Project Report for "*Applied Neuroscience*"

Biomedical Engineering MSc

By: *Agapi Konstantina Liontou 1057757*

For this project we were assigned to use an EEG data which was recorded from 60channels for 700 timepoints across 80 trials during a face.vs.car categorization task. According to the dataset, each timepoint represents 1ms of EEG activity and the stimulus presentation starts at the 100th ms and lasts for 50ms, so the first 99ms of each trial are pre-stimulus EEG recordings. Moreover, we had in our disposal a 1x80 vector, named stim, indicating which stimulus was presented on each trial, 0 is for face and 1 is for car.

The code for the project was created and executed in Matlab version 2022b.

Part A

First, I must load my data and reshape it in such a manner so as to use it for the next questions. So, I use the reshape function to convert this 3D array (60, 700, 80) into a 2D array where each column corresponds to the EEG data from a single trial. The reshaped data will have a shape of (60, 700*80) which means there will be 60 rows corresponding to the number of channels, and 700*80 columns corresponding to the timepoints across all trials. This reshaping is necessary for some data analysis techniques that expect a 2D input format, such as performing PCA.

```
%% Load data, plots and reshape

load("EEGdata.mat")

% Reshape my data into a 2d matrix
% where column:trial and row:timepoint
reshaped_data = reshape(EEGdata, [60,700*80]);

% transpose my data
reshaped_data=reshaped_data.';
```

A1.

In this part, we were asked to perform PCA to identify EEG channels that covary and define the number of principal components that are needed to describe this dataset.

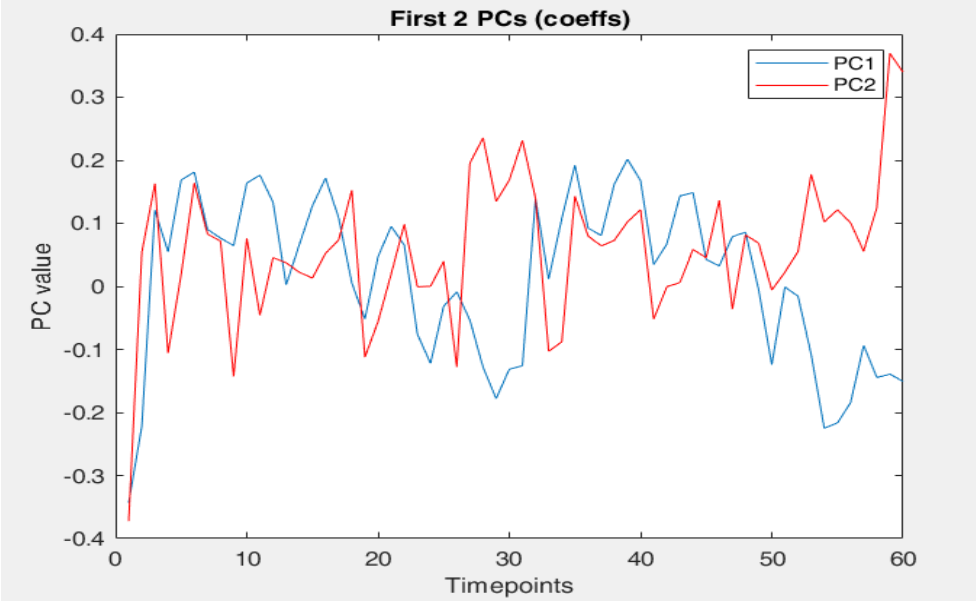
```
%% A1: Perform PCA and plot the output

% Perform PCA
[coeff, score, latent] = pca(reshaped_data);

% Plot first 2 PCs
figure;
plot(-coeff(:,1));
hold on;
plot(-coeff(:,2),'r');
legend('PC1','PC2');
```

```
% Plot data projected on first 2 PCs
figure;
plot(score(:,1), score(:,2),'.','MarkerSize',1);
axis equal;
xlabel('PC1');
ylabel('PC2');

% Plot eigenvalue magnitude
figure;
bar(latent/sum(latent))
xlabel('Dimension');
ylabel('Eigenvalue magnitude');

% Plot variance explained as a function of number of PCs
% Plot proportion of variance explained by each PC
figure;
plot(cumsum(latent)/sum(latent)*100);
xlabel('Number of PCs'); % xlabel('Number of PCs')
ylabel('Proportion of variance explained'); % ylabel('Proportion of variance
explained')

% Find the the number of PCs that are needed by using a threshold
PC_number = find(cumsum(latent)/sum(latent) > 0.9, 1);
fprintf("The number of PCs by a threshold of 0.9 is %d\n", PC_number);
```

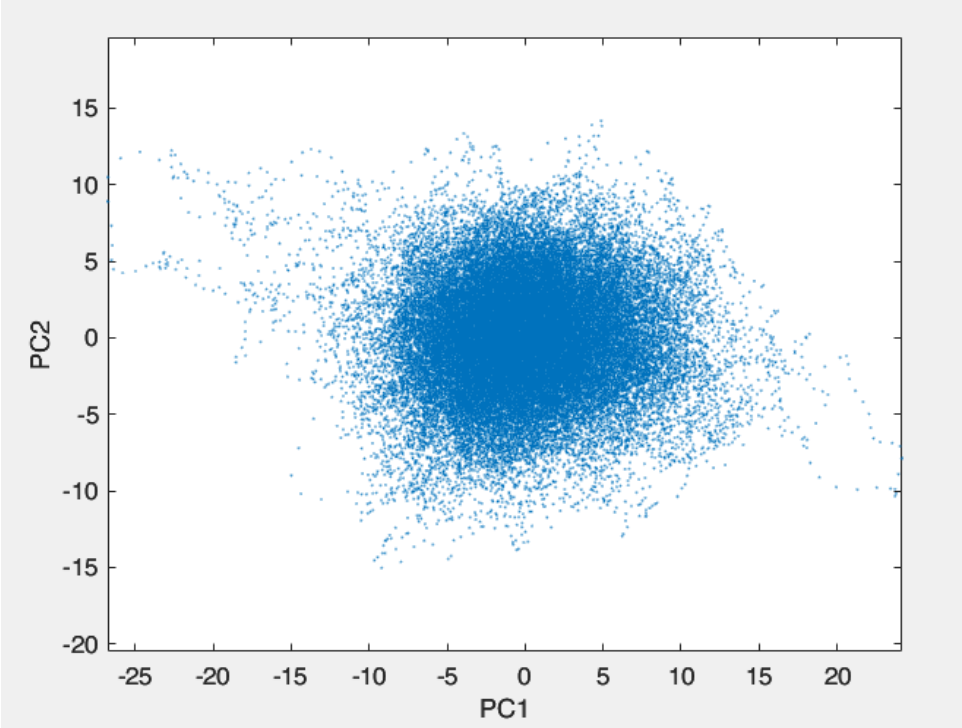
After reshaping the data, I input it into the pca.m function to get the output [coeff, score, latent]. Coeff are the principal component coefficients, score is the principal component scores and latent are the eigenvalues that represent the proportion of total variance explained by each principal component.

For the plots:

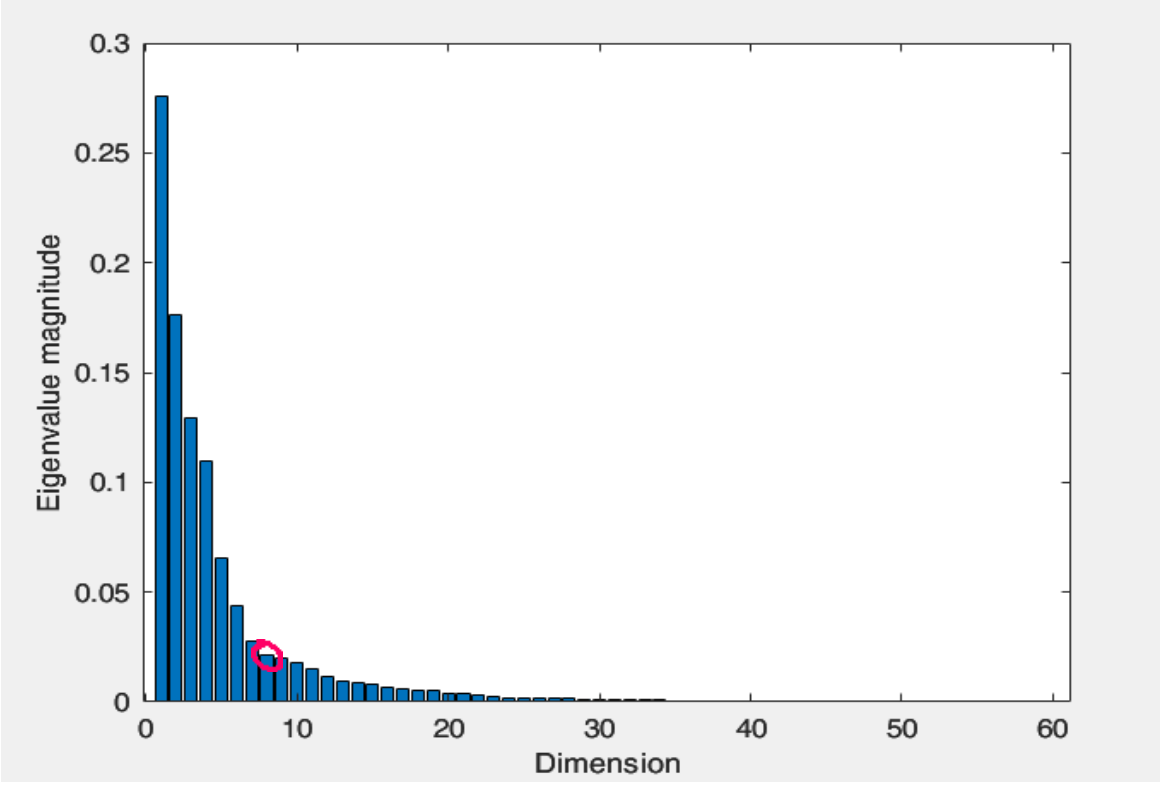
Firstly, I plot the first two principal component coefficients (**coeff**) with PC1 in blue and PC2 in red.



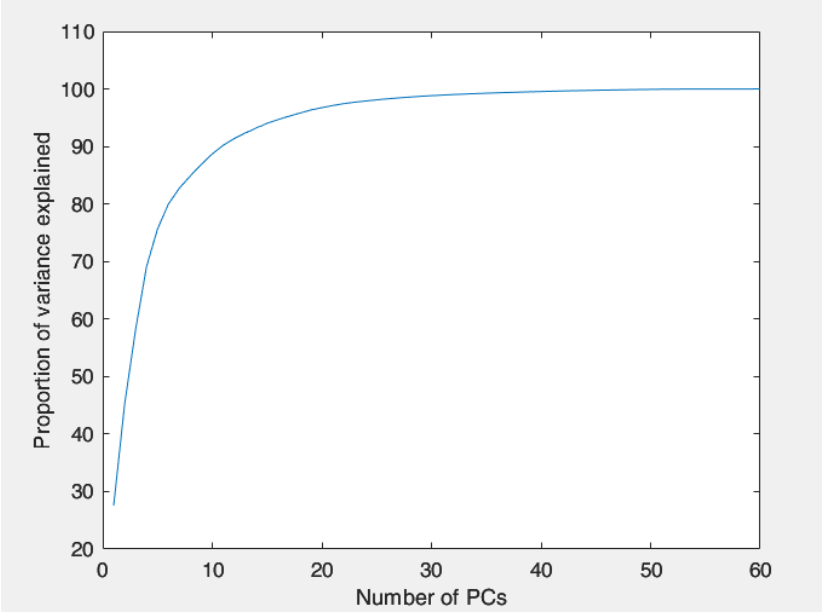
Then, I create another figure and plot the first two principal component scores (**score**) against each other, with PC1 on the x-axis and PC2 on the y-axis.



The next plot is for the magnitudes of the eigenvalues (**latent**) for each dimension, normalized by the sum of all eigenvalues. The **bar** function creates a bar chart.



The last plot is the proportion of variance explained by each additional principal component, as a percentage of the total variance. The **cumsum** function calculates the cumulative sum of the eigenvalues.



Finally, I find the number of principal components needed to explain 90% of the total variance.

```
Command Window
  The number of PCs by a threshold of 0.9 is 11
```

In general, there are 3 ways to answer how many PCs are needed:

1) from the graphic you choose the point that becomes flatter,
2) you choose those that have an eigenvalue above 1,
3) you put a percentage, e.g. 85%, and you choose the first ones that will cumulatively reach the percentage

Here, from the graphic we could say that 8 PCs are enough, and by a threshold of 0.9 we could say that the appropriate number is 11.

A2.

In this part, we were asked to plot the first 2 PCs (coeffs) and their trial-averaged activations (scores) for each one of the 2 stimuli.

```
%% A2: Plot the first 2 PCs (coeffs) and their trial-averaged activations (scores)
for each one of the 2 stimuli

% Find the kind of stimulus on each trial (0=face, 1=car)
stim_0 = find(stim == 0);
stim_1 = find(stim == 1);

% Extract scores for each stimulus
stim_0_score = score(stim_0, 1:2);
stim_1_score = score(stim_1, 1:2);

% Find trial-averaged activations (scores) for each one of the 2 stimuli
stim_0_score_mean = mean(stim_0_score,2);
stim_1_score_mean = mean(stim_1_score,2);

% Plot first 2 PCs (same as before but with reversed coeffs)
plot(coeff(:,1:2))
xlabel('Timepoints')
ylabel('PC value')
legend('PC1', 'PC2')
title('First 2 PCs (coeffs)')

% Plot the first 2 PCs (coeffs) for each one of the 2 stimuli
figure
subplot(2,2,1)
scatter(stim_0_score(:,1), stim_0_score(:,2),'b')
title('scores for stim 0: face')
xlabel('PC 1')
ylabel('PC 2')

subplot(2,2,2)
scatter(stim_1_score(:,1), stim_1_score(:,2),'r')
title('scores for stim 1: car')
xlabel('PC 1')
ylabel('PC 2')

% Plot the first 2 PCs for each stimulus in one diagramm
subplot(2,2,[3,4])
scatter(stim_0_score(:, 1), stim_0_score(:, 2), 'b');
hold on;
scatter(stim_1_score(:, 1), stim_1_score(:, 2), 'r');
title('scores for both stimulus')
xlabel('PC1');
ylabel('PC2');
legend('stim 0: face', 'stim 1: car');

% Reshape scores into trials
score_trials = reshape(score, [700, 80, 60]);

% Compute trial-averaged signals separately for each stimulus for PC1 and PC2
stimulus_0_scores = score_trials(:, stim_0, :);
stimulus_1_scores = score_trials(:, stim_1, :);
stimulus_0_av_pc1 = squeeze(mean(stimulus_0_scores(:, :, 1), 2));
stimulus_1_av_pc1 = squeeze(mean(stimulus_1_scores(:, :, 1), 2));
stimulus_0_av_pc2 = squeeze(mean(stimulus_0_scores(:, :, 2), 2));
stimulus_1_av_pc2 = squeeze(mean(stimulus_1_scores(:, :, 2), 2));

% Plot trial-averaged signals separately for each stimulus for PC1
figure;
```

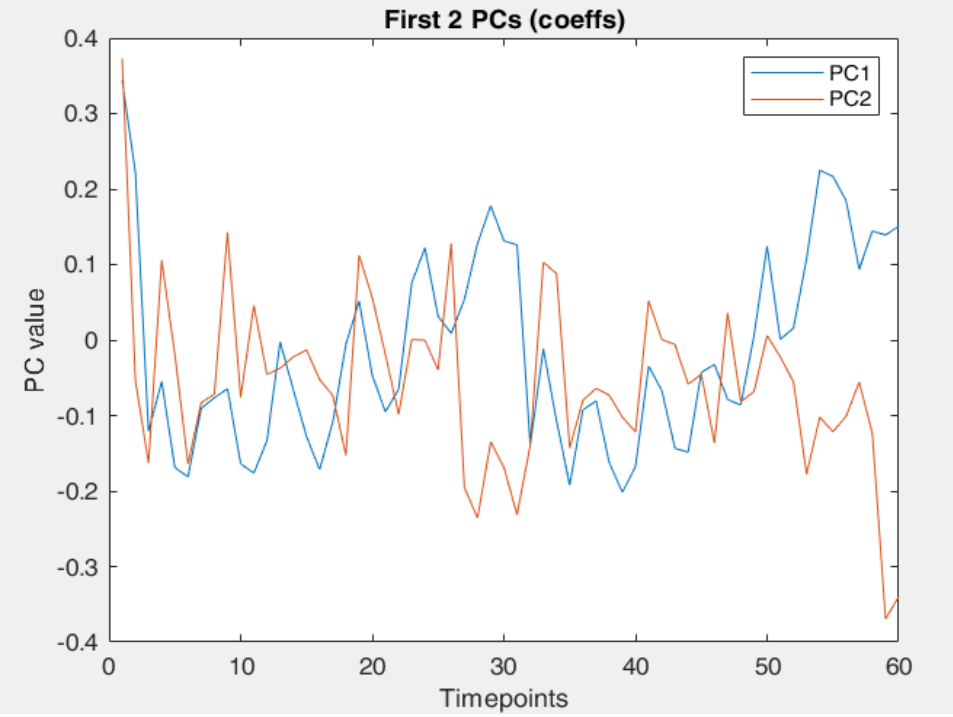
```
subplot(2, 1, 1);
plot(stimulus_0_av_pc1, 'b');
hold on;
plot(stimulus_1_av_pc1, 'r');
xlabel('Timepoints');
ylabel('Amplitude');
legend('stim0: face', 'stim1: car');
title('Trial-Averaged Scores for PC1');

% Plot trial-averaged signals separately for each stimulus for PC2
subplot(2, 1, 2);
plot(stimulus_0_av_pc2, 'b');
hold on;
plot(stimulus_1_av_pc2, 'r');
xlabel('Timepoints');
ylabel('Amplitude');
legend('stim0: face', 'stim1: car');
title('Trial-Averaged Scores for PC2');
```

This code performs a series of operations on the PCA output to analyze and visualize the data for two stimuli (face and car). Specifically, it:

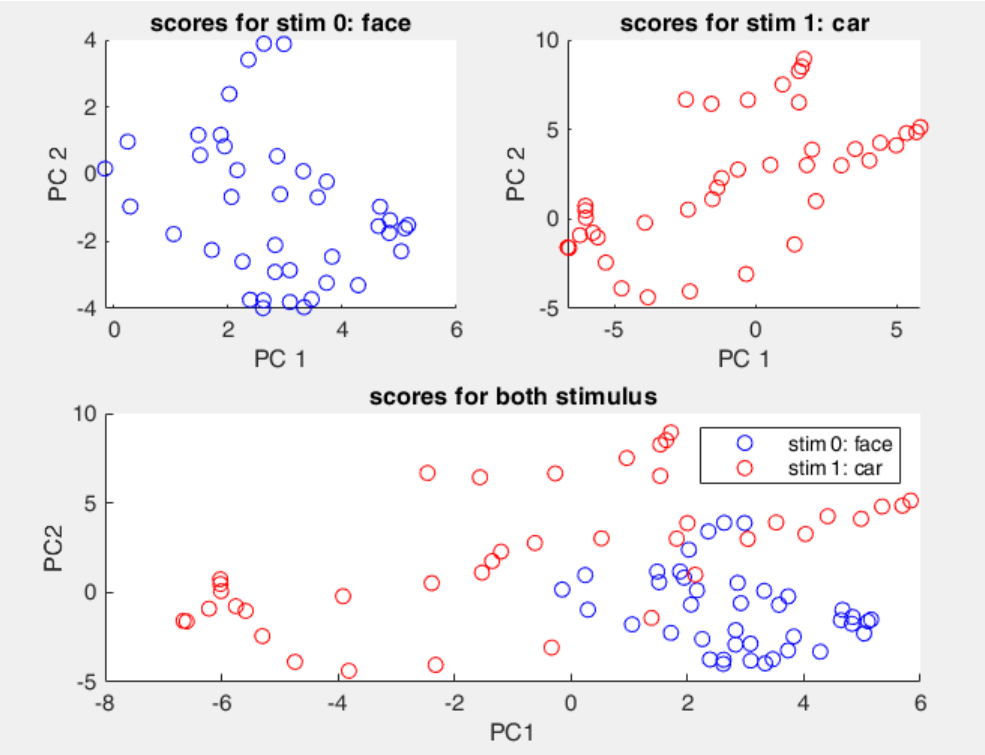
1. Finds the indices of the trials for each stimulus (**stim_0**, **stim_1**).

2. Extracts the PCA scores for each stimulus (**stim_0_score**, **stim_1_score**).
3. Computes the trial-averaged activations (scores) for each stimulus (**stim_0_score_mean**, **stim_1_score_mean**).

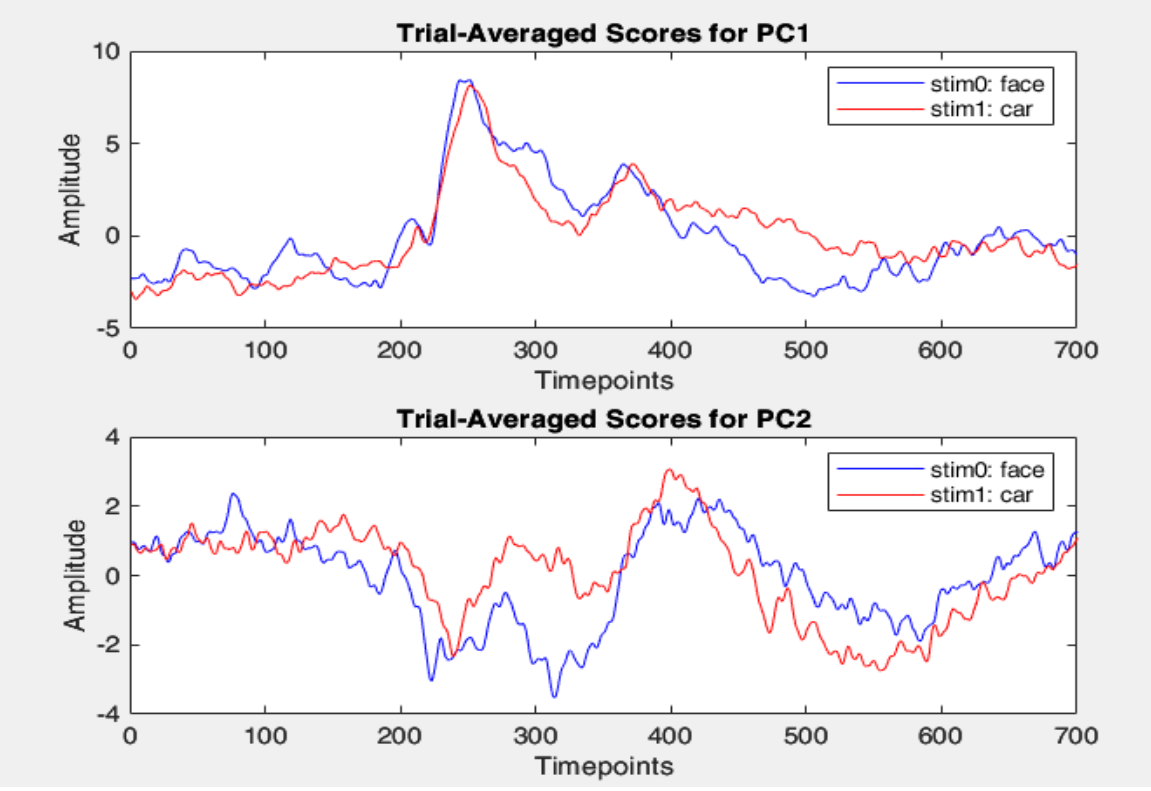
4. Plots the first 2 PCs (figure below).



5. Plots their trial-averaged activations for each stimulus (figure below).



6. Reshapes the PCA scores to trial format.
7. Computes trial-averaged signals separately for each stimulus for PC1 and PC2.
8. Plots the trial-averaged signals separately for each stimulus for PC1 and PC2 (figure below).



For the discrimination of the 2 stimuli:

- Based on the scatter plots of the first 2 PCs for each stimulus, we can see that the scores for the 2 stims are separated along both PC1 and PC2 axes. This is more distinct at later time points. So, we could say that there might be some degree of discrimination between the 2 stimuli based on these two components.
- Based on the trial-averaged signals we can say if the 2 stimuli are discriminated if the plots of the two signals for the 2 stimuli diverge. As we can see this happens at around 400-580 ms for PC1 and at around 280-380 ms for PC2. Separately for each stimulus for PC1 and PC2, it seems that discrimination of the two stimuli is possible at around 300-500 ms after stimulus onset. This is more distinguishable at around 500 ms. However, the discrimination of the two stimuli is not as clear in the PC2 plots as it is in the PC1 plots. This suggests that PC1 carries more information regarding the differences between the two stimuli.

However, to determine if this separation is statistically significant and at which time windows, further analysis is needed.

Part B

B1.

In this part we were asked to compute EEG classification performance (measured by the area under the ROC curve) Az in the following two time windows [201,250]ms and [451,500]ms, by using the single_trial_analysis.m Matlab function which was given.

```
%% B1: classification performance Az in two time windows

reshaped_data = reshape(EEGdata, [60,700*80]);

% Define time indices for the two time windows of interest
time_window_1 = 201:250;
time_window_2 = 451:500;

% Reshape data to match input format for single_trial_analysis
num_samples = length(time_window_1);

X1 = EEGdata(:, time_window_1, :);
X2 = EEGdata(:, time_window_2, :);

% Compute Az values for each time window
skipLOO = 0; % with leave-one-out cross-validation
duration=num_samples;
```

```
[Az1] = single_trial_analysis(X1, stim, duration, skipLOO);
[Az2] = single_trial_analysis(X2, stim, duration, skipLOO);

% Print Az values
fprintf('In time window [201, 250]ms the Az is: %.2f\n', Az1);
fprintf('In time window [451, 500]ms the Az is: %.2f\n', Az2);
```

This code performs classification analysis by computing the area under the ROC curve (Az) to evaluate the ability of the EEG data to discriminate between the two stimuli (stim 0 is for faces and stim 1 is for cars) in two time windows of interest: [201, 250] ms and [451, 500] ms. To do that, we must reshape again the data to match the input format for the single_trial_analysis function, which performs the classification analysis using leave-one-out cross-validation.

And these are the Az values for the two time windows as shown in the command line:

```
In time window [201, 250]ms the Az is: 0.67
In time window [451, 500]ms the Az is: 0.93
```

B2.

In this part we were asked to plot an Az curve across time to illustrate how stimulus classification evolves over the duration of a 700ms trial.

```
%% B2: Az curve across time to illustrate how stimulus classification evolves over
the duration of a 700ms trial

% Define some variables
window_size = 50;
step_size = 1;
num_windows = size(EEGdata, 2) - window_size;
timepoints = window_size+1:step_size:700;
skipLOO=0;

% Initialize the Az array to store Az values
Az_array=[];

% Loop through each window and compute Az
for k = 1:step_size:num_windows

    % Extract the data within the current window
    data_window = EEGdata(:, k:k+window_size-1,:);

    % Reshape the data for input to single_trial_analysis
    data_window=reshape(data_window,[size(data_window, 1),win-
    dow_size*size(data_window, 3)]);

    % Compute Az using single_trial_analysis
    Az = single_trial_analysis(data_window, stim, window_size, skipLOO);
    Az_array=[Az_array,Az];
```

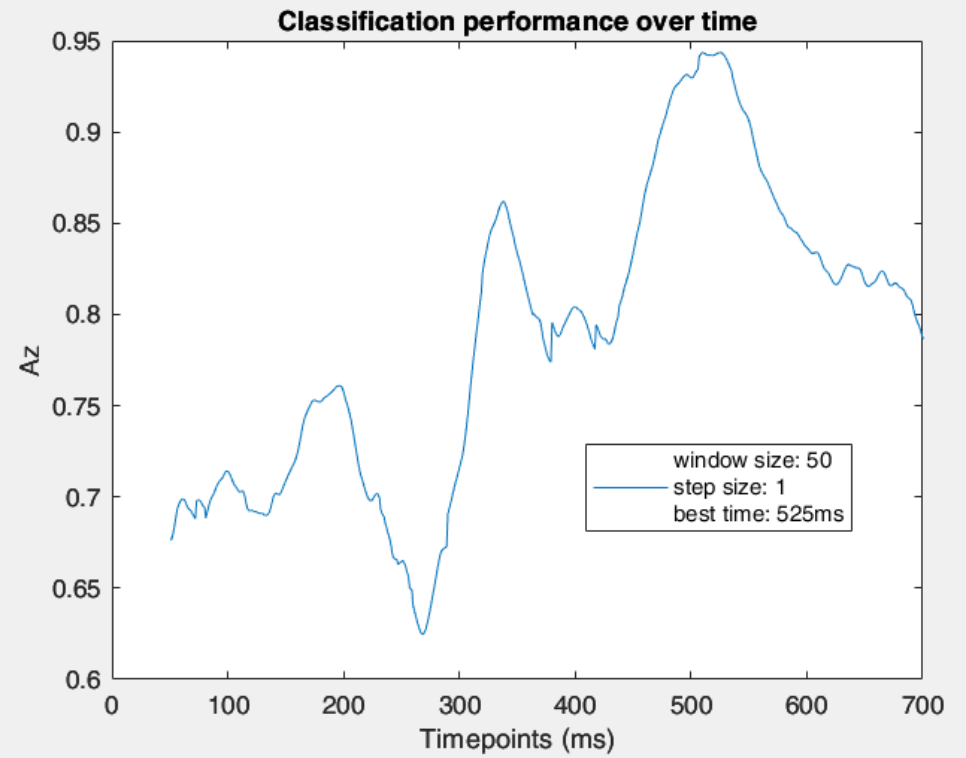
```
end

% Find when the highest Az value occurs
[pks, locs] = findpeaks(Az_array);
[max_Az, index] = max(pks);
time = timepoints(locs(index));
fprintf('Best stimulus classification is at %dms\n', time);

% Plot the Az curve
figure;
plot(timepoints,Az_array);
xlabel('Timepoints (ms)');
ylabel('Az');
title('Classification performance over time');
text = sprintf('window size: %.f\nstep size: %.f\nbest time: %.fms', window_size,
step_size, time);
legend(text,'Location','best');
```

This code computes the Az values for stimulus classification using sliding windows of 50 timepoints (ms) with a step size of 1 timepoint. Specifically, in this script I divided the EEG data into non-overlapping windows of size window_size, and then slided a window of size window_size across the data with a step size of step_size in order to compute Az values for each window. Of course, the window_size and the step_size must take other values too, to observe the differences. In addition, I plot the evolution of the Az values over time to visualize how stimulus classification performance changes throughout a 700ms trial and finds the timepoint with the highest Az value. Specifically:

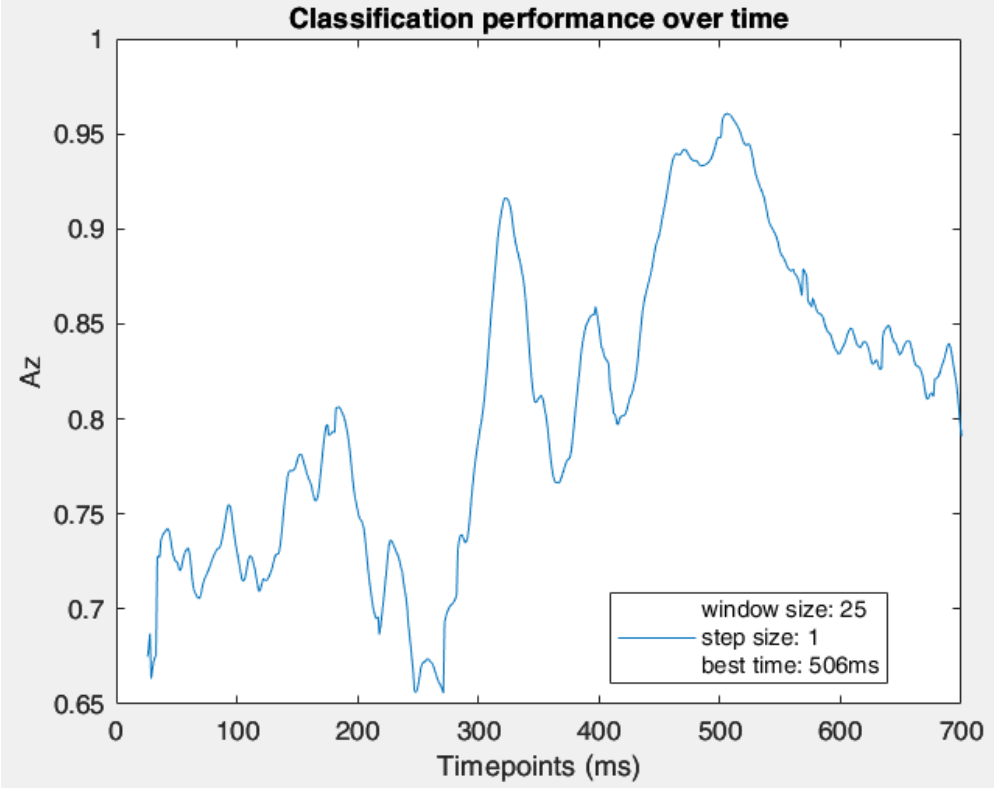
1. **window_size** and **step_size** are defined as the size of each window of data and the step size between consecutive windows, respectively.
2. **num_windows** is the number of windows that can be generated from the EEG data and is calculated by subtracting window_size from the total number of time points in the data.
3. **timepoints** is a vector that contains the time points at the center of each window. This vector starts at window_size+1 and increments by step_size until it reaches the end of the EEG data.

Then, there is a for loop which iterates through each window of data and computes the corresponding Az value using the single_trial_analysis function. And finally, there is a plot of the Az values across time.
I have chosen to plot my Az curve across the 700ms.

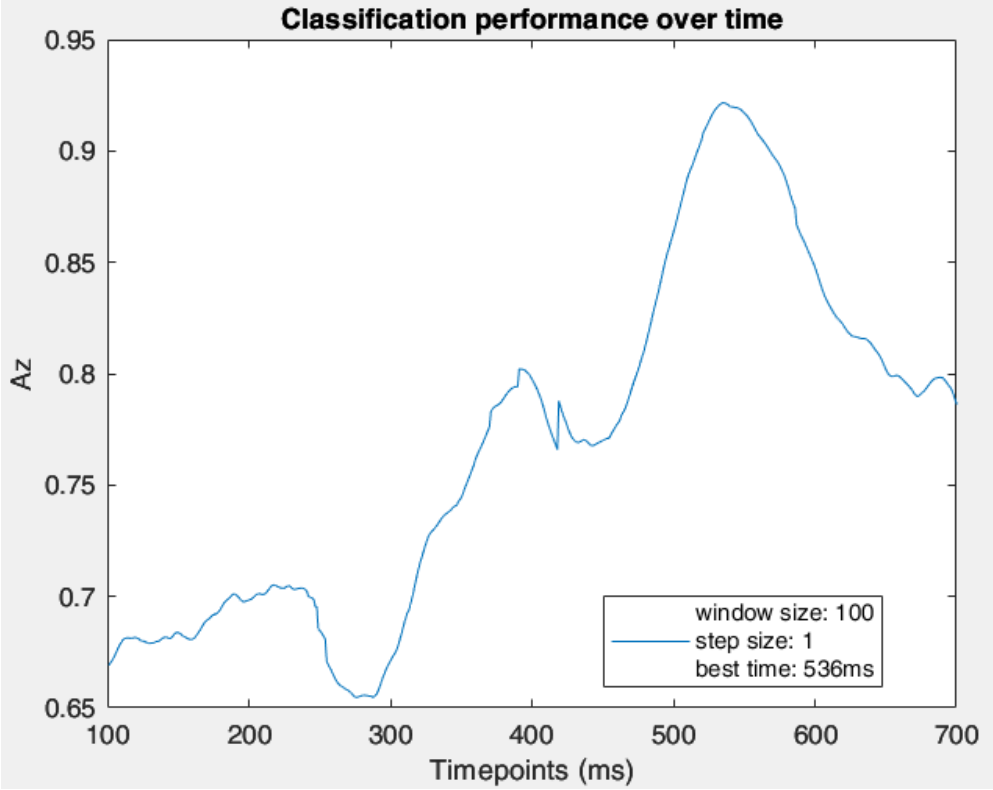
Here are some plots with *different* values for the *window_size* and *step_size=1*:



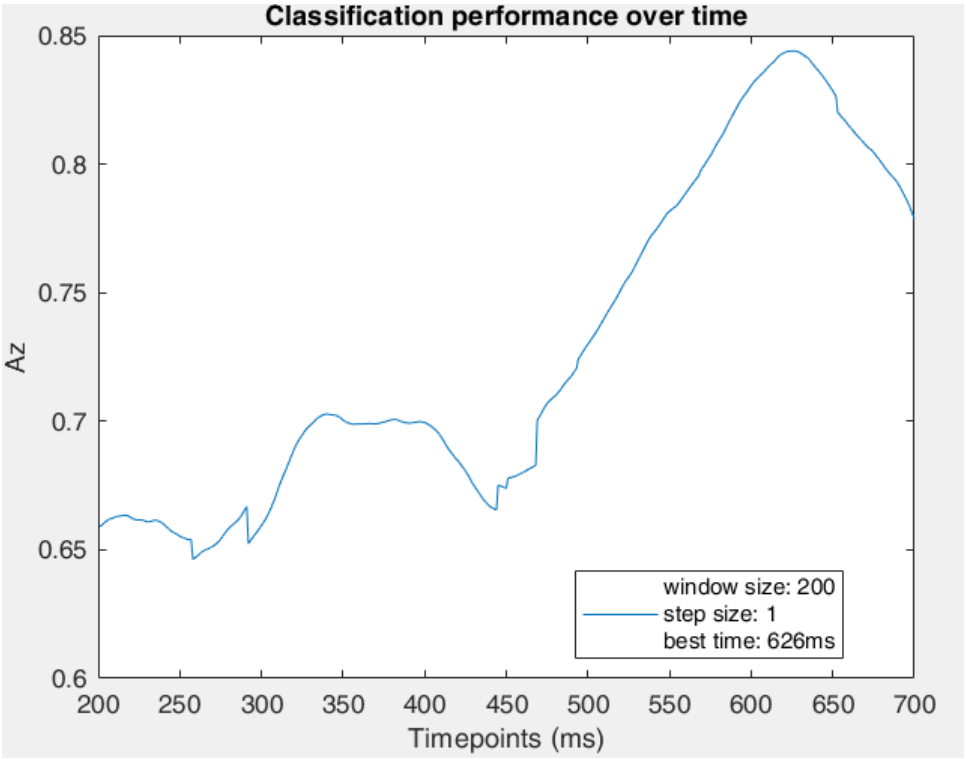
425 ms after stimulus presentation we get the best stimulus classification from the EEG signals



406 ms after stimulus presentation we get the best stimulus classification from the EEG signals

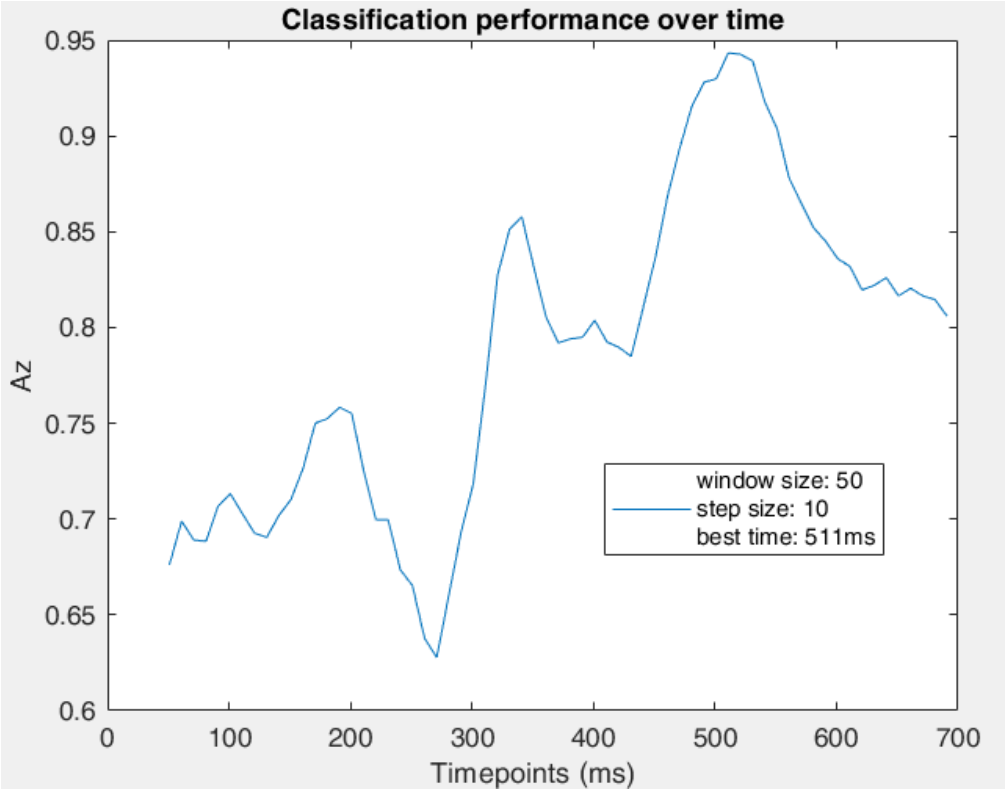


436 ms after stimulus presentation we get the best stimulus classification from the EEG signals

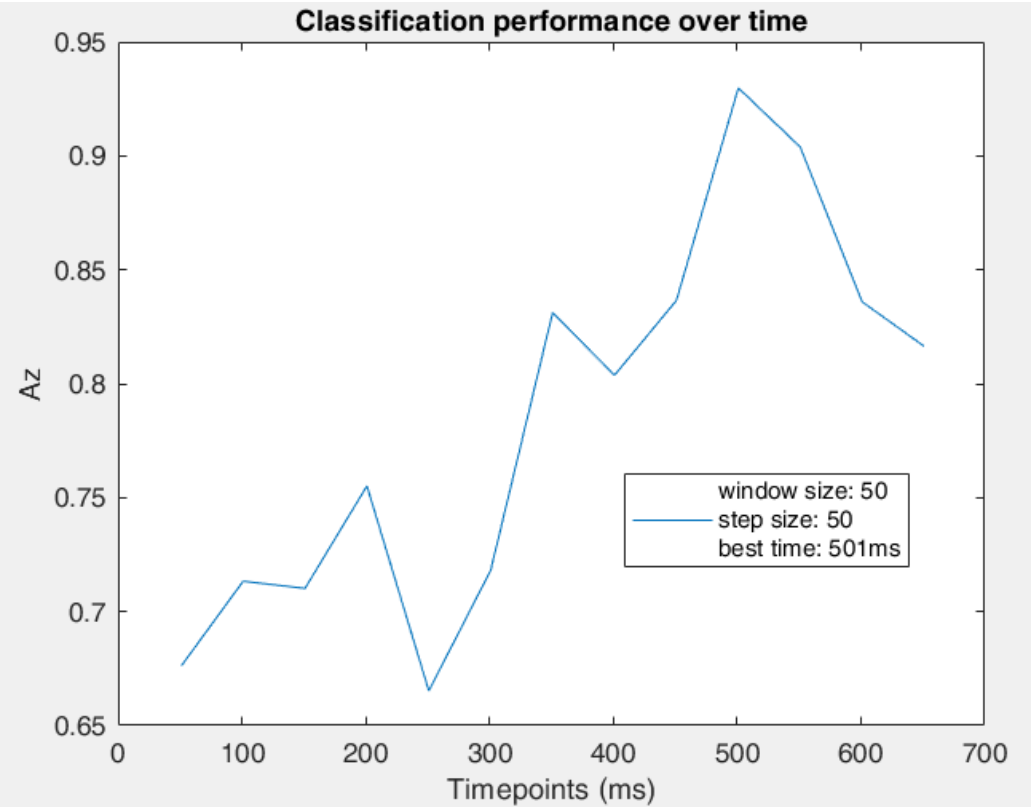


526 ms after stimulus presentation we get the best stimulus classification from the EEG signals

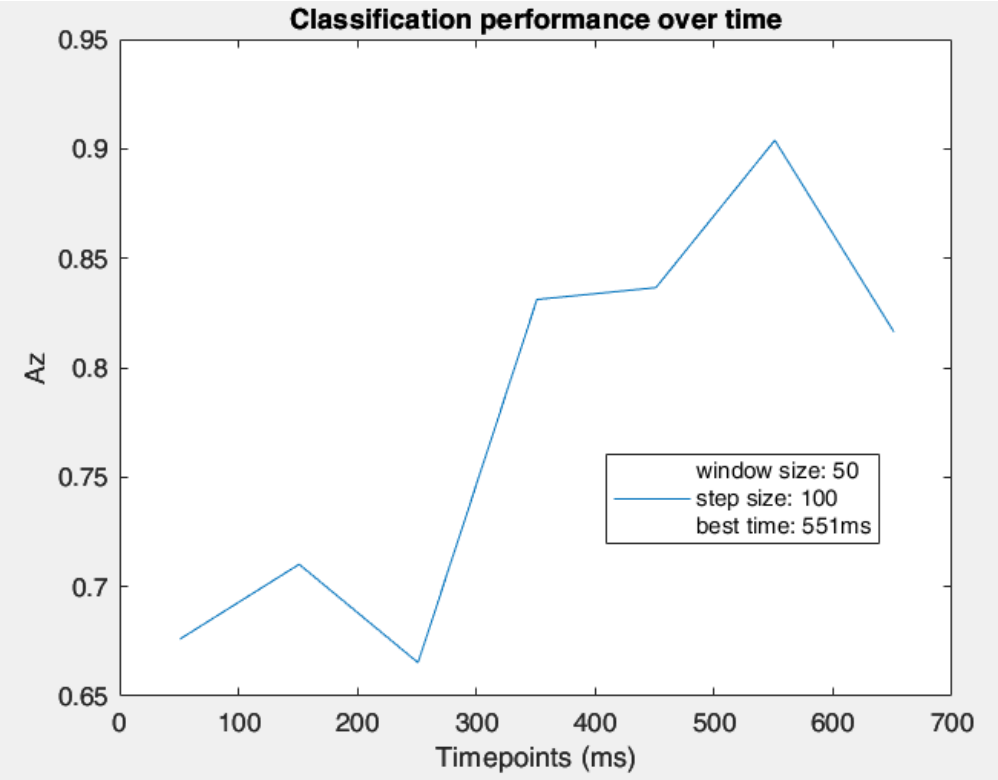
Here are some plots with *different* values for the *step_size* and *window_size=50*:



411 ms after stimulus presentation we get the best stimulus classification from the EEG signals



401 ms after stimulus presentation we get the best stimulus classification from the EEG signals



451 ms after stimulus presentation we get the best stimulus classification from the EEG signals

Note: The time that we get the best stimulus classification from the EEG signals after stimulus presentation is 100ms before the time we get the peak, because stimulus presentation starts at the 100^{th} ms.

In general, we could say that if we increase the window size the resulting Az values may be more stable across time, as each window of data contains more information. However, there is a disadvantage that there are fewer time points at which Az values are computed. On the other hand, if we increase the step size, the resulting Az values may not be that accurate because there are fewer time points at which Az values are computed. Moreover, by increasing the window size or the step, we get slower accuracy in our classification.