Capstone Project

Alfonso Sánchez De Lucio

December 25, 2017

DEFINITION

PROJECT OVERVIEW

Different discoveries about the brain have been made since started being considered as the main organ in the human body. One of the most used discoveries these days is the one made by Richard Caton [Finger, 2001], whom found that brain signals, generated from various stimulus, can be measured outside the head in the form of scalp potentials, known as Electroencephalograms or EEG signals. Nowadays, EEG readers headsets such as EMOTIV and NEUROSky are starting to be popular, and probably they would be a mainstream gadget not too far from now.

Brain Computer Interfaces (BCI) are control systems based on brain signals, which are used to find the patterns of specific thoughts in order to be used to control devices such as prostheses, pointers, exoskeletons, and many others. Unfortunately, EEG signals are normally affected by different artifacts that complicate the findings of relationships between them and specific actions. This reduction in the quality of the signals is normally improved by using signal processing techniques, however, because of the brain complexity, there will always be uncertainties about where the brain signals were generated and their relationships.

As a result, Machine learning techniques have been closely related with BCI systems, from supervised learning techniques to, more recently, deep learning techniques [Cecotti and Graser, 2011]. The importance of Machine learning techniques come from their ability to adapt to the data in order to find the best model that allows the recognition of specific actions from brain signals. Therefore, the combination of BCI systems, EEG readers headsets and Machine

Learning techniques would generate great benefits, especially for people affected by conditions and diseases that reduce their motor functionality [Birbaumer, 2006].

PROBLEM STATEMENT

There are many ways to find relationships between brain signals and specific actions, where, as mentioned before, machine learning techniques have been facing this problem. Some of the most implemented methods are Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Multilayer Perceptrons Neural Networks (MLP), Decision Trees, Hidden Markov Models (HMM) and Convolutional Neural Networks (CNN). Although, the performances of the methods depend on different factors, such as their application, the parameters used to represent the brain signals, the length of the signals and many others.

The problem to solve in this document is the classification of six actions of interest from EEG signals, presented in the *Kaggle* webpage as the *Grasp-and-Lift EEG Detection* competition [1]. In this case, it will be assume that the problem is related with an online EEG classifier, and therefore, it is required the use of a low computational cost classifier. The objective of the online classifier will be to identify the following actions from the EEG signals [2] .

1. HandStart
2. FirstDigitTouch
3. BothStartLoadPhase
4. LiftOff
5. Replace
6. BothReleased

For this, it would be required to train a classifier able to determine movements from EEG signals, considering their complexity. This, because in order to work with electrophysiological signals, signal processing techniques are going to be needed. For example, low pass filters to reduce the effects of external artifacts, or modeling of the signals to find useful characteristics. However, the use of these techniques also increment the computational cost, which is important as mentioned in the last paragraph. On the other hand, it is important to represent signals in such a way that allows the classifier to relate them with the events of interests. As result, the balance between robustness and performance needs be to be consider. Then, a SVM and and a MLP will be implemented as a solution model and as a benchmark, respectively, algorithms that will be discussed in more detail later in this document.

[1] https://www.kaggle.com/c/grasp-and-lift-eeg-detection
[2] https://www.kaggle.com/c/grasp-and-lift-eeg-detection/data

METRICS

To find the best solution for the proposed problem it would be considered the speed of the algorithm to find the solution, the amount of correct classified signals (accuracy) and the amount of correct classified movements from the total signals that were classified as movements (precision). Furthermore, it would be also considered the area under the Receiver Operating Characteristic (ROC) curve, to follow the *Grasp-and-Lift EEG Detection* competition rules. The last metric represents the area under the curve that is generated by plotting the fraction of signals that were corrected classified as movements out of the total classified signals (TPR = true positive rate) against the fraction of signals that were false classified as movements out the total signals that were not related with the desired movement (FPR = false positive rate), figure 1.

In this case, the specified problem in the last subsection will be divided into six binary classifiers, one for each movement of interest. Thus, the area under the Receiver Operating Characteristic curve makes sense since it is a graphic representation of the sensitivity against the specificity for a binary classifier. That is the reason because the ROC area will be used as metric to determine the performances of a SVM model and a MLP benchmark, as binary classifiers.

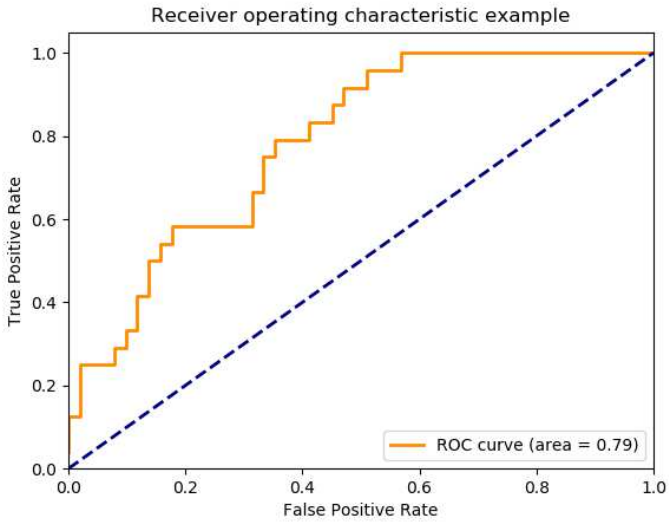

Figure 1: Example of a ROC curve and the area under the curve [3].

[3] http://scikit-learn.org/stable/modules/model_evaluation.htmlroc-metrics

ANALYSIS

DATA EXPLORATION

The data for this project was obtained from the *Grasp-and-Lift EEG Detection competition* [Luciw et al., 2014]. The data found in the *Kaggle* competition is composed by three sets, a set containing a submission example, a set of EEG signals provided to solve the competition problem and a set of EEG signals to prove the proposed solution for the mentioned competition. However, for the proposed capstone project, only the set of signals provided to solve the problem would be used, the *train.zip* file. The file contains information about 12 subjects, 8 series of trails for each of them and approximately 30 trials for each series. Additionally, it is important to mention that the data was acquired with a sampling rate of 500 Hz.

To be more specific, there are two files for each series of experiment inside the train.zip compressed folder, the *subject_data.csv* files containing the EEG data and the *subject_event.csv* files containing the labels for each of the events found in the EEG data. The events in the train.zip compressed folder are the actions presented in the PROBLEM STATEMENT section, and can be found on each of the columns of the event files. On the other hand, the files containing EEG data are composed by 32 columns, each of them representing one of the channels used to acquire the brain signals, as can be seen in figure 2, while the rows represent the series of experiments

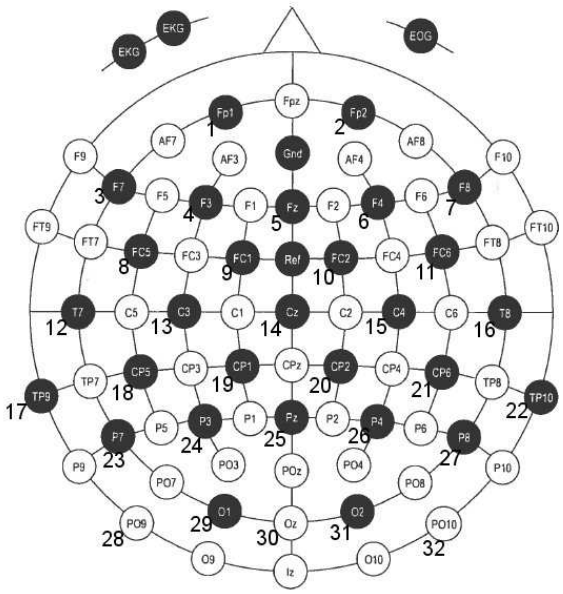

Figure 2: Locations for the electrode channels used to acquire the EEG data [4].

[4] https://www.kaggle.com/c/grasp-and-lift-eeg-detection/data

Thus, the data acquired from the competition will be used to obtain the signals EEG features to be related with their specific labels, events, using a supervised machine learning technique [Lotte et al., 2007]. However, there are many ways to define the features that allows the algorithm to learn about the signals. The most straight forward way is to use the signals as they are, considering the time domain amplitude as features, however, their spectrum can also be used, as well as their Autoregressive (AR) coefficients of the time signals or spectrum [Lotte et al., 2007].

EXPLORATORY VISUALIZATION

The *Grasp-and-Lift EEG Detection competition* mentions that the recognition of movements should be considered, for the training and testing processes, if movements occurs within ±150 ms, or ±75 frames, of the data being analyzed. Hence, the signals obtained from the *Kaggle* competition will be segmented into signals of 300 ms or 150 frames, as minimum, and, if any of their corresponding labels represent a specific movement, then their labels would represent that movement, as can be seen in tables 1 and 2. The same applies in case the signals are analyzed in the frequency domain or by using their AR coefficients as features, the spectrum and the AR parameters will be obtained from the segmented signals.

As mentioned in the last paragraph, the time domain analysis of the segmented EEG signals can be considered. For this, different methods can be discussed, one of the most commonly used is the Power Spectral Density (PSD) estimation, obtained by using the Fast Fourier Transform (FFT) or the Welch periodograms, although, the spectral AR estimation is also commonly used [Fabiani et al., 2004, Hosni et al., 2007]. In figure 3 is presented the mentioned pre-processing methods for the segmented EEG data.

Table 1: Subjects EEG data

id	Feature 1	...	Feature n
Frame 1	-31	...	704
⋮	⋮	⋮	⋮
Frame 150	122	...	402
⋮	⋮	⋮	⋮
Frame 300	167	...	566

Table 2: Subjects Labels events

id	Label 1
Frame 1	0
⋮	⋮
Frame 150	0
⋮	⋮
Frame 300	1

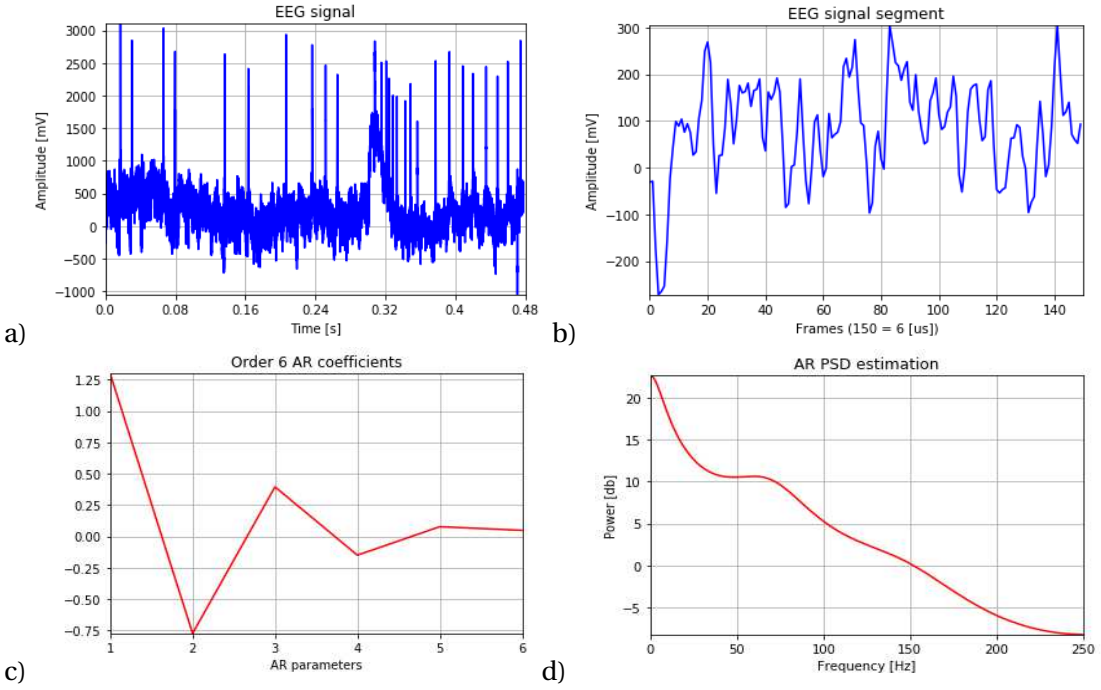

Figure 3: Commonly preprocessing of EEG signals for BCI applications: a) Raw EEG signal, b) Segment of the raw EEG signal, c) AR coefficients of the segmented EEG signal, and d) PSD obtained using the AR coefficients.

ALGORITHMS AND TECHNIQUES

In order to solve the previously mentioned problem, a Support Vector Machine (SVM) classifier it is going to be use, because are algorithms that relies on discriminants hyperplanes that maximize the margins. The maximization of the margins allows the algorithm to find the optimal hyperplane that divides the classes of interest, as can be seen in figure 4. Moreover, SVM classifiers have been used in online BCI systems before with considerable success [Lotte et al., 2007]. However, it is a complex algorithm, if its linear function can not solve the problem, a non-linear function is always required, which increment its complexity depending on the chosen function. This is know as the *kernel* trick.

The SVM model will be trained using a percentage of the obtained EEG data, while the rest of the data will be used to test the performance of the trained method. The training of the SVM algorithm is going to be implemented using a vector of features generated by modeling the segments of the signal as AR processes, in order to use their AR coefficients as features, [Hosni et al., 2007]. The AR parameters will be obtained using the Yule-Walker algorithm, for each of the segmented EEG signals, for all channels.

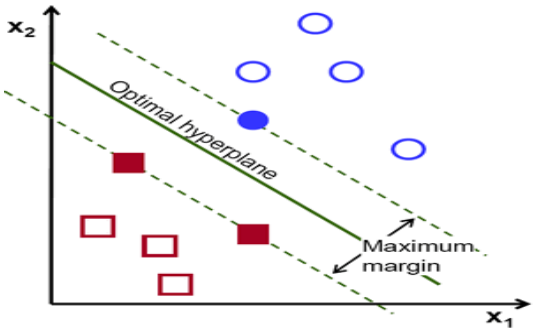

Figure 4: Maximization of the Hyperplane margins [5].

BENCHMARK

As a benchmark model, it is going be implemented a Multilayer Perceptron (MLP) Neural Network, since it is one of the most used methods in BCI applications [Lotte et al., 2007]. Neural Networks, and therefore, Multilayer Perceptrons, are widely used and, when composed by enough layers and neurons, they can approximate any continuous function. The input and output layers of the network would depend on the number of features used and the number of actions of interests to classify, respectively. Additionally, a hidden layer is normally used to complete the network architecture, composed with a number of perceptrons that varies from 10 to 100 [Huan and Palaniappan, 2004], as can be seen in figure 5.

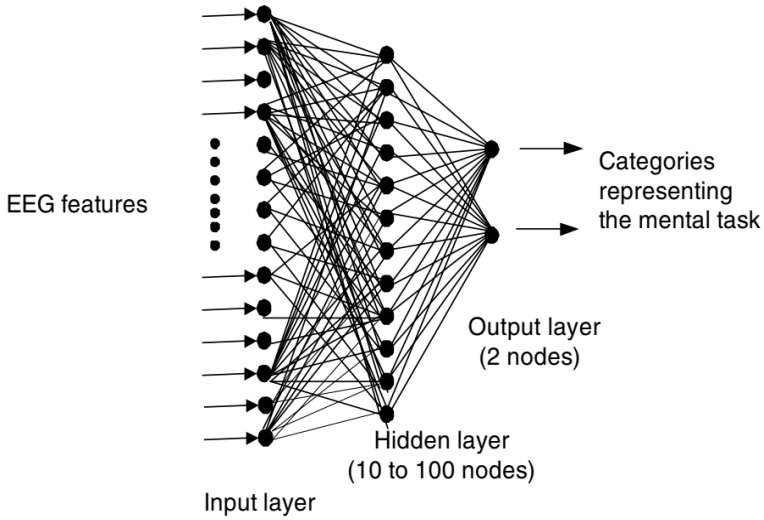

Figure 5: Architecture for a Multipayer Perceptron [Huan and Palaniappan, 2004].

[5] https://www.kaggle.com/c/grasp-and-lift-eeg-detection/data

In this case, the weights of the MLP will be trained with the same percentage of the data used to train the SVM model, while the rest of the dataset will be used to test the performance of the network during the classification of EEG signals as movements or actions of interests. As mentioned in the METRICS subsection, the performance will would be the probability of how many EEG signals in the testing dataset are classified correctly.

METHODOLOGY

DATA PREPROCESSING

In order to implement the mentioned Machine Learning algorithms, it would be necessary to pre-process the EEG signals. To be more specific, the segmented EEG signals, which will be modeled as Autoregressive processes to obtain their coefficients. As a result, the coefficients will be use to generate vectors of features. Furthermore, the features will allow the algorithms to relate them with the events or actions of interests. Unfortunately, PSD wont be used for the features, because of uncertainties about the data being use, about the remaining frequencies on the signals, even when their acquisition is well documented [Luciw et al., 2014].

For the SVM case, the idea would be to pre-process the signals following the procedure shown in figure 3, explained in the EXPLORATORY VISUALIZATION subsection. For this, an order 6 will be used for the AR modeling [Guger et al., 2001, Hosni et al., 2007, Huan and Palaniappan, 2004, Schlögl et al., 1997], by relying on the python *nitime.algorithms.autoregressive* library[6], on the *AR_est_YW* function. What does this function do, is to determine the AR model of a random processes using the Yule Walker equation, which is possible since the EEG signals can be considered as Wide Sense Stationary random processes [Zetterberg, 1969].

On the other hand, the MLP case will be implemented using the same features as in the SVM case. For both cases, they are going to use vectors of features with 192 elements (32 channels by 6 AR parameters), for each segment of the EEG signals. It is important to remark that, before being applied the mentioned techniques, the mean of the EEG was subtracted from it, by using the python *sklearn.preprocessing* library [7].

IMPLEMENTATION

The implementation of the BCI system was based on the *Grasp-and-Lift EEG Detection Kaggle* competition. The reason to chose this problem was because of my interest in the topic. Hence, before starting implementing the code to achieve what the competition requires, I decided to read about BCI systems. After that, I started looking for solutions presented in the competition [8], in order to find cleaver solutions about how to accomplish the requirements of the presented problem. Unfortunately, the solutions found do not follow most of the literature. My conclusion was that, because of the *Kaggle* limit of time running scripts, the most

[6] http://nipy.org/nitime/api/generated/nitime.algorithms.autoregressive.html
[7] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

used techniques were not implemented.

Therefore, I decided to use one of the solutions found in the *Kaggle* competition [9] as a base for my solution, with the objective of implementing a more documented approach. While doing research about the problem, I found that most BCI systems, based on SVM classifiers, use as features PSDs generated from AR coefficients, which are obtained by modeling the EEG signals as Autoregressive processes. One way to achieve this is by using the *AR_est_YW* and *AR_psd* functions of the *nitime.algorithms.autoregressive* python library. First, by obtaining the parameters with the *AR_est_YW* function, and then by using the AR coefficients to generate the PSDs using the *AR_psd* function.

BCI literature also mentions that MLP Neural Networks are widely applied to find solutions in order to recognize events of interest. Moreover, the literature indicates that MLP are normally used with features formed by combining AR coefficients of modeled EEG signals. For both cases, the SVM and the MLP, the EEG signals are normally modeled as Autoregressive processes of order 6. Thus, the AR coefficients, for the MLP case, were obtained in the same way as in the SVM case. Finally, the implementation of the supervised Machine learning techniques was possible using the *SVC* and *MLPClassifier* functions from the python *sklearn.svm* and *sklearn.neural_network* libraries, respectively.

Unfortunately, for the SVM case, it was not possible to recognize any event of interest using the EEG signals PSDs as features. Different combinations of parameters were used to train the SVM classifier without any success, such as for example the *kernels*, the C constant and the γ parameters. That was the reason because different features were used instead of the PSDs for the SVM case. Moreover, if the same features are used for both classifiers their performances against each other can be compared better.

The mentioned procedure was implemented using two different scripts, one containing functions that pre-process the EEG signals, while the other contains the implementation of the BCI classifiers, their names are *MLCapstoneFunctions.py* and *MLCapstoneProject.py*, respectively. Additionally, there is a third implemented script which contains functions to visualize the obtained results, the name of the script is *MLCapstoneVisuals.py*.

It is important to remark that, because of the presented problem, BCI systems are normally trained for a specific subject, and therefore, BCI systems need to be trained and tested for each individual independently. For this, because for each subject there are 8 series of experiments, 6 series were used to train the supervised Machine learning classifiers, while the remaining series were used to test their performances. Then, this procedure was repeated for each subject. At the end, the metrics used to measure the performance of the algorithms would consider the performance for all subjects.

[8] https://www.kaggle.com/c/grasp-and-lift-eeg-detection/kernels
[9] https://www.kaggle.com/elenacuoco/simple-grasp-with-sklearn/code

REFINEMENT

The presented problems is related with a complex application and as a results the tuning of the classifiers is a difficult task, especially considering the complexity of the origin of the data. As mentioned earlier, the more standard way to implement a solution for the problem is to use a SVM algorithm as a classifier, using the PSDs on segments of EEG signals as features. Unfortunately, it was not possible to find a solution following that procedure. Probably because of the way the spectrum was used, the frequencies used for the classification or the number of EEG channels used. This will be discussed in more detail later in this document.

As a consequence, different features were used instead of the PSDs, as mentioned before, it was use the AR parameters of segmented EEG signals, for all acquisition channels. In order to find the best settings for the SVM classifier the procedure started by finding the kernels that solved the problem in less that 10 minutes, for one individual. This, because there were kernels that used a lot of time training the method. Then, by setting the parameters manually, the best classifier was trained for all subjects in order to measure its performance. Finally, an optimized version of that classifier was found by implementing a *GridSearch* algorithm, by varying the penalty parameter C for the error, the degree of the kernel since the one selected was polynomial and γ the kernel coefficient. The values used for the GridSearch are presented in table 3, while on table 4 are presented the coefficients used for the manually tuned classifier and for the optimized classifier.

Table 3: Parameters used for the GridSearch

Classifier	**Kernel**	**Penalty (C)**	γ	**Degree**
SVC	poly	1	.01	2
SVC	poly	5	.1	3
SVC	poly	10	1	5

Table 4: SVM classifiers used to solve the problem

Classifier Tuning	**Kernel**	**Penalty (C)**	γ	**Degree**
Manually	poly	1	1	3
GridSearch	poly	1	.1	2

The same procedure was used to select the MLP classifier parameters. First, by finding the parameters of the classifier that solved the problem efficiently, and then, by finding the optimal MLP classifier parameters using the GridSearch algorithm. In table 5 are presented the parameters used for the MLP GridSearch, while on table 6 are presented the MLP classifiers used to solve the presented problem

Table 5: Parameters used for the GridSearch

Classifier	**solver**	**hidden_layer_sizes**	α
MLP	adam	(25,2)	$1e^{-6}$
MLP	sgd	(50,2)	$1e^{-5}$
MLP	lbfgs	(75,2)	$1e^{-4}$

Table 6: MLP classifiers used to solve the problem

Classifier Tuning	**solver**	**hidden_layer_sizes**	α
Manually	lbfgs	(75,2)	$1e^{-5}$
GridSearch	sgd	(25,2)	$1e^{-6}$

Nonetheless, the performance of the classifiers improved using the GridSearch algorithm, but only in terms of the accuracy. The precision was less than results obtained from the manually tuned classifiers. In many cases, the optimal classifiers gave as results models that gives 0 precision. This is important to consider since precision is a measure that let us know how bad a classifier behaves. These results are discussed deeply in the next section.

RESULTS

The results obtained by solving the presented problem were obtained using the trained models to classify, for each subject, each of the events of interest independently, as shown in tables 1 and 2, as binary classifiers. The features were used to train the classifiers for each of the movements independently, using binary labels for each of the movements classifiers. Furthermore, it is important to mention that all results, for all trained classifiers, were obtained using the *random_state = 0*.

MODEL EVALUATION AND VALIDATION

At this point in the development of the Capstone project, there are two version of the SVM classifier that can be considered solutions for the presented problem, with the characteristics presented in the REFINEMENT subsection. As result , it is necessary to find a metric that determines which classifier, between the one tuned manually and the one tuned using the GridSearch algorithm, is more appropriated for this case. It is important to mention that in the *Kaggle Grasp-and-Lift EEG Detection* competition was used as metric to decide the winner the area under the ROC curve. The area under the Receiver Operating Characteristic curve makes sense since is a graphic representation of the sensitivity against the specificity for a binary classifier and the SVM classifier will be used to classify the events of interests independently for each of the subjects, as a binary classifier.

Then, once the metric is estimated for each of the proposed solutions, it would be possible to determine which classifier behaves better. However, other metrics are considered also, such as the accuracy and the precision. Figures 6 and 7 illustrates the metrics obtained for the SVM classifier tuned manually, while in figure 8 and 9 can be observed the metrics obtained for the SVM classifier tuned used the GridSearch algorithm.

Time elapsed: 106.80809670000001[min]

********** BCI Average AR Capstone Project results **********

Metric\Event	HandStart	FirstDigitTouch	BothStartLoadPhase	LiftOff	Replace	BothReleased
Accuracy	0.897	0.899	0.897	0.894	0.898	0.895
Precision	0.169	0.208	0.173	0.148	0.171	0.134

	precision	recall	f1-score	support
HandStart	0.17	0.09	0.11	1624
FirstDigitTouch	0.21	0.11	0.14	1627
BothStartLoadPhase	0.17	0.09	0.12	1612
LiftOff	0.15	0.08	0.10	1631
Replace	0.17	0.08	0.11	1628
BothReleased	0.13	0.07	0.09	1627
avg / total	0.17	0.09	0.11	9749

Figure 6: Metrics obtained training the manually tuned SVM classifier. At the top is presented the duration for training the classifier, and below the accuracy, the precision and results given by the *classification_report* function.

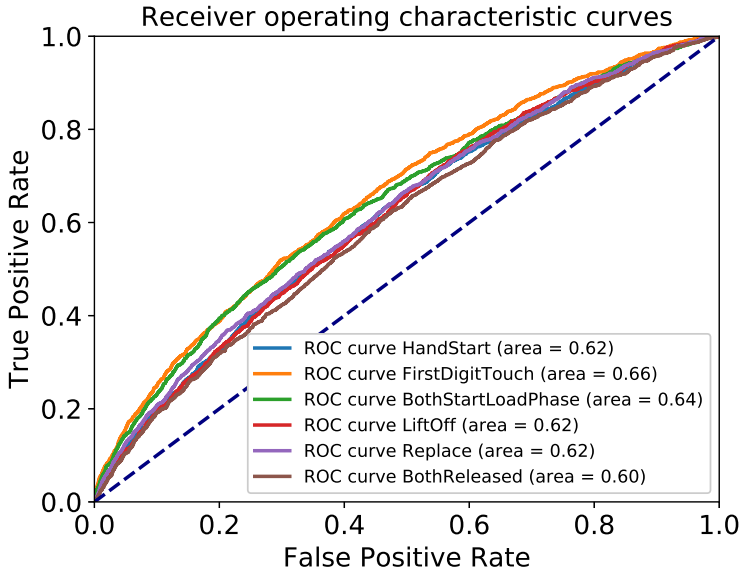

Figure 7: ROC curves for the SVM classifier tuned manually, for each of the events of interest to classify, considering the data for all subjects.

```
Time elapsed: 100.25133871054204[min]
**********************   BCI Average AR Capstone Project results    **************************

Metric\Event    HandStart    FirstDigitTouch    BothStartLoadPhase    LiftOff    Replace    BothReleased
  Accuracy        0.923           0.923                0.924           0.923      0.923        0.923
 Precision        0.000           1.000                0.000           0.000      0.000        0.000

                   precision    recall  f1-score   support

        HandStart       0.00      0.00      0.00      1624
   FirstDigitTouch      1.00      0.00      0.00      1627
BothStartLoadPhase      0.00      0.00      0.00      1612
          LiftOff       0.00      0.00      0.00      1631
          Replace       0.00      0.00      0.00      1628
      BothReleased      0.00      0.00      0.00      1627

       avg / total      0.17      0.00      0.00      9749
```

Figure 8: Metrics obtained training the SVM classifier tuned using *GridSearch*. At the top is presented the duration whdile training the classifier, and below the accuracy, the precision and results given by the *classification_report* function.

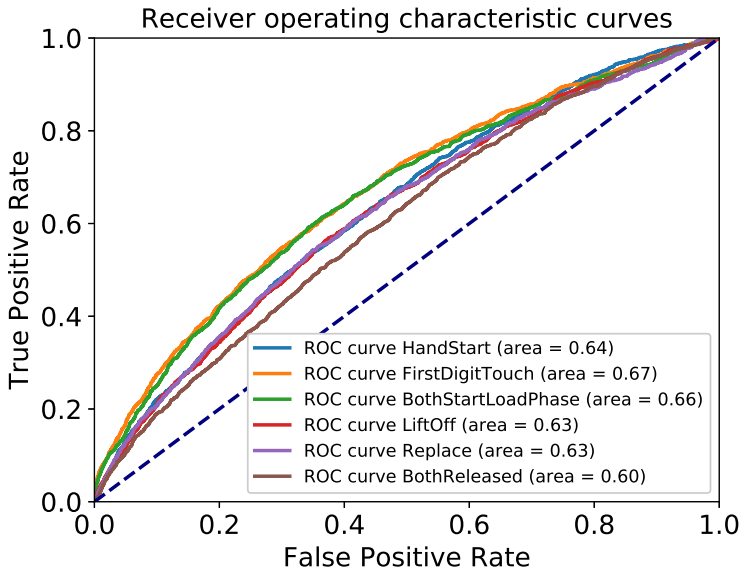

Figure 9: ROC curves for the SVM classifier tuned using *GridSearch*, for each of the events of interest to classify, considering the data for all subjects.

Then, the SVM tuned manually classifier will be used as solution for this problem, because of its precision, as seen before, and sensitivity, as can be seen in figure 10. In figure 11 is show the sensitivity of the benchmark model. For both cases, the random_states 0, 8, 18, 28, 38, 48, 58, 68, 78 and 88 were used to form the sensitivty analysis.

Time elapsed: 1287.606243432107[min]

********** Accuracy of the SVM Model for 10 different cases **********

Metric\Event	HandStart	FirstDigitTouch	BothStartLoadPhase	LiftOff	Replace	BothReleased
Accuracy1	0.897	0.899	0.897	0.894	0.898	0.895
Accuracy2	0.897	0.899	0.897	0.894	0.898	0.895
Accuracy3	0.897	0.899	0.897	0.894	0.898	0.895
Accuracy4	0.897	0.899	0.897	0.894	0.898	0.895
Accuracy5	0.897	0.899	0.897	0.894	0.898	0.895
Accuracy6	0.897	0.899	0.897	0.894	0.898	0.895
Accuracy7	0.897	0.899	0.897	0.894	0.898	0.895
Accuracy8	0.897	0.899	0.897	0.894	0.898	0.895
Accuracy9	0.897	0.899	0.897	0.894	0.898	0.895
Accuracy10	0.897	0.899	0.897	0.894	0.898	0.895

********** Precision of the SVM Model for 10 different cases **********

Metric\Event	HandStart	FirstDigitTouch	BothStartLoadPhase	LiftOff	Replace	BothReleased
Precision1	0.169	0.208	0.173	0.148	0.171	0.134
Precision2	0.169	0.208	0.173	0.148	0.171	0.134
Precision3	0.169	0.208	0.173	0.148	0.171	0.134
Precision4	0.169	0.208	0.173	0.148	0.171	0.134
Precision5	0.169	0.208	0.173	0.148	0.171	0.134
Precision6	0.169	0.208	0.173	0.148	0.171	0.134
Precision7	0.169	0.208	0.173	0.148	0.171	0.134
Precision8	0.169	0.208	0.173	0.148	0.171	0.134
Precision9	0.169	0.208	0.173	0.148	0.171	0.134
Precision10	0.169	0.208	0.173	0.148	0.171	0.134

Figure 10: Average accuracies and precisions for 10 cases of training the SVM model, for 10 different random_states.

Time elapsed: 305.71848889160304[min]

********** Accuracy of the Benchmark for 10 different cases **********

Metric\Event	HandStart	FirstDigitTouch	BothStartLoadPhase	LiftOff	Replace	BothReleased
Accuracy1	0.916	0.919	0.919	0.918	0.919	0.920
Accuracy2	0.916	0.920	0.920	0.921	0.922	0.922
Accuracy3	0.910	0.914	0.915	0.916	0.919	0.915
Accuracy4	0.922	0.919	0.919	0.921	0.920	0.921
Accuracy5	0.921	0.922	0.920	0.922	0.923	0.920
Accuracy6	0.910	0.916	0.915	0.919	0.919	0.914
Accuracy7	0.921	0.916	0.920	0.920	0.923	0.921
Accuracy8	0.923	0.923	0.924	0.923	0.923	0.923
Accuracy9	0.921	0.921	0.924	0.921	0.920	0.921
Accuracy10	0.921	0.920	0.920	0.919	0.920	0.919

********** Precision of the Benchmark for 10 different cases **********

Metric\Event	HandStart	FirstDigitTouch	BothStartLoadPhase	LiftOff	Replace	BothReleased
Precision1	0.188	0.307	0.309	0.197	0.182	0.301
Precision2	0.167	0.335	0.307	0.273	0.276	0.136
Precision3	0.167	0.283	0.293	0.268	0.269	0.192
Precision4	0.312	0.314	0.293	0.176	0.165	0.262
Precision5	0.253	0.320	0.173	0.229	0.214	0.265
Precision6	0.141	0.279	0.259	0.248	0.242	0.157
Precision7	0.276	0.256	0.310	0.258	0.375	0.282
Precision8	0.280	0.000	0.000	0.417	0.429	0.000
Precision9	0.214	0.297	0.000	0.261	0.143	0.227
Precision10	0.259	0.364	0.328	0.212	0.206	0.195

Figure 11: Average accuracies and precisions for 10 cases of training the Benchmark model, for 10 different random_states..

Unfortunately, the chosen model does not behave better than the selected benchmark if the averaged metrics are considered, benchmark that was also tuned manually since the one tuned using the GridSearch algorithm gave zero precision for all movements of interest. Moreover, the area under the Receiver Operating Characteristic curves for all events of interest were smaller than the ROC values obtained from the MLP classifier benchmark. In figures 12 and 13 are presented the metrics obtained from the MLP tuned manually, which can be used to measure the performance of the proposed SVM classifier against the performance given by the selected benchmark.

Consequently, by comparing figure 6 against figure 7 and figure 13 against figure 12, it can can be seen that the benchmark classifier, the MLP Neural Network, has a better performance than the SVM proposed classifier. However, if it is observed the behavior of both binary classifiers during the recognition of movements of interest for all subjects, then the SVM classifier behaves better than the benchmark, as can be seen in figure 14. It behaves better in terms of the precision, because the MLP precision is affected by very large outliers and because the median of the metric let us know that there are more probabilities to recognize a movement incorrectly in comparison with the SVM classifier. In terms of the accuracy metric, both classifiers behaves similarly, but the MLP got better averaged results than the SVM classifier.

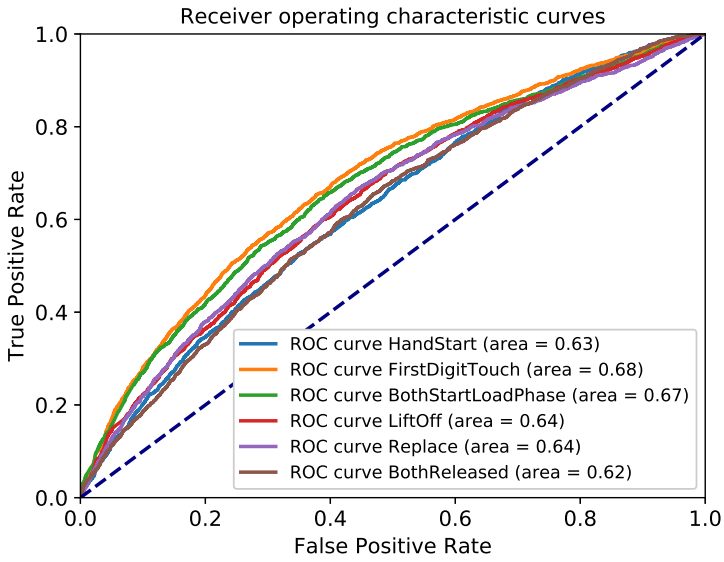

Figure 12: ROC curves for the MLP classifier tuned manually, for each of the events of interest to classify, considering the data for all subjects.

```
Time elapsed: 34.98258434097054[min]

************************   BCI Average AR Capstone Project results    ***************************

Metric\Event    HandStart    FirstDigitTouch    BothStartLoadPhase    LiftOff    Replace    BothReleased
  Accuracy        0.916          0.919                0.919           0.918      0.919        0.920
 Precision        0.188          0.307                0.309           0.197      0.182        0.301

                    precision    recall  f1-score   support

         HandStart       0.19      0.03      0.05      1624
    FirstDigitTouch      0.31      0.04      0.08      1627
BothStartLoadPhase       0.31      0.05      0.08      1612
           LiftOff       0.20      0.02      0.03      1631
           Replace       0.18      0.01      0.03      1628
      BothReleased       0.30      0.03      0.05      1627

         avg / total       0.25      0.03      0.05      9749
```

Figure 13: Metrics obtained training the manually tuned MLP classifier. At the top is presented the duration for the classifier training, and below the accuracy, the precision and results given using the *classification_report* function.

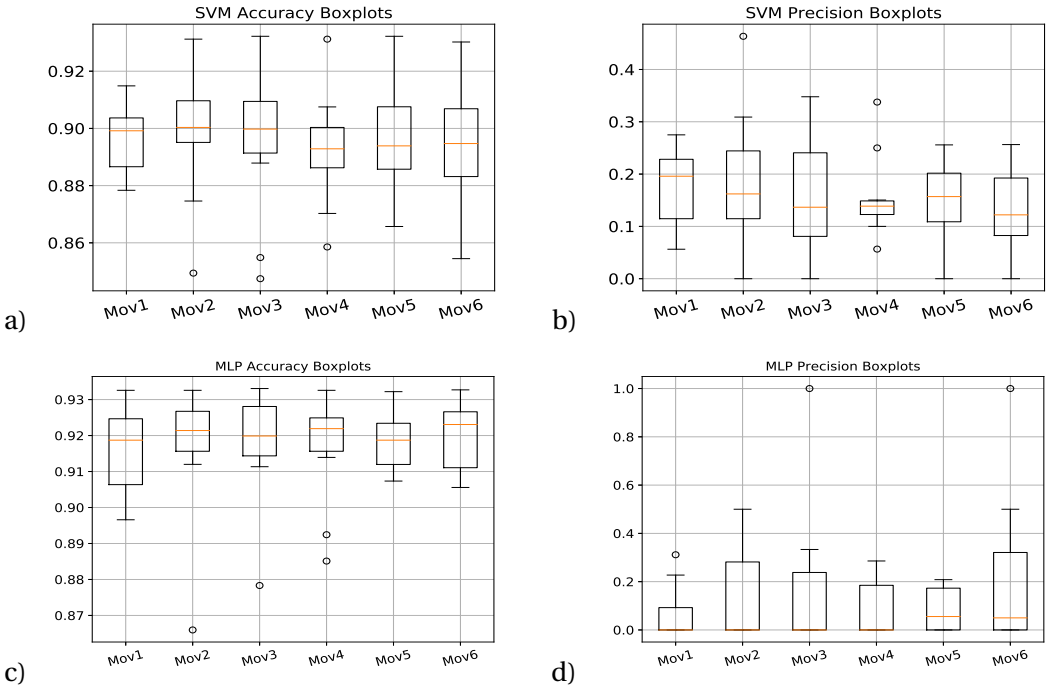

Figure 14: In a) and b) are presented the statistics for the SVM classifier in the form of box-plots, while on c) and d) are presented the statistics of the benchmark classifier.

CONCLUSION

FREE-FORM VISUALIZATION

As can be seen in figure 15, the features used for training the classifiers can varied significantly, from using the signals time-domain amplitudes, to the use of their AR parameters or their spectrum generated from the AR parameters. Even when I could not implement BCI systems based on the PSD of the EEG signals as features, literature says that it is possible. This characteristic is an important quality of the Machine Learning algorithms, which shows their adaptability and robustness solving problems, even when the same problem is faced in different ways.

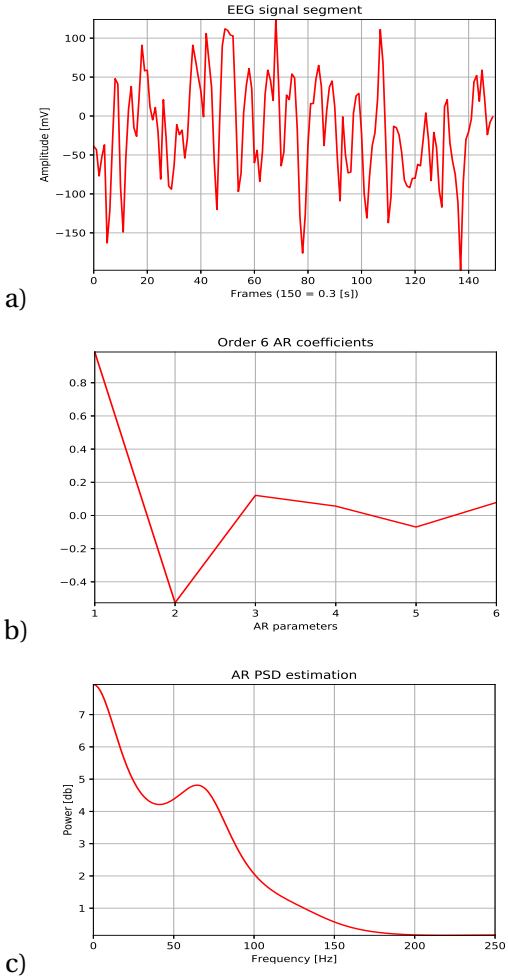

Figure 15: In a) is presented a segment of one channel EEG used data, while on b) and d) are presented the AR parameters of the segmented signals and the PSD obtained using the AR parameters, respectively.

REFLECTION

As conclusion, the development of BCI systems involves many uncertainties that are difficult to handle because of the brain complexity, that affected the training of the classifiers implemented. One of the most problematic characteristics was the data used to train the classifier, not only because of the complexity of the brain, but also because was obtained from the *kaggle* webpage, and even when its acquisition is well documented there will always exist unknown variables to consider. Additionally, limitation in knowledge about the topic was a big issue, for example, the amount of channels used to train the classifiers could be decreased if I had more experience relating brain processes with its anatomy. Nevertheless, it was gratifying to developed an algorithm able to recognize events of interest from brain signals, even when its performance was not the expected.

IMPROVEMENT

As mentioned in the previous subsection, the amount of data used for training the classifiers can be reduced, especially since the data was acquired from channels that can be correlated between them, and therefore, there will be redundant data, as shown in figure 16. For example, there exist literature where just a few channels are used for the recognition of similar movements [Hosni et al., 2007, Huan and Palaniappan, 2004]. This could reduce the time required to train the classifiers and to improve their performances. Figure 16 show the amount of correlation between the acquired EEG channels, for each subject.

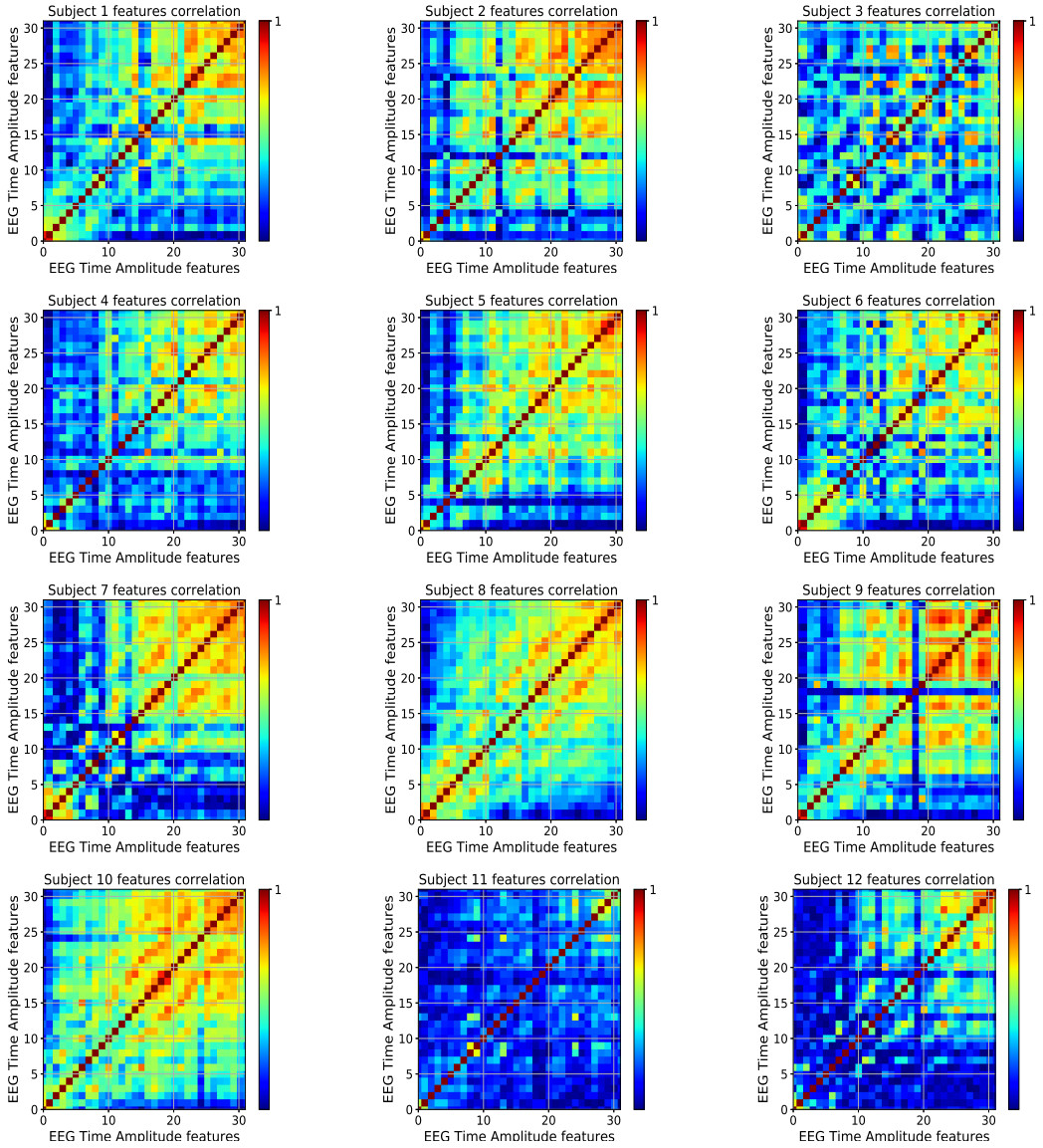

Figure 16: Correlation of featurers for the data obtained on each experiment, for each subject. In this case the feautures were formed using the order 6 AR parameters for each of the EEG channels.

REFERENCES

Birbaumer, N., 2006. Breaking the silence: brain–computer interfaces (bci) for communication and motor control. Psychophysiology 43 (6), 517–532.

Cecotti, H., Graser, A., 2011. Convolutional neural networks for p300 detection with application to brain-computer interfaces. IEEE transactions on pattern analysis and machine intelligence 33 (3), 433–445.

Fabiani, G. E., McFarland, D. J., Wolpaw, J. R., Pfurtscheller, G., 2004. Conversion of eeg activity into cursor movement by a brain-computer interface (bci). IEEE transactions on neural systems and rehabilitation engineering 12 (3), 331–338.

Finger, S., 2001. Origins of neuroscience: a history of explorations into brain function. Oxford University Press.

Guger, C., Schlogl, A., Neuper, C., Walterspacher, D., Strein, T., Pfurtscheller, G., 2001. Rapid prototyping of an eeg-based brain-computer interface (bci). IEEE Transactions on Neural Systems and Rehabilitation Engineering 9 (1), 49–58.

Hosni, S. M., Gadallah, M. E., Bahgat, S. F., AbdelWahab, M. S., 2007. Classification of eeg signals using different feature extraction techniques for mental-task bci. In: Computer Engineering & Systems, 2007. ICCES'07. International Conference on. IEEE, pp. 220–226.

Huan, N.-J., Palaniappan, R., 2004. Neural network classification of autoregressive features from electroencephalogram signals for brain–computer interface design. Journal of neural engineering 1 (3), 142.

Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., Arnaldi, B., 2007. A review of classification algorithms for eeg-based brain–computer interfaces. Journal of neural engineering 4 (2), R1.

Luciw, M. D., Jarocka, E., Edin, B. B., 2014. Multi-channel eeg recordings during 3,936 grasp and lift trials with varying weight and friction. Scientific data 1, 140047.

Schlögl, A., Lugger, K., Pfurtscheller, G., 1997. Using adaptive autoregressive parameters for a brain-computer-interface experiment. In: Engineering in Medicine and Biology Society, 1997. Proceedings of the 19th Annual International Conference of the IEEE. Vol. 4. Ieee, pp. 1533–1535.

Zetterberg, L. H., 1969. Estimation of parameters for a linear difference equation with application to eeg analysis. Mathematical Biosciences 5 (3-4), 227–275.