Project #4

Riccardo De Zen - 2019295
Mattia Fanan - 2019138
Niccolo' Turcato - 2021446

a.a. 20/21

Contents

1 Introduction **2**

2 Methods **2**

3 Results **7**

4 Discussion **7**

1 Introduction

This document has the goal to report our analysis of the provided EEG data, collected during a MI experiment, and the simulation of a BCI loop.
Data has been recorded with 16-channel EEG amplifier (g.USBamp, g.Tec) @512Hz. Electrodes were placed accordingly to the 10-20 standard layout (Figure 1).
Each subject participated in 3 recording days, performing “offline” (calibration, no real feedback) and “online” (with real feedback) runs.
The partecipants of the experiment were asked to perform two classes of MI tasks (both hands and both feet) while their EEG was collected, moreover some resting periods were also recorded.
The final purpose of a BCI such as this, is to train a system to recognize from EEG data, which MI task is performed.

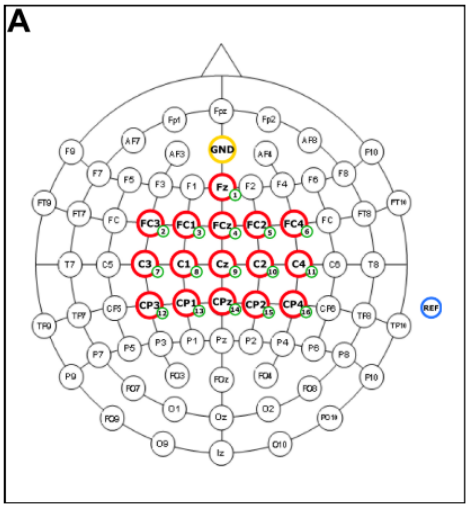

Figure 1: Electrodes layout

2 Methods

The implemented methodology follows the BCI literature.
Work is divided in two main tasks for each subject:

- Creation and calibration of a classifier based on ”offline” data
- Classification of ”online” data (serving also as a test)

Data manipulation: The provided data is in the form of raw EEG [samples x channels], and since in this state it is not useful for classifications, we used the procedure described in class to compute the corresponding PSD [windows x frequencies x channels].
Before applying the actual PSD procedure, the EEG data is first transformed into the laplacian referencing with a filter [channels x channels] (Figure 3) in order to enhance the localization of the events related to the executed tasks (Figure 2).

```
PSD_data = cell(size(patient.offline_files));
for f = 1:length(patient.offline_files)
    % Raw EEG
    [s, h] = sload(patient.offline_files{f});

    % Apply Lap and compute PSD
    PSD_data{f} = psd_extraction(s, h);
end
```

Figure 2: load EEG for offline data - compute Laplacian referencing and PSD for one patient

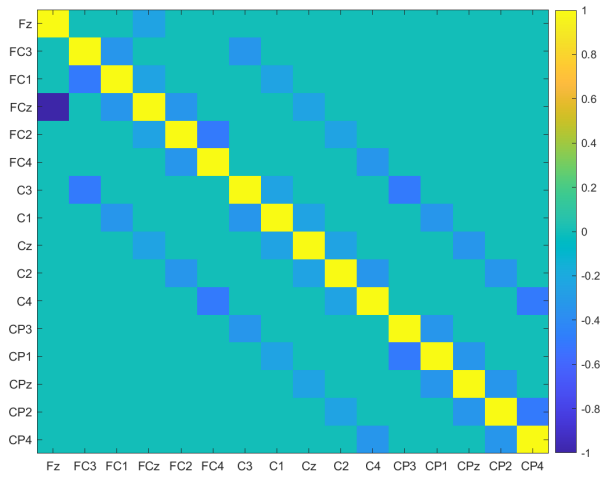

Figure 3: Laplacian filter (16x16)

After this we splitted the data into the offline part used to train an tuning and the online part used as test.

Offline task:

- **Feature selection:** For each patient, from the data in form of PSD, then the more discriminant features are selected, each feature corresponds to a (frequency, channel) pair. In this step, we take into account only if the pairs that are meaningful for MI tasks that are the beta [1] and mu [2] rhythms.
  In other words:
  - around the channels C3 and C4 for the task of both hands and around Cz for the task of both feet
  - in the mu band for the ERD [3] in the beta band for the ERS [4]

[1] for beta rhythm we mean the activity of populations of neurons in the motor-cortex region(also frontal region but is not related to motor-imagery tasks) approximately with frequencies 13-30 (can slightly change from subject to subject)

[2] for mu rhythm we mean the activity of populations of neurons in the motor-cortex region approximately with frequencies 8-13 (can slightly change from subject to subject)

[3] for ERD -event related de synchronization- we mean when some populations of neurons starts to fire in asynchronous way in response to a task to execute causing a decrease in power of the EEG signal in that area

[4] for ERS -event related synchronization- we mean when some populations of neurons after an ERD starts to re-synchronize their firing rate causing an increment in power of the EEG signal in that area

In order to correctly select the most discriminant features we have first normalized the distribution of the samples with respect to the two features we are interested in, applying a logarithmic transformation to the PSD matrix (Figure 4). Then we computed the Fisher's score (Figure 6, Figure 7) and for removing features that from the literature we know are not related to the tasks, we have applied weights (Figure 5) to them and then selected the best k (Figure 8).

Choice of k: We ran a version of subset selection [5] to choose the optimal number of features for the patients. We used offline data only, so that the results on the online data are as close as possible to a reasonable estimation of the true error of our classifiers, choice of k was in the range [1, 10]. For each patient, we divided the (offline) data in train and validation set (ratios $\frac{2}{3}, \frac{1}{3}$), obtaining the optimal number of features as well as the accuracy on validation data.
Then we had to choose if using foreach patient different number of features based on the results obtained as described, or choose k that would be applied to all patients. We went for the second option, as we felt that the tradeoff between optimality and complexity was more reasonable.
We computed a weighted average over the patients' results (with the best accuracy as weight) found that about 4 features is probably close to a global optimal number of features for our patients' data.

common data

here we suppose the frequecies of the PSD don't change between the files

```
info.frequencies = PSD_data{1}.frequencies;
```

label data

```
[cue_k, trial_k] = label_data(EVENT, length(PSD));
```

log and sub-frequences extraction

```
[PSD, info.frequencies]= extract_frequencies(PSD, info.frequencies, info.selected_frequencies);
PSD = log(PSD);
```

Figure 4: Extracting frequencies, labeling data and normalization

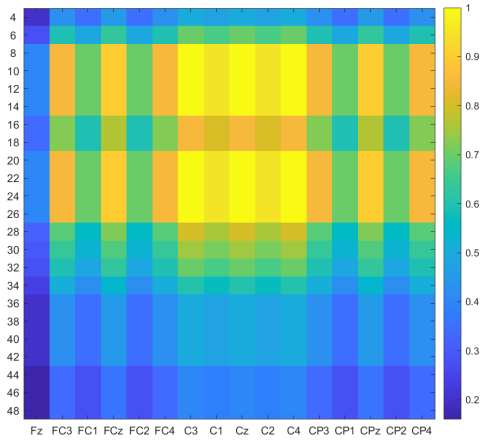

Figure 5: Features filter (23x16) (all patients)

[5] Insert Subset selection reference

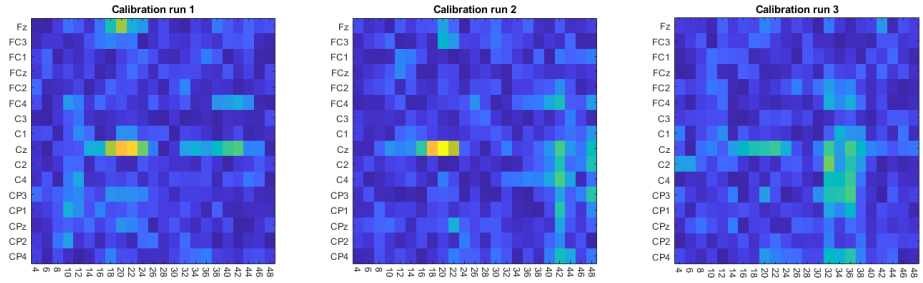

Figure 6: Fisher Score (16x23) (subject ai6)

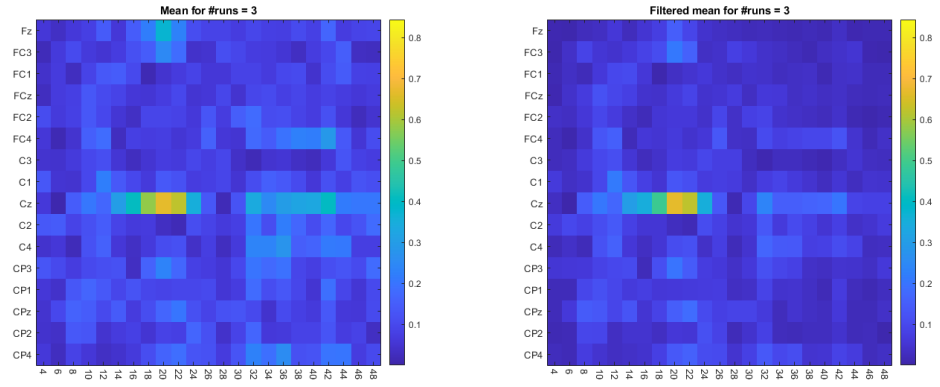

Figure 7: Fisher Score Average on 3 runs and filtered with Features weights (Figure 5) (16x23) (subject ai6)

- **Classifier training:** The previous steps allow to build a training set for each subject (one can use all the available offline data), by selecting the windows associated with the MI tasks. Then for each subject we applied k-fold cross validation to choose between three differ kind of classificators (LDA, QDA, SVM with rbf kernel). (Figure 10)

  At the end of this step, for each subject we have a trained classifier, that takes in input the selected features from a window of the PSD, and returns in output the label of the MI task that it recognizes.

- **Evidence accumulation tuning:** We implemented 3 evidence accumulation frameworks, based on different smoothing functions:
  - Exponential smoothing: $\Delta_{y_t} = \alpha(x_t - y_{t-1}), y_t = \alpha x_t + (1-\alpha)y_{t-1}$
  - Dynamic smoothing: $\Delta_{y_t} = \beta(\alpha F_{free}(y_{t-1}) + (1-\alpha)F_{bmi}(x_t)), y_t = y_{t-1} + \Delta_{y_t}$
  - Moving average smoothing: $y_t = \alpha x_t + (1-\alpha)\frac{\Sigma_{i=1}^{t-1}(y_i)+x_t}{t}$

```
%% Features selection

filtered_mean_fisher = mean_fisher_score .* features_weight;
selected_features = best_features(filtered_mean_fisher, num_features);
```

Figure 8: Features selection

extract training set

```
bh_index = (cue_k== info.cue_BH);
bf_index = (cue_k== info.cue_BF);

dataset = extract_features(PSD, selected_features);
train_set = dataset(bh_index | bf_index, :);
true_labels = cue_k(bh_index | bf_index);
```

Figure 9: Training set extraction

train classifier

```
model = train_binary_model(train_set, true_labels);
```

```
LDA obtained an average test accuracy of: 67.009022.
QDA obtained an average test accuracy of: 67.800139.
SVM (rbf) obtained an average test accuracy of: 68.299792.
```

Figure 10: Results of Cross validation on subject ai6

For each framework we performed parameters' tuning using a genetic algorithm [6]. We performed two different tunings:

- First tuning on the whole offline data, that is then tested on the online data
- Second tuning on the first run of online data, that is then tested on the rest of online data

Results are avalaialble in section 3.

Online task:

- **Feature extraction:** Given the online data we extract from it the same features we found in the offline data for the subject (Figure 12).
- **Classification test:** Once we have extracted the features, the classifier trained for the current patient is used to estimate the performed task, with a certain confidence value (Figure 13).

  At last, we can evaluate the performance of the system with single-sample accuracy.
- **Evidence accumulation test:** Then we used the evidence accumulation frameworks (with the parameters found in the tuning sessions for the subject) over the predictions of the classifier.

[6] Global optimization toolbox (Matlab)

train set results

```
predicted_labels = predict(model, train_set);
[accuracy, accuracy_per_class] = evaluate_classifier(true_labels, predicted_labels, classes)
```

```
accuracy = 74.0736
 accuracy_per_class = 1×2
      62.2814   85.8690
```

```
figure;
confusionchart(true_labels, predicted_labels);
```

Figure 11: Results of the trained classifier on training data (subject ai6)

log and sub-frequences extraction

```
[PSD, info.frequencies] = extract_frequencies(PSD, info.frequencies, info.selected_frequencies);
PSD = log(PSD);
```

load classifier data

```
load(strcat(classifiers_fold_root, patient.name, '_classifier'));
```

convert PSD into dataset for the classifier

```
dataset = extract_features(PSD, selected_features);
% features were selected in the script for training on offline data
```

Figure 12: Load of online data, trained classifier (for current subject), features extraction

At the end we evaluate the performance on the continuos feedback period in order to obtain a trial level accuracy.

3 Results

4 Discussion

test the model on online data

```
predicted_labels = predict(model, test_set);
[accuracy, accuracy_per_class] = evaluate_classifier(true_labels, predicted_labels, classes)
```

```
accuracy = 73.4216
 accuracy_per_class = 1×2
      74.8002   72.5689
```

```
confusionchart(true_labels, predicted_labels);
```

Figure 13: Results of the trained classifier on test data (subject ai6)