Università degli Studi di Padova

Neurorobotics and Neurorehabilitation A.A. 2020 2021

Corso di Masters'degree in Computer engineering

Project 5

Longitudinal analyses on MI BMI data

09 September 2021

Candidato:
Francesco Agostini, francesco.agostini.5@studenti.unipd.it
Matricola 2005833

:
Prof. Luca Tonin
Prof. Emanuela Formaggio
Prof. Maria Rubega

Anno Accademico 2020-2021+1

Contents

1 Introduction

The Project 5 is a real application of the techniques of EEG signal processing studied in class on a real case.

1.1 Description of the case

The case is the 2019 Cybathlon race, where a pilot have to interact with a game using a BMI, the pilot use 2 motor imagery to control the game interface. The motor imagery is both hands and both feet. In a period of 3 month the pilot in 11 sessions performed two kinds of runs. Offline where the pilot not have real feedback, used only for calibration, and Online session where pilot receive visual stimulus as a feedback of his commands. The pilot on the online runs can see on the screen two colored bars related to the imagery tasks. The colored bars is filled according with the output of the classifier on actual signals.

1.2 Data Description

The 11 sessions are composed with some gdf file, containing all information regarding the run type and ordered by timestamp:

- 20190502_F1 Executed in 02nd of May 2019 and composed by 3 offline runs and 3 online runs.
- 20190506_F1 Executed in 05th of May 2019 and composed by 1 long offline runs and 5 online runs.
- 20190513_F1 Executed in 13th of May 2019 and composed by 1 long offline runs and 3 online runs.
- 20190521_F1 Executed in 21th of May 2019 and it started with an online run, followed by 3 offline runs and closed with a last online run.
- 20190610_F1 Executed in 06th June 2019 and composed only by an online run.
- 20190624_F1 Executed in 24th June 2019 and composed only by an online run.
- 20190627_F1 Executed in 27th June 2019 and composed only by an online run.
- 20190701_F1 Executed in 1st July 2019 and composed by 7 online runs.
- 20190709_F1 Executed in 9th July 2019 and started with 2 online runs followed by 4 offline runs anc closed with a last online run.
- 20190711_F1 Executed in 11th July 2019 and composed by 2 online runs.
- 20190715_F1 Executed in 15th July 2019 and composed by 2 online runs.

1.3 Pipeline description

For this project the first important thing to do is configure the project with fundamental parameter used for all tasks, after that we can proceed loading data session keeping divided for each session online data from offline data. The main idea is to analyze offline sessions before computing fisher score and extracting channel and frequencies that could be used as discriminant for the classifier, create a classifier for each session, test classifiers with online runs measuring the performance. On Figure 1 we can see the 4 main component that compose the pipeline. I used factory/builder pattern to approach the problem, During the first phase we have to process data in order to identify a good discriminant for feature selection, during this phase we'll use only the first part of the pipeline for analyze offline sessions. Only in the second phase after feature selection will use the complete pipeline.

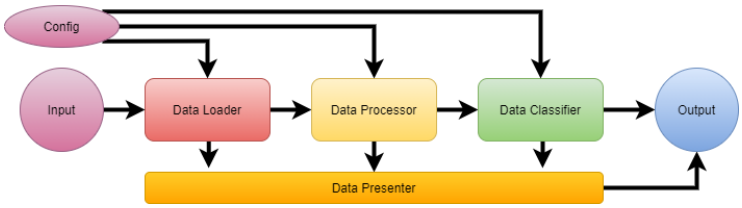


Figure 1: Pipeline description.

- Data Loader: Responsible of data loading and early processing. This component is a standalone class with collection container for the data in the gdf files. Another important responsibility of this class is the early data processing with spatial filtering and spectrogram computing.
- Data Processor: The class responsible for the data processing and manipulation during analysis procedure. This class do all step necessary to extract the information necessary for the feature selection and classifier training.
- Data Classifier: This is the last component of the pipeline, another class with all method necessary for the dataset (trainset and testset) and classifier train test operation.
- Data Presenter: The only class of the pipeline parallel to all component with only static methods. Each component of the pipeline have a reference to this class, in order to call necessary method to plot output and chart. This component of the pipeline not have class member except the output configuration.

The pipeline was created for allow to each module easy access to output of previous module. Data processor has a reference of data loader class for get data, and data classifier have a reference to data processor. With this method each component can easily access to the past history of the data.

2 Methods

All method used in this pipeline come from the classroom exercise and home works. But are organized and structured in order to work with different datasets just changing configuration on the main script and inserting session data in the right place.

2.1 Cofiguration

For this project following the data specification we choose the following configuration parameter. The first part of the configuration is used to set the necessary parameter to spectrogram computation. The second part of the config is dedicated to the classifier setup after feature selection procedure in phase 1. The last config part is dedicated to the Data Presenter class, and allow to choose which plot show during pipeline. Datapath var is the root source data folder that will used by the project, and var f is the frequency vector (from:step:to)

```
%% CONFIG PROJECT
datapath        = "../data/";     % Folder with sessions
    data
f               = 4:2:48;         % SelFreqs
% Spectrogram params
ml              = 1;              % mlength
wl              = 0.5;            % wlength
ps              = 0.25;           % pshift
ws              = 0.0625;         % wshift
wc              = 'backward';     % winconv
% Feature selection params
sc              = {'C4','FC2'};
sf              = [22 22];
% output configuration > (0 = Not Display | 1 = Display
    if possible)
display_input_data           = 0;
display_erd_ers              = 0;
display_fisher_score         = 0;
display_classifier           = 0;
display_accumulated_evidence = 0;
```

2.2 Data Loading

Data loading procedure starts looking with listSession() function for all subfolder in datapath, exluded . and .. for obvious reason. Each folder contained in datapath will be considered a session for the system and will be sorted by name. In our case with the timestamp at the beginning of each session folder the system will be sort in chronological order. After session listing the DataLoader class procede iterating each session (folder) with loadSessions() function and start to load the each session with loadSession() function.

2.3 Data Chaining

The loadSession() function start enumerating all gdf files and procede loading gdfs file using sload function. For each runs the function use the preprocessing function in order to do spatial filter operation and spectrogram computing. After this preprocessing operation the function chain all event vectors TYP (type) DUR(Duration in sample) POS(Position in sample). Keeping splitted offline runs from online runs. These information are stored as class member with name sessionsDataOffline(sessionIndex) that contain all chained data of offline runs, and sessionsDataOnline(sessionIndex) that contain all online runs information.

2.4 Spatial Filtering

For the spatial filtering the laplacian filter is used with function contained in laplacian16.mat file. This operation is necessary for a better comprension of data across the EEG channels during the spectrogram computation. In fact differently from Common average reference tecnique that introduce 'noise' from near channels, laplacian filtering approssimate the 2nd order derivative subtracting the mean acrivity of the neighbor channels. The result is an highlited spatial location of SMR.

2.5 Spectrogram and PSD

Using the spectrogram compute function provided in clas,s the spectrogram of each run is computed and stored in vector named P, as TYP DUR POS events information, the spectrogram vector are chained and inserted in the appropriate online/offline session vector, ready to be used from the next step in DataProcessing class.The spectrogram computing are necessary for enhance the spectral resolution in order to improve performances making a spectro temporal analysis of the signal, obtained from spatial filtering output. Using Welch's method configured with projects parameter we can compute power spectral density over time. This operation add a new dimension to the original dataset, giving us the possibility to extract more information from EEG signals.

2.6 Feature Selection

Now all necessary information for feature selection is ready and available in the DataLoader class, and DataProcessing class can start to run using the DataLoader instance for session data fetch. Differently from DataLoader the functions for elaborate offline and online session are splitted, because not all processing function needed for feature selection on offline sessions are necessary for online sessions. The class start with Cue and Fix vector preparation based on the events vector from DataLoader, now the DataProcessor is ready to do ERD/ERS data extraction necessary for analysis of event sinchronization and desinchronization. The project is now ready to plot the first output Power Spectral Density over time (Figure 2). This first output give us a first idea about the interesting frequency for feature selection. After Log operation on vectors the pipeline proceed computing fisher score and showing it using DataPresenter's plot function. Fisher score generate a feature matrix where each cell represent

the distance between distribution of the two feature classes. With this method we can see frequencies and channels with higher distance between classes, that allow us an easy classification. Finally we are ready to understand which channel and frequency are good choice for the classifier as discriminant. The result of this analysis on offline sessions give us the parameter to configure the classification operation in the project configuration. All extracted data are saved in different class member splitted in offline and online sessions, and accessible as a vector by using session index.

2.7 Dataset Creation

Now finally we can start with classification operation, as first task we have to create datasets for train and test operation. For each session the offline session log data from DataProcessor class, are used to create classifier train datasets, online session are used to prepare test datasets for models validation.

2.8 Model Training

At the end of dataset creation procedure the modelCreation() function are invoked for all session that provide appropriate train dataset, after model creation the function try to create a single model, trained with all train dataset and used only for final discussion and comparison. Every model were stored in Models(sessionIndex) class member that contain the model instance. DataPresenter class if configured show for each model the related plot with the feature cluster, giving us the possibility to understand if the model is able to find a discriminant between the given features. This operation is the first checkpoint to understand if we did a good job with feature selection. In Figure 5 we can see the classification discriminant output on each offline runs.

2.9 Classifier Test

After model creation and plot we can start with model accuracy validation using test datasets. Testing procedure starts using test dataset of each session on the same session's model. If the session is composed only by online runs the test are done on the last valid model of previous sessions.

2.10 Performance on Test

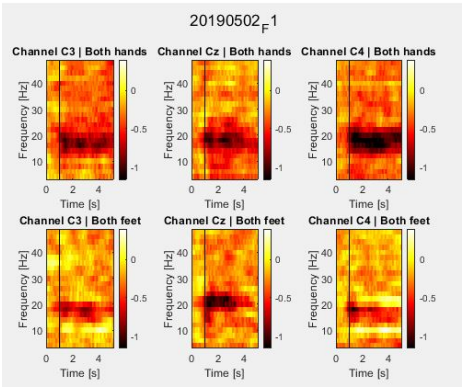
At the end of the pipeline, the computation of performance parameter is done in order to provide a performance parameter of each classifier. Using evidence accumulation framework and plotting results for each test sessions.

3 Results

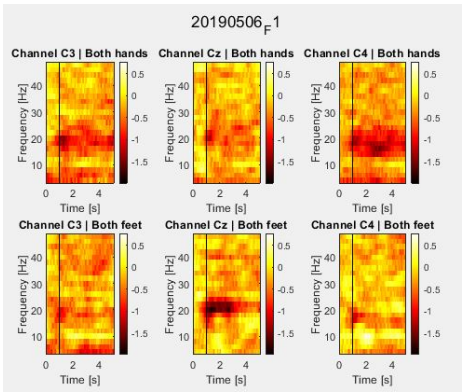
Follow the project results plots described in previous section.

3.1 Feature Selection

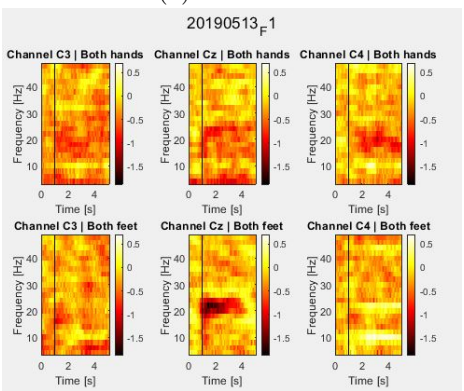
Power spectral density over time plots obtained from DataProcessing class.



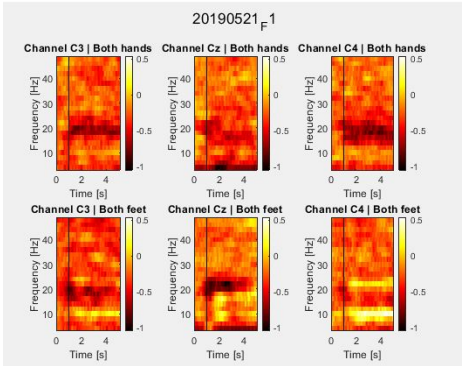
(a) 20190502



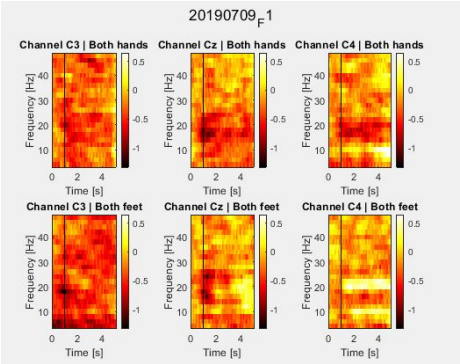
(b) 20190506



(c) 20190513



(d) 20190521



(e) 20190709

Figure 2: PSD Plots

3.2 Fisher Score results

From the fisher score result is possible to highlight good discriminant between desired imagery motor in frequency 22Hz and from FC2 and C4 signals. Considering single case we can consider other frequencies like 20Hz and 18Hz and Channels Like FCZ. But this discriminant are present only in some runs, differently from C4,FC2 on 22Hz that are almost present in each offline calibration sessions. (Figure 4)

3.3 Classifiers

Follows the classifier results after training. Obviously we have a better discriminant on classifiers that result strong in Channels FC2,C4 on 22Hz during fisher score analysis. This is the proof that this frequency and channels is not the better choice for all sessions. (Figure 5)

3.4 Single Classifier

Just for check I tried to create a single classifier using all offline sessions data, the resulting output is very bad. The pilot change on each run and the classifier of first session is completely useless on last sessions. In fact the plot show a total overlap of the features without the possibility to find a good global discriminant (Figure 3)

3.5 Test Sessions

Plot of posterior probabilities over time, with 0.7 threshold, that help to understand evidence accumulation.

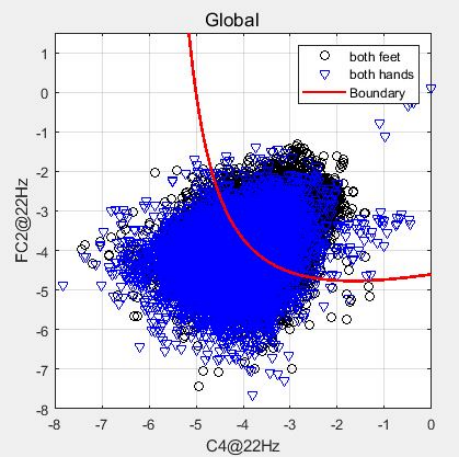
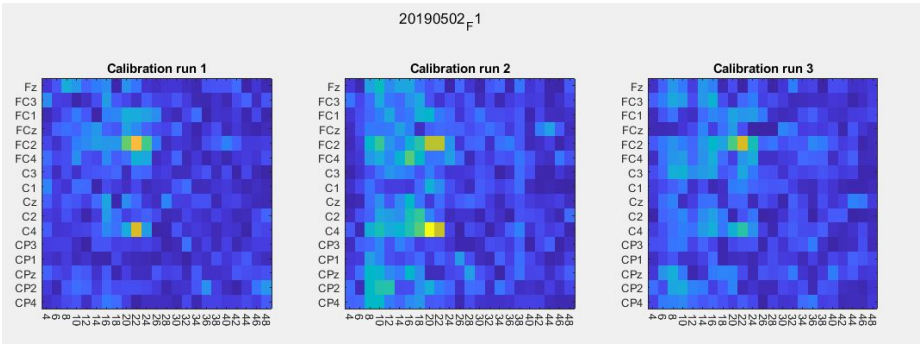
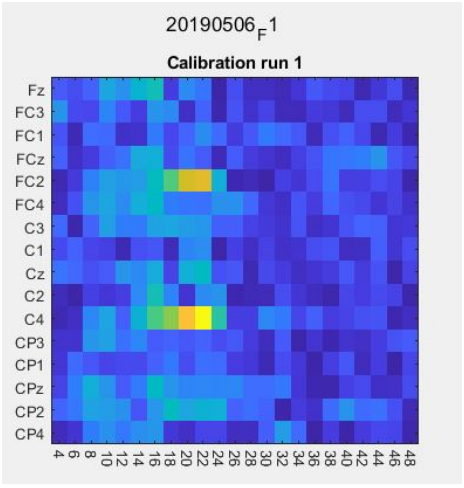


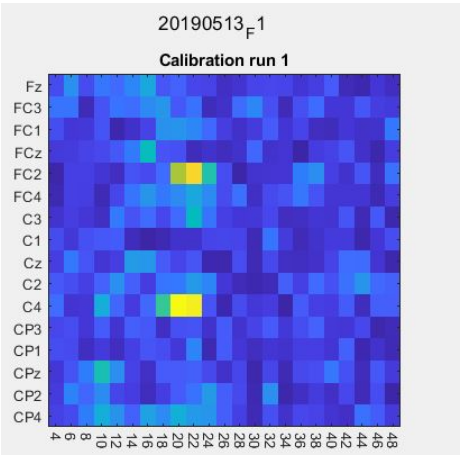
Figure 3: Single classifier.



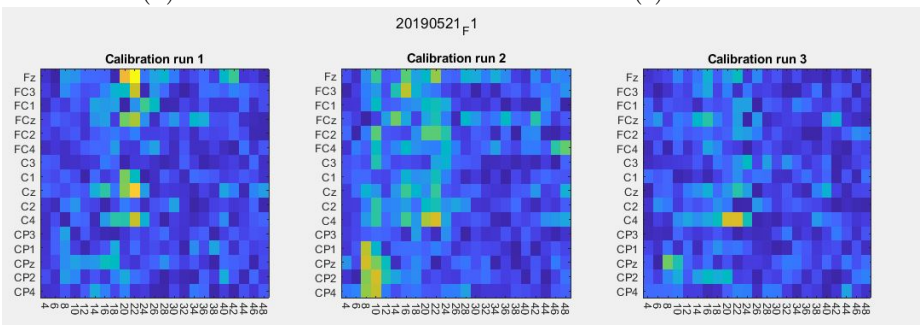
(a) 20190502



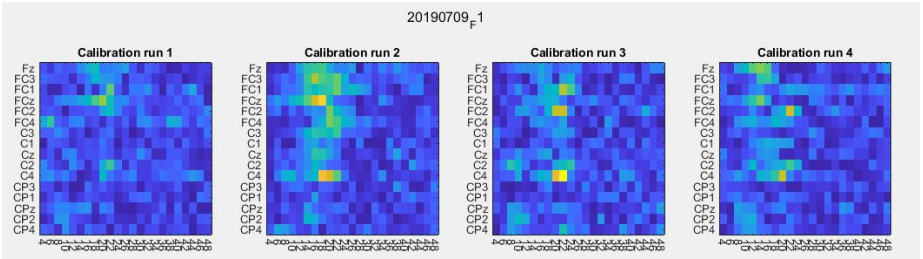
(b) 20190506



(c) 20190513

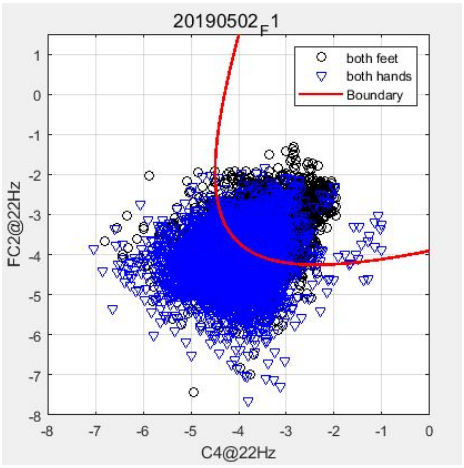


(d) 20190521

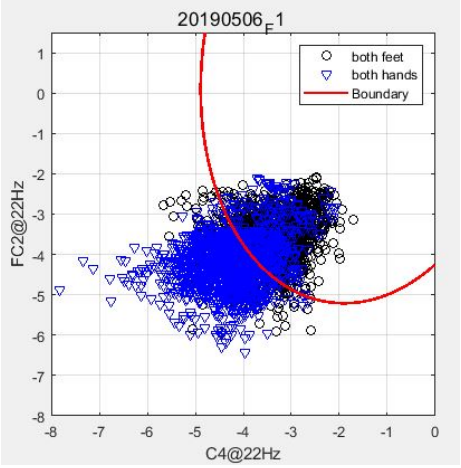


(e) 20190709

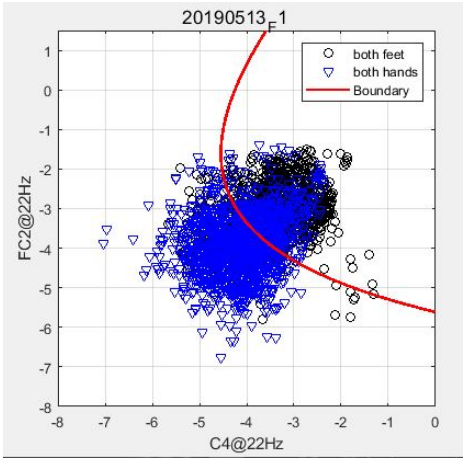
Figure 4: Fisher score results



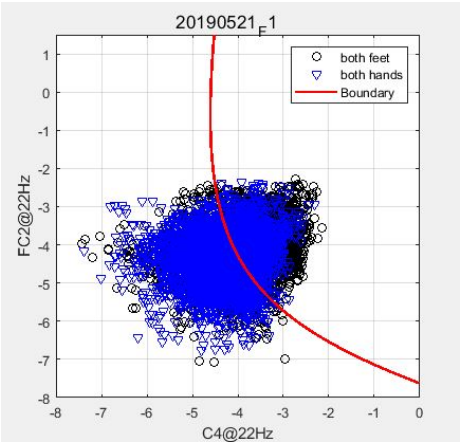
(a) 20190502



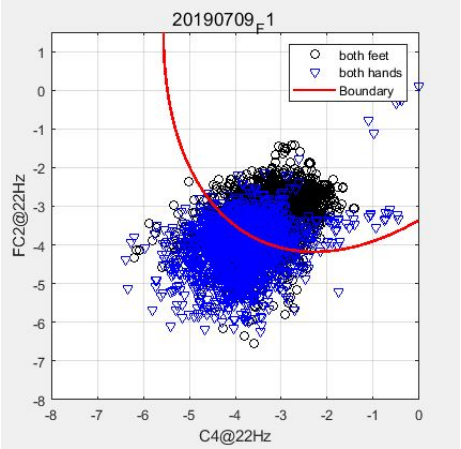
(b) 20190506



(c) 20190513



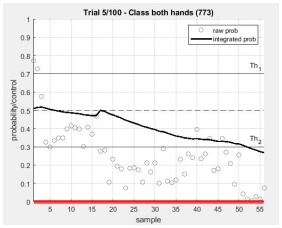
(d) 20190521



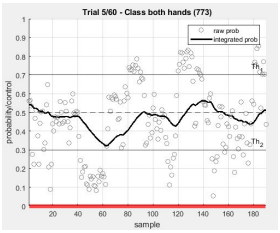
(e) 20190709

Figure 5: Models

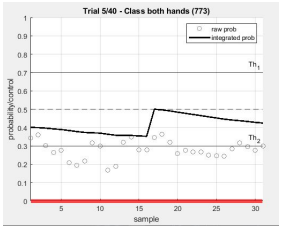
(a) 20190502



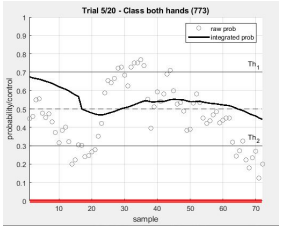
(b) 20190506



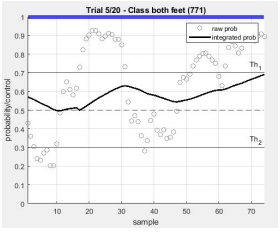
(c)20190513



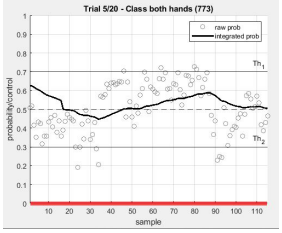
(d) 20190521



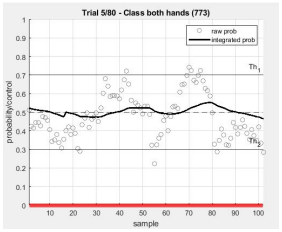
(e) 20190610



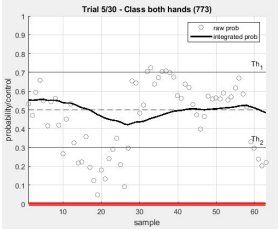
(f) 20190624



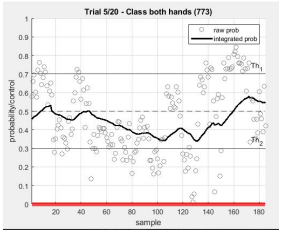
(g) 20190627



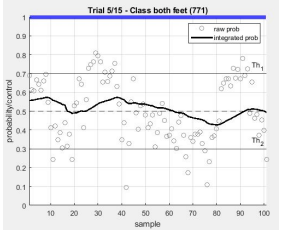
(h) 20190701



(i) 20190709



(g) 20190711



(h) 20190715

Figure 6: Models

4 Discussion

In this section we'll discuss the result obtained at the end of the pipeline, splitting the analysis in three steps, in the first step we'll discuss about results on feature selection task, on the second step the discussion will focus on classification task, and the conclusion step will discuss about the evolution of the results and performance over time.

4.1 Feature selection

Feature selection task over time highlight an alternations between dominant discriminant. The main dominant channels is FC2 and C4, but not always is the best choice for classifier. As example in session 20190502, the first run highlight FC2 and C4 on 22Hz, but at third run the discriminant on C4 is weak, and during the second run we can observe some movements in 20Hz band. This situation is also observable on 20190506 runs, where the beahaveior is similar at the second run of previous session. This uncertainty continues to recur in subsequent sessions, making it difficult to identify a unique discriminant valid for all sessions. Also in some sessions, such as the second run of the 20190709 session, where the fischer score highlights the presence of discriminants around the FCz channel on lower frequency bands between 12Hz and 18Hz. Certainly the univocal choice of channels and a common frequency band simplifies the operations but strongly penalizes some sessions, in which there is no strong discrimination between the selected channels and frequencies.

4.2 Classification

As explained in the previous step, the choice of a discriminant common to all the sessions penalized all those that after each run presented discriminants distant from the common ones. Or all the sessions in which between the various runs presented a large variance in the fisher score. All the graphs representing the classifiers are able to give a minimum separation. But unfortunately without guaranteeing the efficient separation that we would have obtained using an ad hoc feature selection tailored to each single session. Absolutely useless to try with a single classifier, it would certainly enrich the large dataset, but without taking into account the evolution over time. As a consequence, the results obtained in the testing phase fail to give strong evidence, despite the fact that the threshold set at 0.7 is not stringent as a parameter.

4.3 Evolution over time

The strength of a static learning system is the invariance of the subject over time. In this case the learning adapts to the subject, and at the same time during the online sessions it is the same subject to adapt to the learning model. This makes the learning process more complex across sessions, the data used for the previous models become of little use to the development of the current one. The evolution and the great variance between the various offline sessions after the sessions with feedback, demonstrates how the pilot, helped from feedbacks, adapts to the model. The real difficulty and the big difference with reinforcement learning is that with each change of the pilot, the previously acquired data

cannot always be used for the train of the new model. Especially during the month of June where there were few offline sessions, we were able to observe the strong changes during the feature selection, that required a recalibration. And above all highlighted the need to carry out both types of sessions continuously, in order to prevent similar gap between one session and another. Avoiding that the patient with online-only sessions, adapting to the model introduces new features that cause loss of precision in the current model.