 Aviv Dotan, Daniel Poliakov, Nir Getter, Erez Wolfson

Final Project – Analysis of Motor Imagery Data

Background

- This project will cover the fundamentals of the world of Machine Learning (AI, Data science or any other buzzword of your choice).
- The project focuses on a dataset of cued motor imagery EEG data.
- The PDF file "motor imagery data" describes the data in detail (The file describes a paradigm of foot vs. hand imagination. In our data the idea is the same only with left vs. right hand imagination, and we don't have the 2 sec' fixation step).
- Briefly, in each trial the subject was asked to imagine motor activity in one of 2 classes: *left* hand or *right* hand. 160 trials were preformed, for each trial data from 2 EEG channels (C3 and C4) were recorded.
    - You are given the data of 128 of the trials with their corresponding labels (*left* or *right*) for *training*; the other 32 trials are given without their labels to be used as a *test* set.
- The training data is stored **`motor_imagery_train_data.mat`** as a struct called **`P_C_S`**.
    - The sampling rate is stored in the **`samplingfrequency`** field.
    - The acquired data are stored in the **`data`** field, of dimensions $128 \times 768 \times 3$. The first dimension represents the trials, the second dimension represents time samples (6 sec with sampling rate of 128Hz) and the third dimension represents the channels. Channel 1 is electrode C3, Channel 2 is electrode C4 and Channel 3 is a trigger channel (not relevant for this project).
    - The correct labels of each trial are found in the **`attribute`** field, of dimensions 4×128. The second dimension represents the trials. The first dimension (rows) represents 4 possible labels (see the **`attributename`** field): **`ARTIFACT`** (1st row), **`REMOVE`** (2nd row), **`LEFT`** (3rd row), **`RIGHT`** (4th row). Only use trials where the attribute is **`LEFT`** or **`RIGHT`**.
- Your goal is to extract useful features from the data and then train a classification algorithm on these features to predict the label (*left* or *right*) on a single trial basis.
- First, you should explore the data and search for informative features. Once you identify such features, extract them and build your classifier.
- Once your classifier is ready, fiddle with the features to improve the classifier's accuracy.

Neural data-analysis Workshop

Aviv Dotan, Daniel Poliakov, Nir Getter, Erez Wolfson

1. Visualization

- Write a script that will allow you to visualize the EEG signal in a single channel for trials from a single class. For each class (*left* and *right*) draw a figure with 20 subplots corresponding to 20 random trials. Each subplot should plot the data from both channels (C3 and C4).
- Eyeball the data and see if you can identify qualitative differences between the different classes.

2. Power spectrum

- Calculate the power spectrum from all samples in each class: Calculate the spectrum using Welch's method (use MATLAB's **`pwelch`** function).
  - Make sure to use time windows of at least 0.5 seconds (preferably over 1 second) to allow for a good estimation of the power in low frequencies.
  - Plot a spectrum for each class and each channel separately, total of 4 spectrums. Use subplots to present the results in a meaningful way.
  - DO NOT use samples from the entire trial, remember that motor imagery took place only in a specific time segment. Extract only the samples from the relevant time window.
- Compare the power spectra of both classes. Are there any frequency bands that seem useful for separating the classes? Plot both spectra on a single graph to see the difference (do this for C3 and C4 separately).
- Look at the spectrogram of the data (see class ex. 3) and use it too to identify informative frequency bands. Use time windows of at least 0.5 seconds (preferably over 1 second).
  - Find a way to use spectrograms to obtain a meaningful representation of the spectrum for different conditions. *Hint*: this time you should use the **entire trial**, not just the imagery time window (explain why). Also, a different spectrogram for each trial is NOT a good idea.
  - For each informative frequency band, calculate the energy in this band for each trial in the relevant time window (the power of each band per trial is 1 scalar value). Prepare a figure with histograms depicting the energy distribution for each of the 2 classes.
  - One way to calculate the energy in a frequency band is to calculate the area under the power spectrum in the relevant frequency range. Use MATLAB's function **`bandpower`** (do NOT split into frequency bands by yourself like you did in ex. 5).

3. Classification

3.1. Feature extraction

- Extract from each trial the informative features you found by comparing the spectrograms.
- Add any features you think might help the classification. You may use features from ex. 5 or any other features. Make sure to add **at least** 2 more features.
- Visualize the trials in the feature space in 2D/3D using a dimensionality reduction method of your choice (e.g. PCA (MATLAB's **`pca`** function) or t-SNE (MATLAB's **`tsne`** function)), like in exercise 5. Color each trial according to its class (*left* and *right*). Is the data linearly separable?
- Make sure not to use more than 32 features, as it can result in overfitting the training data.
    - If you would like to use more than 32 features, apply some method of feature selection or dimensionality reduction to obtain a lower number of features.
- Fiddle with these settings until you find the features that yield the best results.
- Try classifying with a different set of features (Show 1 example of results using only 1 feature and 1 example of results using more than 6 features). Compare the results.

3.2. Training a classifier

We will focus on linear discrimination using Linear Discriminant Analysis (LDA).

- Use MATLAB's **`classify`** function to train a classifier on part of the data and then test its performance on the rest of the data. Make sure the algorithm used by the **`classify`** function is indeed LDA.
- Use k-fold cross-validation to measure the classifier's performance:
    - Split the data into k-folds, then train k classifiers. Each classifier will use $k-1$ of the folds for training and the remaining fold for validation.
    - Choose k wisely, considering the variance of the results and the total run time.
    - Split the data randomly. Your results on different runs should change slightly.
    - **Important**: make sure you are not validating on the data that were used for training.
    - Explain why this is so important, what are the advantages and disadvantages of using more of the data for training? Why is it wrong to test for results on data that was used for training?

- Calculate the accuracy ('percent correct') of each of the k classifiers on its respective validation set. Report the average accuracy obtained and the standard deviation (as in 'Accuracy: 31.4±1.5%').
- Report the training and validation performance separately.

3.3. Test your classifier

Once you are perfectly happy with your feature extraction procedure (measured by the classifier's performance), train a new classifier using ALL the trials in the *training* data and measure its performance on the *test* set.

- Load the *test* data from **`motor_imagery_test_data.mat`**. It contains a **`data`** variable, structured exactly as the *training* data (except it has 32 trials instead of 128).
- Apply your feature extraction procedure to the *test* data. Make sure the process is **completely identical** to the one you applied for the *training* data.
- Apply your classifier to the features extracted from the *test* set to obtain a predicted classification (*left* or *right*) for each trial.
- On moodle, fill in your classifier predictions for the test set and get its accuracy.
  - Note: You can only upload your test predictions to Moodle **TWICE (once per user)**. Do that only after you are completely confident in your classifier. Don't worry, we won't take off points for an unexpectedly poor test accuracy, as long as you explain the reason.

4. Exercise deliverables

- Submit according to the submission guidelines in Moodle.
- Describe **briefly** what you have done and accompany your results with discussion.
- Be **creative** in your work. This is a project, hence there is more than one correct way to achieve the goal. Try different things on your way to the solution and see how each path affects your results.
- Your code should be readable and well commented, your report should contain all relevant figures with explanations of what you have done and the results you obtained (in a way that reflects your understanding). You will be graded on your code, your report, **and the quality of your results**.
- As always, feel free to consult with us on any subject.

Good Luck!