Report of Project 3

Neurorobotics and Neurorehabilitation – Group 4

1. Introduction

1.1. Introduction to BMIs

Recent years have seen a growing interest in the neurorobotics field, which aims at studying brain-inspired approaches in robotics and at developing innovative human–machine interfaces. In this scenario, brain–machine interfaces (BMIs) represent a promising technology to directly decode user's intentions from neurophysiological signals and translate them into actions for external devices [1].

BMIs find usage in different kinds of applications: communication and control of virtual devices, motor substitution and control of virtual devices, motor rehabilitation to induce plasticity to specific tasks (e.g prosthetics), entertainment for healthy subjects; plus, these technologies are no longer used only under BMI experts control, but also in patients and end-user homes or clinics.

Great efforts in research have been made in the development of BMI solutions that can be applied in medical and therapeutic fields for subjects suffering severe diseases or which present motor disabilities like stroke, traumatic brain injury, spinal cord injury, amputation, cerebral palsy, etc.

1.2. Related work

In recent years, non-invasive BMI approaches have been explored to enable direct brain control of external devices. One of the most explored approaches is based on the electroencephalography (EEG), to detect the neural processes correlates to the motor imagery (MI).

In rest conditions the neurons in the sensorimotor cortex oscillate at a natural frequency in the α-band, that in this area takes the name of μ-band ; when a subject performs a movement (and on the onset of the movement) we can register a decrement in the amplitude of the EEG in the α $(8 - 13\ Hz)$ and in the $\beta(13 - 30\ Hz)$ bands because groups of neurons are recruited to perform the movement and start working in an asynchronous way. This phenomenon is called Event Related Desynchronization (ERD).

After the movement, in a rest condition, the neurons restart to oscillate synchronously. This phenomenon is called Event Related Synchronization (ERS). ERD/ERS characterize the neuronal activity over the sensorimotor cortex during a movement, i.e., during a motor execution (ME) task, but also during the imagination of the movement, like a motor imagination (MI) task. In this case we can detect an activation of much more other brain areas because of the missing sensory feedback of the body.

MI BMIs exploit the ERD/ERS phenomenon for control purposes, as they detect and classify the endogenous modulation of sensorimotor rhythms while the user is imagining the movement of a specific part of the body. In doing that, the systems continuously decode such brain patterns associated to the MI tasks by means of machine learning algorithms. The responses of the BMI decoder (a probability distribution of possible commands) are integrated over time and a command is delivered to the robot only when a given threshold is reached—i.e., when the control framework is confident about user's intention [1]. This is a discrete control modality both in terms of time and nature of the commands with a low information transfer rate.

1.3. State of the art

Several studies have shown the effectiveness of discrete control strategy in driving MI BMI-based devices. However, most modern studies investigated new approaches to use the BMI output as a continuous control signal for robotic devices [1]. In [1] authors proposed a control framework for MI BMI that allows a continuous control modality of a telepresence mobile robot in a navigation task thanks to a dynamic control, using a BMI decoder. As stated by authors, it works well experimentally, but not in real case scenarios when the user wants to continuously drive the device. It supports the Intentional Non-Control (INC), that allows the subject deciding to not send any command, and the correction of erratic behaviour of the decoder output. These features are useful in real world applications.

Despite the efforts made so far, these devices still have many limitations, thus there are many other challenges ahead: robustness, real-world and daily usage by end-users and translation impact.

Many other approaches have been proposed recently, like the usage of unbiasing and self-adaption approaches in BMIs to solve the uncalibration problem in the use of the decoder by the same user on different days, that can lead to loss of the performance of the decoder; this strong fluctuations over the training sessions may occur because of slightly different montage of the electrodes or loss of concentration or fatigue on different days of use. Another approach consists in the build of a BMI with zero-training, i.e., the creation of a universal decoder that can work for many new subjects.

To conclude we report a most modern approach proposed by [5] in which is used a holistic research approach which, instead of considering each entity separately, tries to study the interaction between machine, subject and application level. Mutual learning, that coordinates the capacities of both learning agents (brain and machine) is considered a critical factor for MI BCI success, and requires the user to learn how to generate distinct brain patterns for specific tasks.

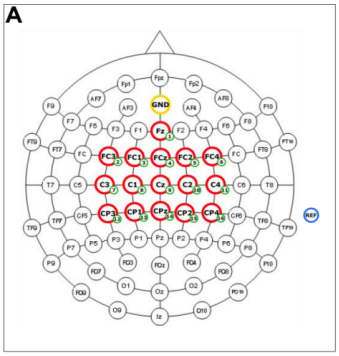
1.4. Contribution and Overview

In this work we lead a simple analysis on data collected during a 3-days experiment with 8 healthy subjects and we describe the implementation of a simple discrete control signal decoder, that implements the traditional control framework strategy with low-pass smoothing filtering. For this purpose, we have implemented 2 different MATLAB projects.

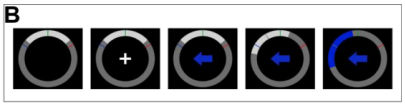
In the following section we present how the experiment was performed and how data was collected. In chapter 2 we describe the methods and the implementation of the MATLAB projects. In chapter 3 we report the data collected by the 3 analyses and we discuss the results and our conclusions.

Presentation of the problem

A



B



C

Code	Description	Decimal value
0x0001	Trial start	1
0x0312	Fixation cross	786
0x0305	Both Hand	773
0x0303	Both Feet	771
0x030F	Rest	783
0x030D	Continuous feedback	781
0x0381	Target hit	897
0x0382	Target miss	898
0x8000	Event OFF	/

Figure 1 - A) 16-channel layout. B) Example of visual paradigm used in the offline and online runs. C) Event codes in the GDF

Participants description:
In this work we analyze the data collected during a 3-day experiment with 8 healthy subjects and we simulate the whole BCI loop. Each subject participated in at least 2 recording days: in the first day, subjects performed 3 "offline" (calibration, no real feedback) and 2 "online" (with real feedback) runs, in the second day and third day, they performed 2 "online" runs each.

MI tasks description:
Participants were asked to perform 2 motor imagery (MI) tasks: both hands and both feet; plus, were asked to rest. The visual paradigm exploited during the training is reported in Figure 1 – A.

During the calibration runs, the feedback associated to the cue was automatically moving towards the correct direction. During the online runs, it moved accordingly to the output of the classifier.

Data acquisition description:
Data has been recorded with 16-channel EEG amplifier (g.USBamp, g.Tec) at $512\ Hz$. Electrodes were placed over the sensorimotor cortex accordingly to the 10-20 international layout (see Figure 1 – B).

Dataset description:
A dataset for each subject was collected and was provided to us:

- Each subject's dataset contains multiple files in the commonly used GDF format.
- Each file represents an offline run or an online run of the subject, registered in one of the 3 days, and contains data relative to all the tasks performed by the subject during that run.
- For each task is recorded one EEG signal for each of the 16 + 1 (reference) electrodes placed over the sensorimotor cortex of the subject (stored in $data$ in the GDF file).
- Each task is subdivided into more events that mark specific states during the task.
- Each event stored in the GDF file has an associated CODE, which represents the kind of event performed (Figure 1 – C), a duration time DUR and a certain position POS in the GDF file. Events, CODE's, DUR's and POS's are stored in the $header$ of the GDF file.

2. Methods

2.1. Project overview

Folder and project structure:
The whole project is contained in the folder **Project_NN_group4** and is organized as follows:

- **"Main" scripts:** $Project3\_part1.m$, $Project3\_part2.m$.
  They are the core of the project that implement the 3 points described in section 1.4.
- **data:** this folder contains the following subfolders:
  - **miscontinuous:** contains the datasets of the 8 subjects (downloadable from [6]), where each dataset contains all the GDF offline files and all the GDF online files.
  - ***outputs*:** contains the following subfolders:
    - **xproc:** (for *Project3_part1.m*) contains the files for the analysis of subjects.
    - **offline:** (for *Project3_part2.m*) contains a folder for each subject, where each folder contains the processed offline files in *.mat* format.
    - **online:** (for *Project3_part2.m*) contains a folder for each subject, where each folder contains the processed online files in *.mat* format.

        - **classifiers:** (for *Project3_part2.m*) contains a folder for each subject, that contains the classifier created for the specific subject in *.mat* format.
- **functions:** this folder contains the following subfolders and scripts:
    - **"Relevant" scripts:** *script1_Processing.m*, *script2_Features_selection.m*, *script3_Classification_training.m*, *script4_Classification_evaluation.m*, *script5_Evidence_accumulation_framework.m*.
    - **helper:** contains the helper functions provided for this project.
    - **util:** contains the util functions that have been implemented for this project.
- **toolbox:** it contains the $biosig\ toolbox$ [7] that allows the using of the $sload.m$ function, which is used to load and to read the . gdf files.

2.2. Grand average analyses on the whole population and on representative subjects

Data processing
The aim of this first section is to load the raw EEG data and to process all the files to extract features that we need for the next steps, both for the whole population and for the single subjects.

Grand average analyses con the whole population

· **Data loading**: the files are stored in the GDF data format, so loading the files we will get a 2D matrix with EEG data [*samples x channels*] but also a header that contains events information.

· **Spatial filtering**: ERD/ERS have not only spectral specificity (μ-rhythm and β-band), but also high spatial specificity (in the primary motor cortex, somatosensory cortex and in parietal cortex). For this reason, for each file, a Laplacian Filter is applied to avoid spatial blur and to highlight the spatial location of the sensorimotor rhythm. This is an approximate second derivative and uses the mean activity of neighbouring electrodes to compute the new signal; it is data and time independent, so this is essentially a linear transform of the signal.

· **Power Spectral Density**: this technique computes the power of the EEG signals at different frequencies. MI tasks are spontaneous activities (i.e., self-placed activities initiated by the subject), then they haven't a common stimulus onset but may have a different response across different trials (i.e., the signal is not phased-locked) and the signal may vary due to attention or fatigue of the subject; so, it is not possible to work in the time domain because the signal is too variable. It consists in taking a moving window of the signal and splitting it into overlapping segments; for each segment, the Fast Fourier Transform is computed and finally, averaging all the values, we obtain the PSD. In this case we use a function that exploits old FFTs to speed up the process.

Representative subject analysis

As we have seen, a clear ERD/ERS may appear only after averaging subjects, because subjects might learn to modulate only in few sub-bands (not in the whole μ -band), that are strictly subject specific. For this reason, we must create a different classifier for each subject and for each of them we must extract only those different discriminant sub-bands/features. To do so, it is essential to have an enhanced

spectral resolution.To visualize ERD/ERS, it is necessary to extract the baseline from the fixation phase (R), that is considered as the baseline period. The activity period (A) starts from the CUE appearance and ends at the end of the continuous feedback phase. The selection of the power of these periods required the extraction of the events position, duration and type exploiting information in the header.

$$ERD/ERS[dB] = log(A/R)$$

2.3. Analyses on BMI decoding on each subject

Description

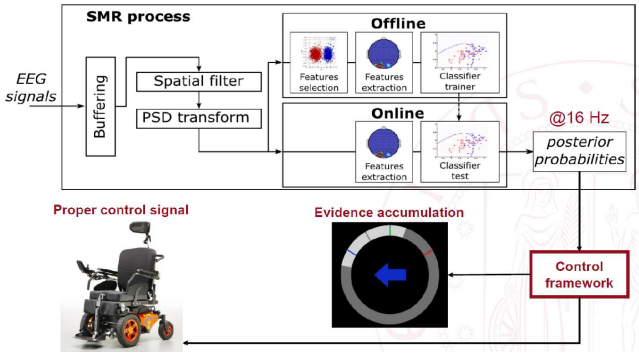


Figure 2 – General MI BMI processing workflow

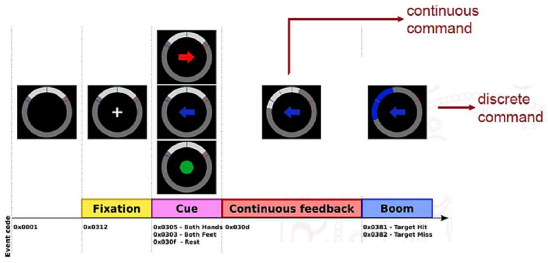


Figure 3 - Visual paradigm for a single trial

Reference scheme:

The general scheme that we have followed to implement the MI BMI decoder is reported in Figure 2. In particular, for the decoder, we have implemented a traditional strategy control framework with low-pass smoothing filtering (or, exponential smoothing integration).

Besides decoding the intentions of the subject in each trial performed, the discrete control signal generated by the decoder is used to control visual paradigm (Figure 3).

As shown in Figure 3, each trial contains the following main events:

- **Trial start:** in this period the trial starts.
- **Fixation:** the subject can start moving the control bar, when the bar is centred/fixed it can start the cue period.
- **Cue:** the MI task to perform ($both\ feet$, $both\ hands$ and $rest$) is shown to the subject.
- **Continuous feedback:** in this period the subject tries to move the control bar to achieve the given task, exploiting the visual feedback.
- **Boom:** the control bar reaches one of the two sides, the control signal is sent and the trial ends.

Notes:

- We create a different decoder/classifier for each subject analysed because, as said before, each subject learns to modulate only subject specific sub-bands of the μ-band. To do so, we selected the features to extract for the classification, the thresholds and the smoothing parameter for the evidence accumulation framework independently for each subject.
- Our decoder interprets only tasks $both\ feet$ and $both\ hands$, and it doesn't support NCI.

Main script

The $Project3\_part1.m$ is the main script and implements the two main phases calibration and evaluation of the decoder, that must be executed for each subject separately.

Calibration phase:

Here we consider only the offline runs. The aim of this phase is that of processing the raw EEG data, select and extract the most discriminative features, in order to represent each window of EEG data as a feature vector in a way that that window, belonging to a specific task, can be discriminated from another window belonging to a different kind of task. The set of (labeled) feature vectors obtained forms the training set that we use to create and train the classifier used in the decoder for the specific subject.

The main part of the program that execute the calibration of the decoder of each subject is reported in Figure 4, where the function/script used are $script1\_Processing.m$, $script2\_Feature\_selection.m$, $script3\_Classification\_training.m$, $script4\_Classification\_evaluation$.m (described below).

```
% Create and calibrate a classifier for each subject
for i = 1 : num_subjects
    disp(strcat("SUBJECT NUMBER: ", int2str(i)));

    try
        % Extracts the manually selected data for the i-th subject
        subject = subjects_list(i);
        selected_features = selected_features_list{i};

        disp(" [SCRIPT 1]: Processing");
        script1_Processing(offline_file_table, output_offline_path, psd_params_struct, subject, i);
        fprintf('\n');
    catch
        disp(strcat("SUBJECT ", int2str(i), " DOES NOT EXIST"));
        fprintf('\n');
        continue;
    end

    disp(" [SCRIPT 2]: Feature selection");
    [~, instances, labels] = script2_Features_selection(output_offline_path, selected_features, i);
    fprintf('\n');

    disp(" [SCRIPT 3]: Classifier training");
    script3_Classification_training(instances, labels, output_classifiers_path, i);
    fprintf('\n');

    disp(" [SCRIPT 4]: Classifier evaluation on training set");
    script4_Classification_evaluation(instances, labels, output_classifiers_path, selected_features, "offline", i);
    fprintf('\n');

    fprintf('\n');
end
```

Figure 4 - Calibration cycle on offline data (for each subject/decoder)

Evaluation phase:

Here we consider only the online runs. The aim of this phase is to extract the same discriminative features selected in the previous phase for the same subject, but this time to form a test set to evaluate the performance of the classifier created in the calibration phase. Then, we apply the evidence accumulation framework in output to the classifier, for smoothing the output and improve in this way the usability and the performance of the decoder.

The main part of the program that executes the evaluation of the decoder of each subject is reported in Figure 5, where the script used are $script1\_Processing.m$, $script2\_Feature\_selection.m$, $script4\_Classification\_evaluation$.m, $script6\_Evidence\_accumulation\_framework.m$.

```
% Evaluate the classifier for each subject
for i = 1 : num_subjects
    disp(strcat("SUBJECT NUMBER: ", int2str(i)));

    try
        % Extracts the manually selected data for the i-th subject
        subject = subjects_list(i);
        selected_features = selected_features_list{i};
        selected_thresholds = selected_thresholds_list{i};
        selected_alpha = selected_alpha_list{i};

        disp(" [SCRIPT 1]: Processing");
        script1_Processing(online_file_table, output_online_path, psd_params_struct, subject, i);
        fprintf('\n');
    catch
        disp(strcat("SUBJECT ", int2str(i), " DOES NOT EXIST"));
        fprintf('\n');
        continue;
    end

    disp(" [SCRIPT 2]: Feature selection");
    [trials, instances, labels] = script2_Features_selection(output_online_path, selected_features, i);
    fprintf('\n');

    disp("[SCRIPT 4]: Classifier test");
    [post_probabilities] = script4_Classification_evaluation(instances, labels, output_classifiers_path, selected_features, "online", i);
    fprintf('\n');

    disp("[SCRIPT 5]: Accumulation framework");
    [decisions, avg_time_to_deliver_command] = script5_Evidence_accumulation_framework( ...
        post_probabilities, labels, trials, selected_alpha, selected_thresholds, pp_frequency, i);
    fprintf('\n');

    fprintf('\n');
end
```

Figure 5 - Evaluation cycle on online data (for each subject/decoder)

Script 1: Processing

The $script1\_Processing.m$ executes the following processing for each file of the selected subject.

Load the EEG signal:

We used the $sload.m$ helper function to load the GDF offline/online file of the subject, then we extracted the matrix of the raw EEG signal matrix $s = [samples \times channels]$ and its relative header h.

Spatial filter:

The solution we adopted is the Laplacian filter. Each filter component is defined as follow:

$$e_i^{LAP} = e_i - \sum_{j \in S_i} h_{ij} * e_j, \qquad h_{ij} = \frac{\frac{1}{d_{ij}}}{\sum_{j \in S_i} \frac{1}{d_{ij}}}$$

where e_i is the raw potential of the electrode i-th, S_j is the neighbour of the electrode i-th and d_{ij} is the distance between electrodes (i, j). So, we obtained the filtered EEG signal $s_{lap} = [samples \times channels]$.

PSD transform:

We moved into the frequency domain computing the PSD using the helper function $proc\_spectrogram.m$ provided (Figure 6), which implements the enhanced Welch's method.

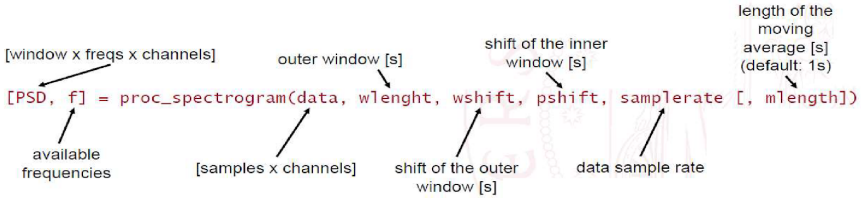


Figure 6 - Helper function which implements the Welch's method

Therefore, what we have obtained is the matrix $PSD = [windows \times frequencies \times channels]$ containing the power spectral density for each window of signal over each channel (where the frequencies extracted are in the selected frequency band$[4: 2: 48]\ Hz$: PSD_filt).

Save PSD data in .mat file:
We have stored the processed PSD_filt data, the selected frequencies for the PSD, the events converted into frequency domain for the PSD and the parameters used for the PSD in a $.mat$ file.

Script 2: Feature selection

The $script2\_Features\_selection.m$ executes the following processing for the selected subject.

Load and concatenate the PSD files:
In this script we have loaded the $.mat$ processed files of the subject created in the previous script.
We extracted and concatenated the data in the variable $PSD = [windows \times frequencies \times channels]$, we loaded the parameters and we loaded and concatenated the events for the PSD in psd_events.

Create the label vectors:
We used the util function $labeling\_data\_1run.m$, that we implemented, to create and concatenate the label vectors, in particular, we created the label vector $trials$, which is a vector containing blocks of NUM_TRIAL's for the periods ranging from start fixation to end feedback and all 0's outside these periods, and the label vector $interesting\_periods$, which is a vectors containing blocks of $CODE$'s (i.e., the TYP of the cue) for the periods from start cue to end feedback and all 0's outside these periods.

Compute and visualize Fisher score:
At this point we computed the power of the PSD: $PSD\_power = \log log\ (PSD)\ [dB]$ and we used it to compute the Fisher's scores matrix (w.r.t. the runs): $FisherScore = [channels \times frequencies \times run]$, using the formula in Figure 7. The Fisher score is the distance between the distributions of the two classes of instances, and we have used it to highlight the most discriminative features.

$$FS(k) = \frac{abs(\mu_{C_1}(k) - \mu_{C_2}(k))}{\sqrt{\sigma^2_{C_1}(k) + \sigma^2_{C_2}(k)}}$$

μ_{C_1}, μ_{C_2} Mean for classes C_1, C_2

$\sigma^2_{C_1}, \sigma^2_{C_2}$ STD for classes C_1, C_2

Figure 7 - Fischer's score formula

Note: we have computed log(PSD) to obtain a normal distribution of data during the classification.

Feature selection and extraction:
Using the visualization of the Fisher Scores over the different runs, we have manually selected (on offline data) the 3 most discriminative features for the specific subject (w.r.t. the trial/task, where $feature\_selected = (frequency,\ channel)$), taking into account their neurophysiological meaning (i.e., spatial and spectral location) and their stability over time. Then we used these features to extract the 3 values of the PSD_power that compose our feature vectors for each window and we obtained the matrix $feature\_vectors = [windows \times 3\_PSD\_powers]$, that we have used to train/test our classifier.

Labels extraction:
To evaluate the performance of our classifier, we extracted the ground truth labels of the instances: $labels = [windows \times 1\_label]$, where the two classes of labels are $Both\_Feet$ and $Both\_hands$.
Note: actually, the real implementation of this point is a bit more complicated, because we have considered $labels = interesting\_periods$ to simplify the code in the next scripts.

Script 3: Classification training

The $script3\_Classification\_training.m$ executes the following processing for the selected subject (this script is used only in the calibration phase).

Selection of the appropriate classifier:
To select the most appropriate classifier, we computed the covariance matrix (that expresses the shape of a given distribution) of the two distributions of instances for the two classes of labels.
The rule used for selecting the appropriate classifier is the following: if the two covariance matrices/distributions of the two classes of labels are equal we have used a linear classifier (LDA), for the other cases the linear classifier is not suitable, so we used a quadratic classifier (QDA).

Classifier training:
To train the selected classifier on the training data obtained, we used the function $fitcdiscr.m$ provided by MATLAB. In the end, we have stored the classifier in a $.mat$ file.

Script 4: Classification evaluation

The $script4\_Classification\_evaluation.m$ executes this processing for the selected subject:

Prediction:
To get the predictions of the selected classifier over the training/test set (for each window) we used the function $predict.m$ provided by MATLAB; in this way we have obtained the $predictions$ vector, that contains the predicted label class for each window and the vector $post\_probabilities$, which contains the posterior probability for the two classes of labels for each window.
Note: LDA and QDA are unsupervised models, the ground truth labels are used only to evaluate the performance, but they are not used for the classification.

Visualization of the data distribution and of the instances:
We plotted the instances of the two classes of labels w.r.t the first two features selected and also the classifier, in order to have an overview of the goodness of the prediction (see Table 1).

Evaluation and visualization of overall and single class accuracies:
To evaluate the performance of the classifier over the training/test set we have computed the single sample accuracy (% of windows correctly classified on average) over each of the two classes:

$$ACC = 100 * \frac{TP+TN}{TP+TN+FN+FP} = 100 * \frac{|windows\ correctly\ predicted|}{|windows|} =$$

$$= 100 * \frac{1}{|windows|} \sum_{i=1}^{|windows|} (ground\ truth\ label(i) == predicted\ label(i))$$

With the same formula we have also computed the overall accuracy, considering this time the predictions over the two classes together. Then, we have plotted the three values (see Table 1).

Script 5: Evidence accumulation framework

The $script5\_Evidence\_accumulation\_framework.m$ executes the following processing for the selected subject (this script is used only in the online part).

Compute the evidence accumulation framework:
At this point we obtained as output of the classifier the $post\_probabilites$. The raw probabilities are quite erratic, so to smooth the control framework we used an evidence accumulation framework; in particular, we used the strategy of the exponential smoothing integration:

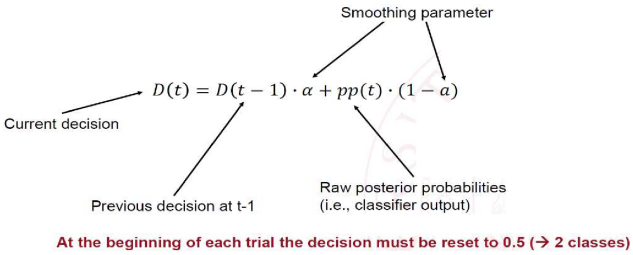


Figure 8 - Exponential smoothing integration formula

We computed the decision (i.e., the control signals) of the decoder for each trial using the formula above and we created a vector $decisions = [trials \times 1\_decision]$. The decision is taken by the decoder as soon as the decision function D cross one of the two confidence thresholds that are subject specific.

Notes: the command of the decoder can be delivered in two modalities:

- **Without rejection:** both when the decoder manage to take a decision (i.e. the decision signal cross one of the thresholds) and when the trial ends and the decision isn't made (i.e. the decision signal can't reach one of the thresholds), in this case the decoder delivers a non-valid decision.
- **With rejection:** only when the decoder manages to take a decision.

Visualize the output of the decoder:

To do so, for a sample trial, we plotted: the posterior probabilities (in each window of the trial), the thresholds lines and the function of the decision over the trial, marking with a red circle the point in which the decision (the control signal generated by the decoder) is made (see Table 1).

Evaluation of the trial accuracy without and with rejection:

To evaluate the performance of the decoder we computed the trial accuracy (% of trials correctly classified on averages) without and with rejection, similarly as the simple sample accuracy (see Table 1):

$$ACC = 100 * \frac{TP+TN}{TP+TN+FN+FP} = 100 * \frac{|trials\ correctly\ classified|}{|windows|} =$$

$$= 100 * \frac{1}{|trials|} \sum_{i=1}^{|trials|} (ground\ truth\ label(i) == decision(i))$$

Note: accuracy depends also on confidence thresholds and smoothing parameter values we selected.

Average time to deliver a command:

Finally, we computed the average time to deliver a command as the ratio between the average number of posterior probabilities needed to the decoder to take a decision (over different trials) and the frequency of the stream of posterior probabilities in output to the classifier ($16\ Hz$):

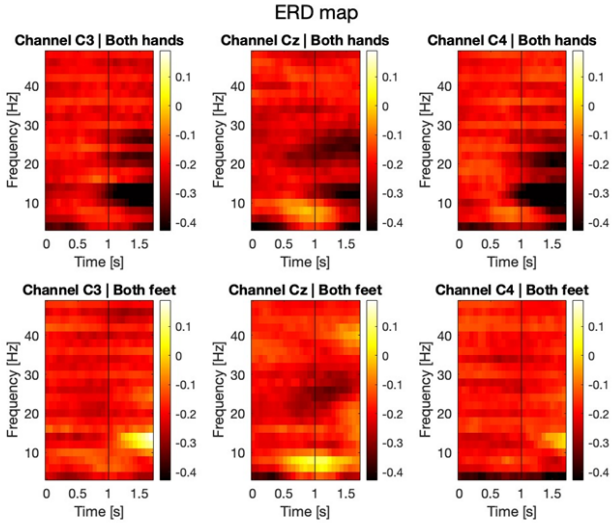
$$avg\_cmd\_time = \frac{mean(\#pp\ to\ take\ a\ decision)}{pp\_frequency}$$

3. Results and discussions

Results

Grand average analyses on the whole population and on representative subjects:

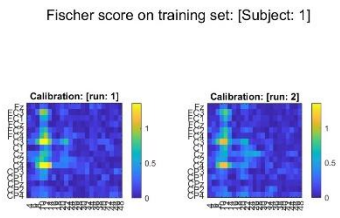
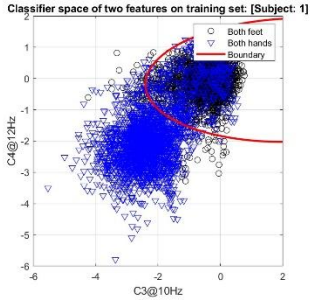
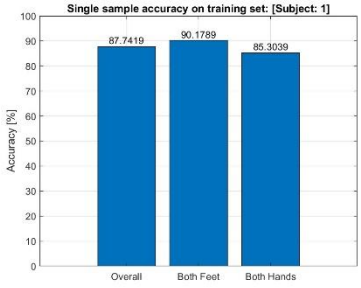
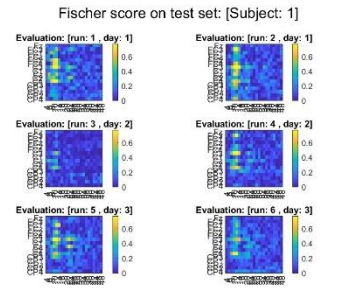
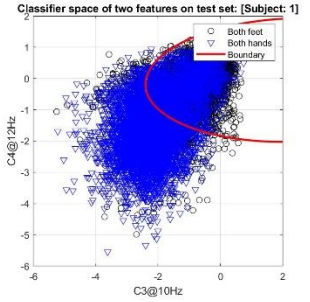
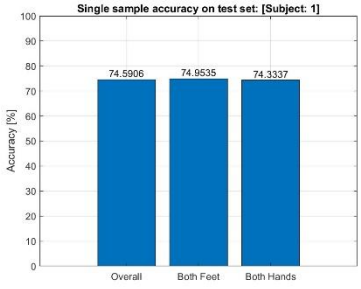
In Figure 9 is reported a visualization of averaged ERD/ERS for both tasks in the selected channels. We can see that for the task *Both hands* is negative, specially in channel C3 and C4, that correspond to lateralized regions of the sensory motor cortex. For the task *Both feet* we can find some positive signal specially in channel Cz, meaning an activity in the medial sensory motor cortex.



Figure(9) ERD maps for tasks 'Both feet' and 'Both hands'

Analyses on BMI decoding on each subject:

In Table 1 are reported the output of the main script $Project3\_part2.m$ for the subject 1. In Table 2, Table 3 and Table 4 are reported the data collected for the same script.

Sample subject 1		
Calibration phase		
		
Evaluation phase		
		
Evidence Accumulation Framework		

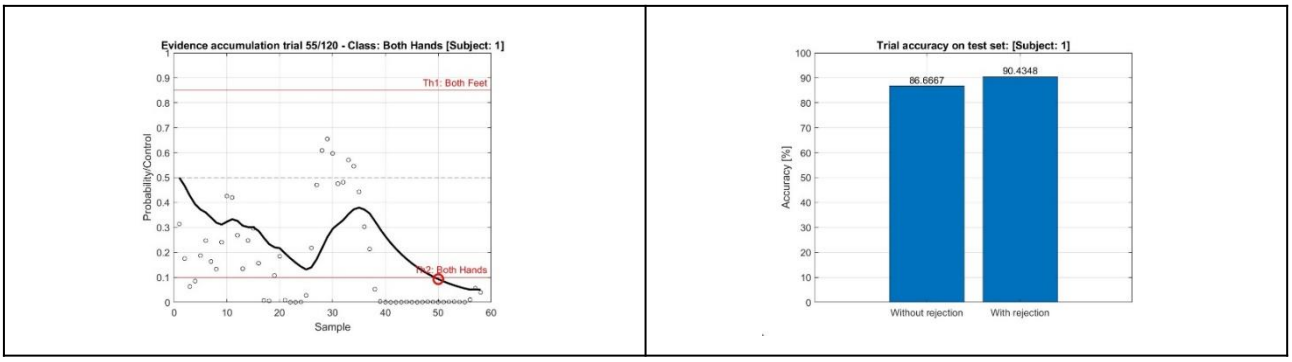


Table 1 – Output Project3_part2.m for the sample subject 1

Single sample accuracy on training set [%]			
	Overall	**Both Feet**	**Both Hands**
Subject 1	87.74	90.17	85.30
Subject 2	82.22	83.79	80.64
Subject 3	74.58	63.26	85.89
Subject 4	72.00	65.19	78.81
Subject 5	81.23	79.97	82.49
Subject 6	68.20	71.04	65.3
Subject 7	81.22	77.47	84.98
Subject 8	78.70	72.76	85.78

Table 2 - Single sample accuracy on training set (for each decoder/subject)

Single sample accuracy on test set [%]			
	Overall	**Both Feet**	**Both Hands**
Subject 1	74.59	74.95	74.33
Subject 2	77.13	71.85	84.03
Subject 3	70.96	81.78	61.85
Subject 4	70.74	71.40	70.18
Subject 5	74.39	76.42	72.52
Subject 6	61.58	36.30	83.43
Subject 7	73.18	66.03	81.66
Subject 8	72.58	52.89	82.94

Table 3 - Single sample accuracy on test set (for each decoder/subject)

	Trial accuracy on test set [%]		**Average time to deliver a command [s]**
	With rejection	**Without rejection**	
Subject 1	86. 66	90. 43	2. 2375
Subject 2	91. 66	94. 82	2. 1443
Subject 3	73. 33	74. 57	1. 9755
Subject 4	78. 33	87. 85	3. 4979
Subject 5	79. 16	84. 07	1. 9979
Subject 6	53. 33	58. 71	2. 1385
Subject 7	80	82. 05	2. 2443
Subject 8	69. 16	77. 57	1. 9057

Table 4 - Trial accuracy on test set (for each decoder/subject)

Discussion of the results and final considerations

In this work we have seen how to implement a simple discrete control signal decoder.

In the first part of the work, we have analysed the raw EEG data in order to extract the features for the final purpose of creating a classifier for the two MI classes. This analysis consists in filtering and translating in a frequency based domain, but also in extracting and concatenating the events. The ERD/ERS maps in the most meaningful channels we found agree with the literature, showing lateralized deactivation for *Both hands* class, and medial activation for *Both feet* class.

In the second part of the work we have built a MI BMI decoder in order to transform a signal input into a decision command. We have considered the 3 most discriminative features, according to the Fisher Scores visualization, to select the feature for the classifier of each subject. Then, selecting the parameters for the evidence accumulation framework, we managed to get a good result in terms of the trial accuracy on the test set. In fact, as we can see from the result tables above, we can notice an improvement in the performance of the decoder, compared to the single sample accuracy result on the test set, and this is what we were looking for.

Eventually, also the time to deliver a command plays an important role as it is related to the velocity of decision: the command is delivered when the signal is above the threshold. Optionally, if needed, we can lower the threshold of the average time to deliver a command by modifying the smoothing parameter.

Again, for a better result, we should focus on the mutual learning approach: instead of considering each entity separately, we should try to include the interaction between those entities, the user, the decoder, and the robot. In other words, it is important not only the optimization of the decoder, but also the training of the subject.

So, we would like to conclude with a quote from Carmena: “The BMI is a skill to learn”.

4. Bibliography

[1] Tonin L et al. The role of the control framework for continuous tele-operation of a BMI driven mobile robot. IEEE Transactions on Robotics, 36(1):78-91, 2020. doi: 10.1109/TRO.2019.2943072.

[2] Pfurtscheller G et al. Motor imagery and direct brain-computer communication. Proceedings of the IEEE, 89(7):1123-34, 2001. doi: 10.1109/5.939829.

[3] Wolpaw JR et al. Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. Proc Natl Acad Sci USA, 101(51):17849-54, 2004. doi: 10.1073/pnas.0403504101.

[4] Leeb R et al. Transferring brain–computer interfaces beyond the laboratory: Successful application control for motor-disabled users. Artificial Intelligence in Medicine, 59(2):121-32, 2013. doi: 10.1016/j.artmed.2013.08.004.

[5] Perdikis S et al. The Cybathlon BCI race: Successful longitudinal mutual learning with two tetraplegic users. PLOS Biology 16(5):e2003787, 2018. doi: 10.1371/journal.pbio.2003787.

[6] Dataset [link].

[7] Biosig toolbox [link].