P300 Evoked Potential (EP) feature extraction

Exercise 2.1: P300 Evoked Potential

In assignment #1 we computed target and non-target P300 curves by averaging individual EEG segments (trials). Using the previously developed code, let's continue with the P300 analysis…

Exercise 2.2: Calculate the standard deviation of the mean

Averaging the EP curves is key to improve the signal-to-noise ratio and visualize the target vs non-target response. However, plotting only the average, eliminates the information on the *variability* of these curves among trials. To get a better understanding of the distribution of the EPs and whether or not individual curves are different, we want to know the typical deviation of the mean (a.k.a. standard deviation (SD)).

Task:

- Compute the standard deviation (SD) of the EP's for every time-point of each condition of exercise 2.1 (the MATLAB function `std` computes the standard deviation).
- Integrate plots of the standard deviation in the plots of exercise 2.1 (`hold on`). To plot the standard deviations at each time point, you can use the Matlab function `errorbar`.

Exercise 2.3: Statistically significant difference?

Now we want to test whether or not target and non-target EP are different. Use the Wilcoxon test (Matlab `ranksum`) to compute a statistical test for each time point of each channel between non-target and target. Wilcoxon is suitable as the amount of trials is different between the conditions and the distribution of the trials is unknown. Use a significant level of α=0.01. Do not forget to correct for multiple comparisons (Bonferroni correction, divide significance level by the amount of comparisons).

Task:

- At which time points and which channel are the two modalities (target, non-target) from exercise 2.1 significantly different according to the statistical test? Make plots!
- Where is the biggest coherent cluster of differences in each channel?

Linear Discriminant Analysis (LDA) classifier

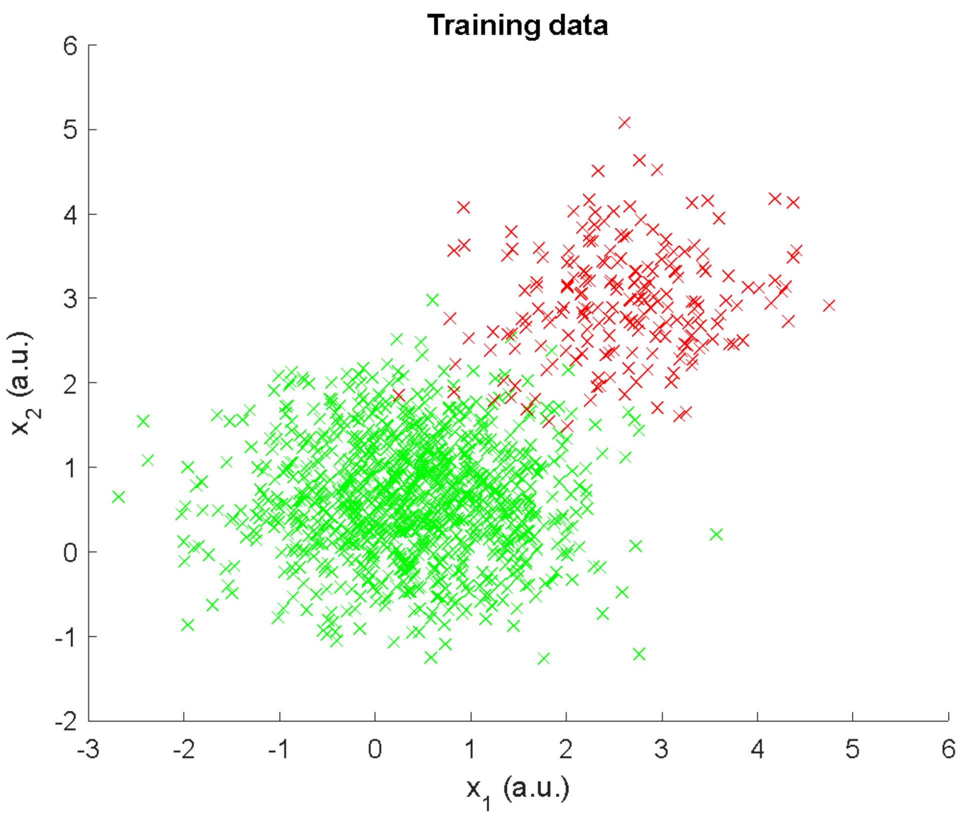
Exercise 2.4: Generate artificial training and testing data

Before implementing the LDA classifier, we want to create artificial training and test data sets. The data will be useful to test and evaluate the LDA implementation. Create a two-dimensional test and training data set for two different classes. Data for the individual classes should be drawn from two different normal distributions (`randn`). Use a center of 0.4/0.6 (dim 1/dim 2) and a standard deviation (SD) of 0.9/0.7 for the first class. For the second class, use a center of 2.5/3.0 (dim 1/ dim 2) and a SD of 0.9/0.7. For the LDA training, generate 1000 trials for class 1 and 800 trials for class 2. For the LDA testing, generate 500 trials for class 1 and 400 trials for class 2.

Task:

- Create matrices `X_train` and `X_test` that contain the trials in the columns and the dimensions in the rows.
- Create column vectors `Y_train` and `Y_test` that contain the corresponding class labels.
- Visualize the distributions (`scatter`).

Toy example:



Exercise 2.5: Implementation of a custom LDA

The idea behind linear discriminant analysis (LDA) is to use hyperplanes to subdivide the feature space in class-specific subspaces. Assuming the normality **and** homoscedasticity of the class distributions, in the case of a binary decision (2 subspaces), the classification of a given sample x is performed by

$$sign(w^T x + b), \qquad (1)$$

where b denotes the bias (offset of the hyperplane) and w the projection vector. The projection vector can be calculated by

$$w = \Sigma_c^{-1}(\mu_1 - \mu_2), \qquad (2)$$

where μ_i denotes the (column) vector of estimated means per feature dimension of class i, and Σ_c is the estimated common covariance matrix (i.e. the weighted average of the individual class covariance matrices). The bias is given by

$$b = -\frac{1}{2}\mu_1^T\Sigma_c^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma_c^{-1}\mu_2 + ln(\frac{P_1}{P_2}) \quad (3)$$

where μ_i and Σ_c are the same as before, while P_1 and P_2 are the *a priori* probabilities of classes 1 and 2. If not known, they can be estimated from the training set as the ratio between the number of samples for that class and the total number of samples in the training set.

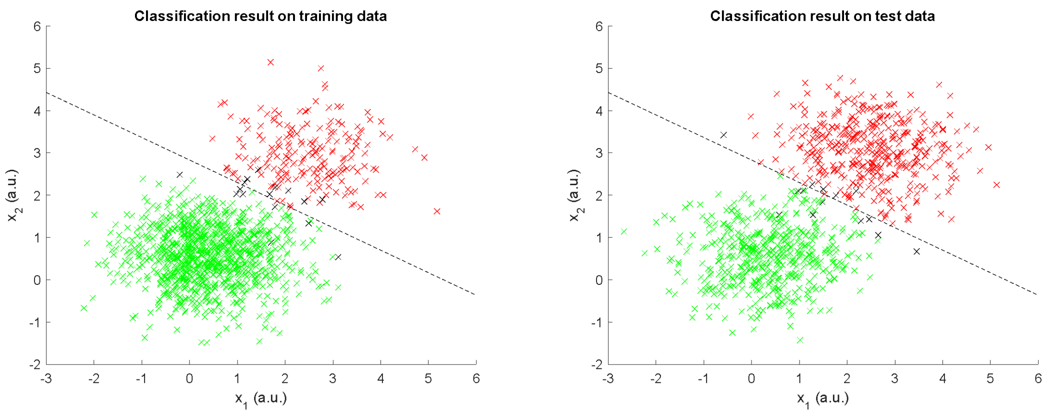
Important notes:

1. please note that the order of μ_1 and μ_2 in equation (2) and (3) is important, as it determines the direction of the unit vector orthogonal to the separation hyperplane, and therefore which class (1 or 2) corresponds to the positive or negative sign in equation (1)
2. since the "a priori" probabilities in equation (3) might be estimated from the training set, and this influences the bias b, it is good practice, when training the LDA classifier, to divide the *training* and *test* set in order to have the same proportion of class 1 and 2 in each of them – i.e. similar "a priori" probabilities

Implement an LDA classifier (function name `custom_LDA`) based on the above equations.

Task:
- Use your artificial test data to test the LDA classifier and plot your results (`scatter`)
- Highlight the decision hyperplane (in the two-dimensional case a line).
- What accuracy do you get?

Toy example:



Exercise 2.6: Classification of EP training and testing data

Create training and testing data out of the real AEP data that we used in the previous assignment (`BI5_segments_HTS.mat`). Take the time point of channel Cz with the lowest p value (most significantly different) at around 300ms for dimension 1 and the most significant time point at around 500ms for dimension 2. Visualize the distributions (`scatter`). Divide the data as following: 900 non-targets and 200 targets for training; 900 non-targets and 200 targets for testing. Select the trials using random subsampling.

Task:

- Create the same data structure as in exercise point 2.4 (`X_train`, `X_test`, `Y_train`, `Y_test`).
- Check if the data are normally distributed to make sure they are suitable for classification with LDA. Use `normplot()` for each of the classes (target, non-target) and dimensions separately.
- Use your test data to test the LDA classifier and plot your results (`scatter`)
- Highlight the decision hyperplane (in the two-dimensional case a line).
- What accuracy do you get?

Exercise 2.7: Classification of EP training and testing data on the recorded data

Repeat the steps on exercise 2.6 on the acquired data from the lab. Take the time point of channel Cz with the lowest p value (most significantly different) at around 300ms for dimension 1 and the most significant time point at around 500ms for dimension 2. Visualize the distributions (`scatter`). Divide the data as following: 900 non-targets and 200 targets for training; 900 non-targets and 200 targets for testing. Select the trials using random subsampling.

Task:

- Create the same data structure as in exercise point 2.4 (`X_train, X_test, Y_train, Y_test`).
- Check if the data are normally distributed to make sure they are suitable for classification with LDA. Use `normplot()` for each of the classes (target, non-target) and dimensions separately.
- Use your test data to test the LDA classifier and plot your results (`scatter`)
- Highlight the decision hyperplane (in the two-dimensional case a line).
- What accuracy do you get?