achieve was 48 percent, using the original p300 formula with modified windows 200:500 ms and 600:1000 ms. I have included the plots comparing the feature averages in each epoch for target and non-target to show the difficulty in training a model based on these elements with a high level of discriminability. It appears as though the modified p300 scores should have worked, but I could not get the logistic regression, k-NN, or SVM models to predict any more accurately whether a stimulus was target or non-target.