MACHINE LEARNING AND BRAIN STATES

A PREPRINT

Gabriel Henrique Alencar Medeiros
Department of Information Systems Architecture
INSA Rouen Normandie
685 Avenue de l'Université, 76800 Saint-Étienne-du-Rouvray
hippo@cs.cranberry-lemon.edu

Stéphane Canu
Department of Information Systems Architecture
INSA Rouen Normandie
685 Avenue de l'Université, 76800 Saint-Étienne-du-Rouvray
stephane.canu@insa-rouen.fr

Guilherme de Alencar Barreto
Department of Teleinformatics Engineering
Federal University of Ceará
Humberto Monte, s/n - Pici, Fortaleza - CE, 60440-593
gbarreto@ufc.br

May 28, 2019

ABSTRACT

In this experiment, we will try to find a tool capable of finding the best method that best fits a binary classification problem on Brain Computer Interface (BCI) Data without using a neural network. So basically, we going to explore the problem, find other past results, try different methods with cross-validation, and finally we will compare the results.

***K*eywords** Machine Learning · Brain States

1 Introduction

The brain and the waves within the brain are extensively studied, and yet we have many questions. The study of brain state is an area that can be really useful for future applications, such as for machines that can respond to human thinking or for machines that can speak the words we can think of. So finding a method that not only can predict the state of mind, but can do it quickly is really important.

The brain is an electrochemical organ. All this electrical activity is responsible for different types of brain waves. Throughout the day, the brain keeps active the 5 types of brainwaves (some act at low frequency, others at a higher frequency). Depending on what we do at any given moment, some waves will show greater activity in certain areas of the brain, and others will work less intensely in other regions. None of them will be, in other words, "disconnected." They are:

1. Delta waves (1 to 3 Hz) : Delta waves are those that have a higher wavelength and are related to deep sleep (but without dreams) and bodily activities that we are not aware of, such as heart rate regulation or digestion ;
2. Theta waves (3.5 to 8 Hz) : is related, above all, to the imaginative capacities, the reflection and the sleep ;

3. **Alpha waves (8 to 13 Hz) :** Appear in the midnight dusk where there is calm, but not sleep, where there is relaxation and a favorable state to meditate. A high level of Alpha waves would prevent us from focusing attention ;

4. **Beta waves (12 to 33 Hz) :** Accented by states that are related to the daily activities in which we focus all the attention, when we are alert and we need, at the same time, to be attentive to numerous stimuli ;

5. **Gamma waves (25 to 100 Hz) :** This wave is related to tasks of high cognitive processing. They have to do with our learning style, with the ability to record new information, and also with our senses and perceptions, states of happiness and the phases of REM sleep.

Of the 5 waves, the last three of them can be divided into low and high frequencies, because they have strong components in both bands.

Knowing the different types of brain waves allows us to understand the mental processes, emotions, activities and dynamics that generate a type of "energy" in the brain. The Alpha, Beta and Gamma waves are very interesting for recognition of brain states linked to actions, feeling and concentration.

2 The Problem

We want to find a classification model that identifies different mental states. In other words, given a range of possible mental configurations and a set of training data, it is desired to find a model capable of predicting such states from similar data.

Our intention is to compare different models already existing in the literature to observe which models have the best performance in the bci (brain computer interface) data.

3 Description of the data

3.1 Introduction to the Data

We chose to work with data from the Kaggle - Synchronized Brainwave Dataset[1]. This data set consists of an experiment with 30 patients in an experiment where each one watched a video showing different mental tasks that should be performed. To simplify our methods, we will focus only on the first experiment that is an experiment that measures the level of concentration of a patient when he makes mental calculations of different equations.

3.2 Description of the type of methods

We will be working with a binary classification algorithm, since we want to know if the patient is relaxed or concentrated. We will use supervised methods since we already know the output types and we have a set of data already with labels.

3.3 Analyzing the data

We have 30 participants, each of whom has performed the experience more than once, so we have 571 points, approximately half are from relaxed states and the other is concentrated in mathematical calculations.

Next, the class histogram is shown. As there are 100 features, it is a little difficult to see important information, however, it is noticed that all attributes have values greater than zero in both classes.

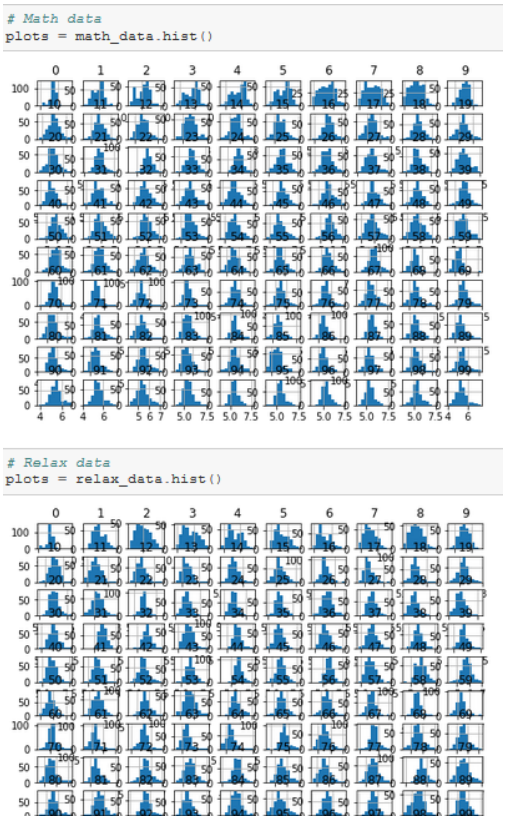

Figure 1: Histogram.

Then, the covariance matrix is shown. It was very important to decide some hyperparametres, because it is noted that the first 10 features have a strong correlation while we have some mime-weak correlations onwards, so it is important to take into account more than ten variables in some hyper parameters.

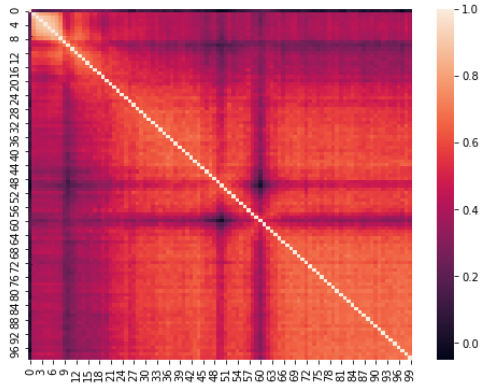

Figure 2: Covariance matrix.

Finally, the PCA shows that the points fit very close to a common center and after we can see some isolated points many distant from the center points, and possible outliers may be possible.

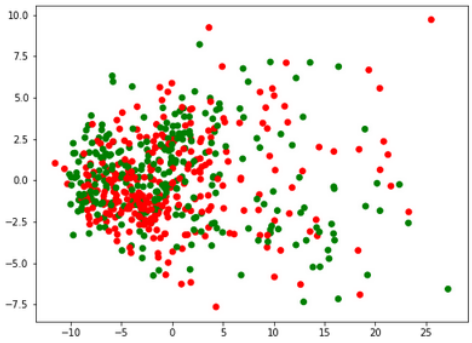

Figure 3: PCA.

4 Other Results

4.1 Other results for this data set

In this section we will talk about results already obtained by other people with the same data. On the site Kaggle - Synchronized Brainwave Dataset[1], several people have posted their codes with their results so that everyone could see and reuse the code, but only two posts are really interesting for us:

1. Classifying relaxation versus doing math[2]

2. math/relax prediction with different classifiers[3]

4.2 Classifying relaxation versus doing math

The interesting thing about this code is the choice of the data that is preprocessed using the Fourier transform, averaging groups of 3 power spectra and logarithmically grouping the result to produce feature vectors of 100 values.

The problem with the code was that the author only used one person as a training set and another as a test, so he does not have a robust model.

4.3 Math/relax prediction with different classifiers

This one is really very well done. He used 8 possible brain waves as attributes, much less than the 512 attributes of raw data. Other than that, the author performed a good analysis of the data and tested different methods on the data. He got a good precision with SVC (Support Vector Classification), but he showed in the end that using the preprocessing of the data set from the previous solution is even better.

5 Methods

5.1 Linear Methods

5.1.1 LDA - Linear discriminant analysis

It is a method used to find linear combinations of features that characterize or separate elements of different classes, resulting in the combination of features a linear classifier. [5]

5.1.2 The logistic regression

It estimates parameters of a logistic function to model a binary dependent variable, thus creating a classification model that assigns probabilities to the results of the predictions and decided which of the classes the element has a greater chance of belonging. [8]

5.1.3 Linear SVM

It is a linear classification method where the model is based on finding a separation boundary (hyperplane) between data of different classes in a way that maximizes the distance between the points closer to the border in relation to each of the classes. [6]

5.2 Nonlinear Methods

5.2.1 Naive bayes classifier

It is a technique to construct classification models that build upon several algorithms that assume that the value of a particular features is independent of the other features. [4]

5.2.2 QDA - Quadratic Discriminant Analysis

Like the LDA, it is assumed that the measures of each class are normally distributed, but different from the LDA, it is not assumed that the covariance between classes is identical, which introduces a quadratic term in the equation making the model non-linear. [7]

5.2.3 Gaussian Mixture Model - GMM

It is a type of clustering algorithm that assumes that the data is generated from mixtures of Gaussian distributions (clusters) with unknown parameters. Unlike a K-means, for each element a probability is associated and how far the element is from a center, the smaller the probability. The element will belong to a class if the probability for that same class is maximal. [9]

5.2.4 K-nearest neighbors - KNN

It is a simple but efficient algorithm where given a value k, for each point that it wishes to predict, the algorithm will look for the k points whose distance is the smallest and will decide the value of the new point with the most common class of neighborhood. [10]

5.2.5 Random Forest and AdaBoost

Both are methods based on decision trees.

Random Forest is a method based on joint learning that builds several decision trees during the training and realizes predictions by voting on the result of each tree. Random Forest benefits from two random principles for training: Bagging (random subsets) and Random Feature Selection.[12]

Adaboost is an iterative and self-adaptive method that each subsequent classification is adjusted from the instances classified incorrectly by previous rankings. It is a noise-sensitive method, but is less susceptible to overfitting than most algorithms. [11]

5.2.6 Kernelized SVM

A kernel is a function that takes two points of a space in another space called the Hilbert kernel space. With this simple change of space, it may take a problem to solve a linear problem in a non-linear space. This brings a myriad of different solutions, and an obvious conclusion would be to use a linear classifier known in the literature that offers great results. Hence we have an SVM with Kernel, where we create a decision border with inputs mapped in another space by several Kernel functions. [6]

5.3 Hyper parameter management

There are two major concerns when creating a machine learning model: performance and reliability.

It is not clear if the first parameters used in the tests will result in the best possible model. For this, there are techniques for finding the best parameters within a set, such as Random Subsampling, Leave-one-out Cross-Validation, Bootstrap, and K-Fold Cross-Validation. All of them are based on dividing the set into two subsets, one of training and one of validation and from the training set, the model will be trained with a parameter configuration and from a metric the model will be evaluated by the validation subset . After repeating this process for all parameter settings, the best will be chosen.

However, it is not known if the model is reliable for future data not used in the training, so it is necessary to have a test set to ensure that the model will work well. This set can not be involved with the training and validation process, so that the result is even more consistent with reality.

5.4 Pipeline

A machine learning pipeline is a set of processes where the result of one process is input from another, and each process acts together with others to generate better results.

Normally, they happen like this: given the definition of the problem, the data is placed in the input of the pipeline, which will first perform the data preparation / preprocessing. After this, the data will be divided into training sets, validation and test with the proper procedures and the model will be trained, and soon after evaluated, repeating the last two processes an arbitrary amount of times. After this, the molding will be implanted and will undergo a process of performance monitoring.

An example of a pipeline used in our case: In preprocessing, the best k features are chosen and the data is transformed to have only those features. Then, a KNN classifier is realized by tuning the hyper-parameters (the number of neighbors).

5.5 Automated Machine Learning - AutoML

This is a set of methods and models of machine learning. It realizes groups of data preprocessing, features engineering, features extraction and features selection. Then it selects an algorithm to make the tuning of the parameters. However, unlike most other techniques, it automatically uses the results of a machine learning model / technique to improve the next model. In the end, the best model is chosen without requiring human intervention during the process, thus reducing bias. It is a relatively new technique in relation to the classical machine learning algorithms, and demands a high computational cost, and needs to be taken care of while applied, since the algorithm itself is responsible for transfer learning, so it is possible that the algorithm does not provide a parameterization that is optimal. [13]

6 Methodology

6.1 Preanalysis

First, the Synchronized Brainwave Dataset data was downloaded and imported. After that, only information that is useful is taken, which in this case is each group of 512 raw values of each person and values linked to whether the person is relaxed (y = 0) or concentrated in mathematical calculations (y = 1).

When one has the data ready, it is necessary to make a preliminary analysis in the data, to obtain information as mean, standard deviation, the covariance matrix and histograms with respect to each class. Data like these help in analyzing the problem and helps to regulate hyper parameters of the methods.

6.2 Pretreatment

For pretreatment, each group of 512 raw values produced by the device was taken, and a fast fourier transformation (FFT) was performed to produce a spectrum of energy. Then groups of 3 power spectra are picked up, averaged and the result is logarithmically grouped to produce feature vectors of 100 values.

After the preanalysis and pretreatment, it is necessary to separate the data into training and test sets. As the data is very scarce, it was preferred to follow the procedure performed in the Kaggle forum and to divide it together in training and validation only, because when new data is collected, it will be possible to create a test set and even increase the sets of training and validation. Following this reasoning, you can also compare our models with the forum templates.

6.3 Method handlers

Basically, experiments were carried out with 2 classes in python:

1. BCIBinaryClassifierModel ;
2. BCIBinaryClassifierModelCrossValidation.

The first class is to have and manage different classification methods, as well as their performances in a given data set, besides being able to organize the values in a results table.

The second class has as input a first class object and a list of parameters (if not given, it has a standard list), and perform a cross-validation to find out the best parameter for each method and save them to show them later on a table.

Both classes are instantiated and executed separately to observe the effects of the raw validation on the hyper parameters of each method mentioned previously.

6.4 Attempts

For each hyper parameter of each method, a study was performed to observe all the possible inputs or group of values that embody optimal parameters within the range of their extremes. For this choice, we took into account elements such as data, quantity of attributes and features. Also used were values found in the literature of each method.

The codes have been executed countless times in various situations. In some methods the data shuffling increased the performance a little, but decreased the performance of others. The preference was to separate the sets by placing the data of each person entirely and in order in each group. This proved to be more effective for methods that already had the greatest performances.

In this case, the normalization of the data in relation to the training data worsened performance a little compared to the original values. It was tested to pick up only the 8 commonly recognized frequencies (delta (0.5 - 2.75Hz), theta (3.5 - 6.75Hz), low-alpha (7.5 - 9.25Hz), high-alpha (10 - 11.75Hz), low-beta (13 - 16.75Hz), high-beta (18 - 29.75Hz), low-gamma (31 - 39.75Hz), mid-gamma (41 - 49.75Hz)), which gave results a little close but still lower, but it is worth noting that the amount of features is much smaller, less data is needed to train and higher training speed of some methods.

6.5 AutoML

Regarding AutoML, some precautions have been taken. Many processes are performed on the data, however, two of them are those that need more attention, the first one is that AutoML uses 15 classifiers, the second is that 14 preprocesses of features are used. Therefore, at least 210 different models must be taken into account in relation to training time. As AutoML parameters, you can define which types of classifiers and preprocessing will be used, but since you do not have much information to know exactly which ones will be important, all default values have been left unchanged. However, the time each set of methods has to perform and the total time of the operation were two values changed. It was first defined that the algorithm would run for 3 days, that is, 259200 seconds. Since we want to test at least 210 models, each model must have at least 259200/210 seconds to run, that is, they have 1235 seconds to run.

7 Results

The code was run twice to show the consistency of the created models. There may be some fluctuations in values, but the results of each method are always in the same ranges.

Here is the tables of methods without cross validation, using only the default parameters.

```
bci_model.showResults()
```

	Naive bayes classifier	LDA	Logistic regression	Mixture model	Mixture model Bayes	k nearest neighbors	k nearest neighbors Cross Validation	k nearest neighbors Grid
Precision on a test set	54.455446	53.960396	55.940594	49.50495	50.0	58.910891	52.475248	58.910891

	Pipeline VS_knn	Pipeline + HP_tuning	Linear SVM Linear	Linear SVM Grid	Random Forest	Ada Boost	Kernel SVM
Precision on a test set	53.960396	56.435644	56.435644	53.465347	53.960396	50.990099	50.49505

Figure 4: Methods without cross validation (run 1).

```
bci_model.showResults()
```

	Naive bayes classifier	LDA	Logistic regression	Mixture model	Mixture model Bayes	k nearest neighbors	k nearest neighbors Cross Validation	k nearest neighbors Grid
Precision on a test set	54.455446	53.960396	55.940594	50.0	49.50495	58.910891	52.475248	60.39604

	Pipeline VS_knn	Pipeline + HP_tuning	Linear SVM Linear	Linear SVM Grid	Random Forest	Ada Boost	Kernel SVM
Precision on a test set	53.960396	61.386139	51.980198	55.940594	49.009901	50.990099	50.49505

Figure 5: Methods without cross validation (run 2).

Next, it is possible to note how much critical validation is important. The performances improved greatly, with special attention to the Gaussian mix models and Random Forest, which obtained good performance.

```
bci_model_cross_validation.showResults()
```

	Naive bayes classifier	Logistic regression	Mixture model	Mixture model Bayes	k nearest neighbors	k nearest neighbors Cross Validation	k nearest neighbors Grid
Precision on a test set	54.455446	57.920792	65.346535	65.346535	58.910891	60.39604	58.910891

	Pipeline VS_knn	Pipeline + HP_tuning	Linear SVM Linear	Linear SVM Grid	Random Forest	Kernel SVM	AdaBoost
Precision on a test set	61.386139	58.910891	58.910891	55.940594	66.831683	62.871287	63.861386

Figure 6: Methods with cross validation (run 1).

```
bci_model_cross_validation.showResults()
```

	Naive bayes classifier	Logistic regression	Mixture model	Mixture model Bayes	k nearest neighbors	k nearest neighbors Cross Validation	k nearest neighbors Grid
Precision on a test set	54.455446	58.415842	66.831683	66.831683	58.910891	58.910891	55.445545

	Pipeline VS_knn	Pipeline + HP_tuning	Linear SVM Linear	Linear SVM Grid	Random Forest	Kernel SVM	AdaBoost
Precision on a test set	61.386139	58.415842	58.910891	53.960396	67.326733	62.871287	63.861386

	Auto ML
Precision on a test set	58.415842

Figure 7: Methods with cross validation (run 2).

Here is an evolution of the attempts with AutoML, and we realize that the more we work on the parameters and change the training time, the better the performance will be (taking into account also the processing power of the computer).

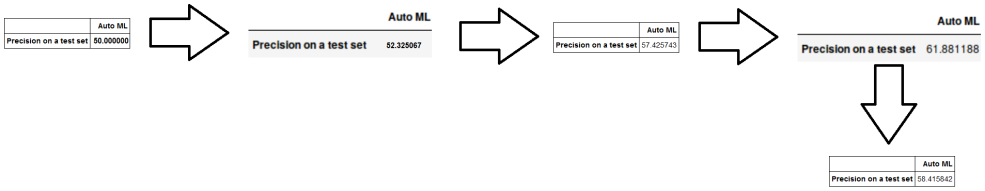

Figure 8: AutoML progress.

It is possible to note that the last value was not bigger than 61 percent accuracy. Therefore, different situations can be taken into account: perhaps overfitting has occurred, perhaps training has taken into account sets of methods that do not provide good estimates for a long time, among other factors.

However, the AutoML still has a lot of potential. To improve, there are many options like: to discover the configurations set that do not provide good performance and remove them, or to increase the number of data, or to change training times, and so on.

Finally, it is shown all the best parameters for each method that used cross validation.

Best Parameters

```
bci_model_cross_validation.showBestParams()
```

	Logistic regression	Mixture model	Mixture model Bayes	k nearest neighbors Cross Validation
Best params of test set	{'lr_penaltys': 'l2', 'lr_solvers': 'saga'}	{'gmm_n_components': 3, 'gmm_tol': 0.888888888...	{'gmm_bayes_n_components': 3, 'gmm_bayes_tol':...	{'knntest_size': 3, 'knn_random': 0.44, 'rando...

	Pipeline VS_knn	Pipeline + HP_tuning	Linear SVM Linear	Random Forest	Kernel SVM
Best params of test set	{'anova k': 1, 'svcc': 1}	{'anova k': 4, 'n_neighbors': 1}	{'svmC': 0.001}	{'maxdep': 200, 'estimators': 2000, 'mfeatures...	{'sgamma': 0.1, 'sC': 1, 'skernel': 'rbf'}

Figure 9: Best parameters.

8 Conclusion

It is easy to see that normally the performances would not be very good, but that with a tunning of hyperparametres one can find better performances than those found in the old results. After that, there are numerous ways to increase performance, one of them being the possibility to increase the number of people in the database.

What is important to note are the methods that gave above 60 percent and especially those that gave above 65 percent because they have shown themselves to be capable of problems involving BCI data, and AutoML is a powerful tool that if handled well can result in performance better than all the performances shown here and hence better than the old results.

Finally, as a result of this experiment, a tool was created that not only is able to import, analyze and pre-treat data related to mental states, but also to test different methods and see which are the best results for the most classical methods and AutoML. Also, the same experience with different data or data captured by better sensors can result in models with great performances.

References

[1] https://www.kaggle.com/berkeley-biosense/synchronized-brainwave-dataset

[2] https://www.kaggle.com/seaneuron/math-relax-prediction-with-different-classifiers

[3] https://www.kaggle.com/elsehow/classifying-relaxation-versus-doing-math

[4] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[5] https://en.wikipedia.org/wiki/Linear_discriminant_analysis

[6] https://pt.wikipedia.org/wiki/M%C3%A1quina_de_vetores_de_suporte

[7] https://en.wikipedia.org/wiki/Quadratic_classifier#Quadratic_discriminant_analysis

[8] https://en.wikipedia.org/wiki/Logistic_regression

[9] https://scikit-learn.org/stable/modules/mixture.html https://towardsdatascience.com/gaussian-mixture-modelling-gmm-833c88587c7f

[10] https://scikit-learn.org/stable/modules/mixture.html https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[11] https://pt.wikipedia.org/wiki/AdaBoost

[12] https://en.wikipedia.org/wiki/Random_forest#Algorithm

[13] https://www.kdnuggets.com/2019/01/automated-machine-learning-python.html