Université Paris Cité

Data series Classification: Automatic detection of epileptic spike on electroencephalography

Guillaume Dufau, Samy Megueni, Logan Servant, Aïssatou Signate
Graduate School of Computer Science and Mathematics
Université de Paris Cité
Paris, France

Abstract

Epilepsy is a central neurological disorder in which brain activity becomes abnormal, causing seizures or periods of unusual behavior, sensations and sometimes loss of awareness. Seizure detection is a routine process in epilepsy units requiring manual intervention of well-trained specialists. This process could be extensive, inefficient, and time-consuming, especially for long term recordings, making electroencephalography epileptic spike detection crucial the field of Neuroscience. In this paper we present a method to detect automatically if a series represents a series of epileptic spikes or a normal electroencephalography series. Our classification algorithm was applied on a dataset consisting of multivariate data series with a constant frequency, taken from two different experiments and preprocessed for desensitization. The method has an accuracy of 94.41% with a Gradient Boosted Tree model.

1. Introduction

Electroencephalography is a method used since the early twentieth century to detect brain tumors, brain damage or dysfunction, inflammation, and, most importantly, epilepsy or seizure disorder, by placing electrodes on the scalp of a patient. Most electroencephalographies are done with a great number of electrodes and each one records the activity of the area of the brain where the electrode is placed on. Typical electroencephalographies record twenty to thirty minutes of brain activity for each patient and each electrode on the scalp.

Electroencephalographic methods create time data series which are read by an expert's eye; some methods have been implemented for computers to read data, process, and classify which type of brain activity the computer is dealing with.

Epilepsy is a very common if not one of the most common neurological disorder. It develops a tendency for the body to enter a seizure state. One of the most recurrent patterns of epilepsy is called Spike-and-wave in Absence epilepsy (primary generalized epilepsy which causes very brief seizures).

However, the line between an epileptic spike and normal brain behavior can be fragile and there is not a plethora of epileptic spike behavior to work with when compared with the normal activity of recorded brain behavior in patients. Some experts cannot detect precisely if a given data series is an epileptic spike or usual behavior. This classification between normal and unusual brain activity has thus been one of the main challenges of both data science and neural science. Computer assistance has gained a lot of traction for the last decade or so because of machine learning improvement, Big Data, and new Data Mining algorithms. Computer Science and Data Science are therefore of tremendous importance because of the speed of data processing it can reach on a very large set of data points. Automatic analysis and classification of a given data series are to be used as a decision support tool for professionals who have to decide whether patients must follow a given treatment or not according to the activity of their brain during the recording of the electroencephalography. Technology thus allows help tremendously professionals by largely reducing the time it takes for a decision to be taken.

2. Problem

Epileptic spikes being a minority compared to normal behavior and epileptic spike morphologies across different patients are the two prominent problems we face in this paper. Our goal is to use the data we have access to (composed of two labeled records of 5 electrodes containing each 768 data points, with each record being from two distinct patients), and implement a machine learning algorithm that can accurately classify if a given data series from any patient is an epileptic behavior or a normal brain activity. The second goal of the algorithm is to be able to display the highest sensitivity possible while still having a low specificity.

Automatic epileptic spike detection has been a recurrent problem treated by computer scientists and data scientists for the past few years, starting with the emergence of deep learning algorithms in the early years of the last decade.

Electroencephalographic data has been processed using Convolutional Neural networks, Sparse Representation enhanced deep learning neural networks, and Markov Models coupled with Discrete Fourier and Discrete Cosine transforms. Some other algorithms tend to use other classification methods, such as K-Means and the Discrete Wavelet Transform (DWT). The former method, the Discrete Wavelet Transform, tends to be used in many processing electroencephalographic time series. Support Vector Machine is also recurrently cited in papers as a method providing satisfying results.

Outside of epileptic spike analysis, a few papers used electroencephalographic data using other methods: Some papers took a statistical approach such as Shannon Entropy and multi-class Support Vector Machine for emotion recognition or used statistical approaches for the pre-processing of the electroencephalographic data analysis, such as Statistical Thresholding (FASTER), or Robust statistical for the pre-processing pipeline.

3. Solution

The team's first approach was to try and experiment with what we had read thus far about solutions and the state of the art of processing electroencephalographic data points, for either, epilepsy spike detection or other objectives. Our goal was to get inspired by former methods and try to upgrade and fix what they couldn't perform well.

Our first experiment was to use time series to try and reduce the noise in the signal: most of the electroencephalographic signals recorded, if not all of them, include strong noise created by the subject's eye motricity and other variables in the room at the moment of the recording such as sound and odor in the room, which is an important aspect to account for in the processing of the signal. Time series ease the use of the Discrete Fourier Transform algorithm (or Cosine Fourier Transform, both of which we have used and tried) by minimizing noise and creating almost continuous and pseudo-periodic sine waves for each of the five signals in each data series.

Applying other data analysis algorithms (which are described below) and then a classification method, we have found that transforming data to fit a pseudo-sinewave signal was not the best approach to take because the noise was indeed reduced but a lot of data was lost in the process, thus rendering this method obsolete, or worse.

Since our dataset could be described as a signal, we tried working with some functional analysis transforms, such as Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), or Discrete Wavelet Transform (DWT).

After a few Trial-and-Error attempts, and some comparisons of computing time and effectiveness of each function, we decided to keep only the best method and not use multiple ones at once; we thus decided to base our method on the Discrete Wavelet Transform. The method possesses diverse ways to treat signals, so we had to choose one: Haar Wavelet was not fit because of the size of our dataset, so we went for the 2D-Daubechies method and decided to base our classification on the Daubechies coefficients.

Using what we had learned about the differences between epilepsy spikes and normal brain behavior, and in view of articles we had read, we tried to consider other features for classifying the two types of brain activity using statistical methods. Our first instinct was to compare the data we had access to, using MatPlotLib in Python. We computed many statistical variables such as the mean of the data points and found very different values in the boxplot of positive and negative instances of an epilepsy spike.

Below is the Boxplot of the mean of absolute values of the second out of the five data points of each instance.

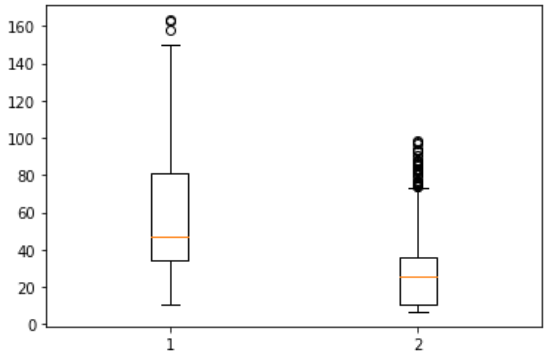

Figure 1.: boxplot of the mean: On the left, positive instances of epilepsy spike, on the right: negative instances.

From this point onward, we tried to find other statistical data we could compute that were discriminative enough, fast to compute, and that the reason we use any variable is relevant to the problem of epilepsy spike detection.
Going back to the definition of an epilepsy spike given in the introduction, we thought it could be relevant to use the minimum and the maximum values of each instance, because of how the signal could vary at any point to a very extreme value in positive instances. As such, we found that on average the minimum and maximum values were discriminative enough variables but, by adding both values for each instance (and by using an absolute value for the minimum that most often reached a negative value), we could obtain two boxplots that almost do not overlap each other, which could be very useful for the model we are trying to build (see Figure 2.).

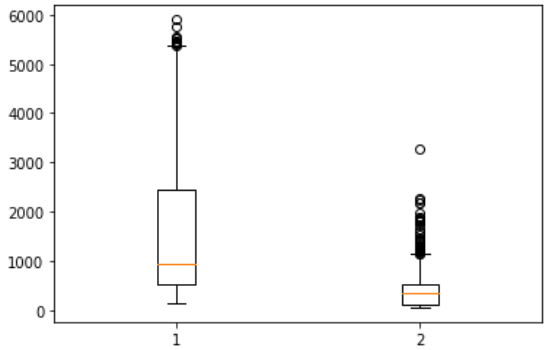

Figure 2.: Boxplot of the sum of min and max values of each instance (Positive on the left, Negatives on the right).

We have found that the value of the points mattered but it was not enough: any instance, be it an epilepsy spike or normal behavior, could reach at one point a high or low enough point in gradual steps, which led us to another variable we decided to compute: the degree of the slope: it meant we would know if and when any signal would go out of its way to an unusual behavior (such as an epilepsy spike).

As such, we have computed the slope of the signal with a custom step. We have found that 5 was the perfect step to consider because a higher step could mean going from one point to another and ignoring a whole spike that could be of utmost importance in the detection. In comparison, a lower step would considerably increase computation time: we had to find the perfect tradeoff to make.

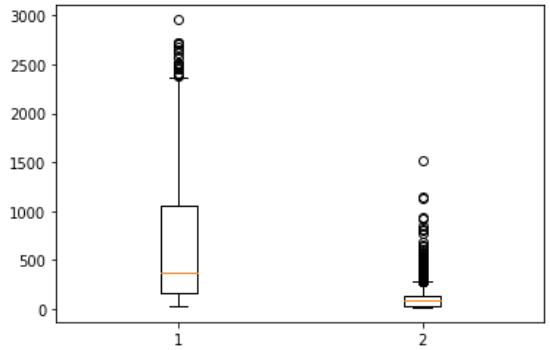

Figure 3.: Boxplot of the maximum slope computed. (Positive instances on the left, Negatives on the right).

Since epilepsy spikes can happen very quickly and imitate a normal behavior for some time afterward, instead of computing the mean of the slopes found, we decided to keep the one with the highest slope for each instance instead of the average slope, resulting in rather scattered values when displaying the difference between positive and negative instances seen on *Figure 3*.
However, we have also pursued some methods that were not as efficient as we thought they would be. One we would like to highlight here is the computation of the intersection of signals. Each data series, with five signals each, is often intertwined together. We thought that epilepsy spikes, being very chaotic signals, could mean they overlap each other more often than normal brain behavior. By implementing a sweep line algorithm to detect when lines intertwine (see simplified pseudocode below in Box 1.) and thus counting the number of intersections of each instance: we expected a large difference between the epilepsy spike and normal behavior.
However, not only was the algorithm very slow, but the discriminative aspect of the method was also not sufficient for our dataset: other faster methods could

replicate better results than said algorithm, so we took the decision not to use it.

```
S0 = [0, 0, 2, 0]
S1 = [1, 1, 1, 1]
intersections = 0
highest = S1 //because S1[0] > S0[0]
for i in range(0, length(S1)):
   if the highest signal at position i changes from i-1:
      highest = new_highest
            //here S0 at i=2
      intersection += 1
```

Box 1.: For S0 and S1, there are two intersections, one at i = 2 and another at i = 3 (considering the array starts at 0), because the value of S0 becomes higher than S1 and then returns to the former order afterward.

With enough variables to work with, we still had to choose the classification method which fits best the problem we have to find a solution for.

4. Classification

Classification is a method used in supervised learning, which aims to output a discrete variable, such as a class or a label. Therefore, the model is trained in such a way to render a discrete output (y) for some input data (x).
Below, the different classification algorithms used throughout our experiment.

· *Decision trees*

A decision tree is a predictive tool used for classification models: the decision variable is categorical. The trees are constructed by an algorithm that splits the dataset in different ways based on certain conditions. Each node represents a predictor value in such a tree, meaning a feature. Each node is linked by a branch that represents a decision. At the end of the tree, each leaf node represents the predicted outcome.
In our case, this prediction technique gave the lowest result, for any given set of features. Even though decision trees are one of the simplest and most interpretable classifiers available, they are known to be unstable learning algorithms: with a slight change in input, the output changes and gives rise to a completely different tree structure.

· *Random Forest*

Random forest is a supervised learning algorithm that creates a set of randomized decision trees on data samples. A voting takes place for every outcome obtained, and the most voted prediction result is selected as the final solution. Since it involves multiple trees producing different values, it reduces the overfitting of the algorithm by averaging the results (one of the Decision Tree method's biggest drawback).
Random forest is known to be a highly effective algorithm that can outperform neural networks in some cases.

Overall, for any given set of features, Random Forest predictions were higher in comparison with Decision Tree and Logistic regression. In our case, the best results were given with a set off 24 trees at 92% accuracy.

· *Logistic Regression*

Logistic Regression is an algorithm which can be used for regression as well as classification tasks, but it is widely used for classification tasks. The response variable that is binary belongs either to one of the classes. Since we had to classify the data between two categories, Logistic Regression was the first technique that we tried. In our case, the highest accuracy from this technique reached 90%.

· *Support Vector Machine*

Classification algorithms such as Logistic Regression needs a manual feature extraction step to classify data with non-linear boundary: this is a time-consuming process in the software development pipeline. In Support Vector Machine it is possible to classify a data set without this step. We can classify data using non-linear boundaries without the manual feature engineering step.
Support Vector Machine has been used widely for the classification of electroencephalography signals for the diagnosis of neurological disorders such as epilepsy and sleep disorders. Support Vector Machine shows good generalization performance for high-dimensional data due to its convex optimization problem. In our case, the highest accuracy for any given set of features was 91%.

· *Gradient boosting trees*

In gradient boosting, an ensemble of weak learners (usually decision trees) is used to improve

the performance of a machine learning model. Combined, their output results in a better model. With classification, the final result can be computed as the class with the majority of votes from weak learners. This method has been the most time-consuming method, at around 100 iterations but has been the most efficient one, with an accuracy of 94.416% for an average of 55 seconds of model training time.

· *Kmeans*

Kmeans is a non-supervised clustering technique. It computes centroids and repeats until the optimal centroid is found. In this method, data points are assigned to clusters in such a way that the sum of the squared distances between the data points and the centroid is as small as possible.

The performance of the K-means clustering algorithm depends upon the highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters.

In our case, we have tried the method on two different cases, one for classifying the Daubechies coefficients and for classifying the other features we had. Although it didn't help us reach a higher accuracy, it helped us to better understand the problems we faced on the project.

· *BIRCH*

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is used in complement to other classic clustering algorithms such as K-means. Indeed, K-means clustering does not perform clustering very efficiently and it is difficult to process large datasets with a limited amount of resources (i.e., memory or a slower CPU).
After our failed attempt to make k-means work, our next step was to use BIRCH. However, it did not give us satisfactory accuracy.

5. Experiments

For this project, we have worked using five different data sets: four labeled brain activity data series and one unlabeled. The goal of the project was to use the four labeled datasets to build a model capable of predicting the label of each unlabeled data series.

To define more precisely the labeled datasets we had, two of them were positive instances of an epileptic spike, and the other two were negative instances.
Each positively labeled dataset contained 400 data series and each negatively labeled dataset contained 2000 data series, thus creating an environment that almost copied the reality where epilepsy spikes are a minority compared to normal brain behavior. Whatever the label was, each data series were composed of five signals of 768 data points each.
All of our experiments were done using Jupyter Notebook, hosted by Google on the Google Colab Notebooks platform.
Google Colab is a free-to-use cloud platform for running code snippets. It does not use the user's hardware, and instead uses the hoster's hardware, meaning that we have all virtually worked in the same environment during the project. Google Colab's hardware is composed of an Intel Xeon CPU at 2.20GHz and an NVIDIA Tesla K80 GPU.
The various models we developed have been done using Python, and more specifically, with the PySpark Framework (version 2.4.4).
Other frameworks and built-in modules have been used in this project: NumPy was the most useful, it has been used thoroughly for handling the large data set we had and for faster numerical computations. We have also used Matplotlib in order to display in a more readable way the signals than with raw numbers, but also in order to build data visualization graphs with boxplots or scatterplots when we were working with K-means classification for example.
We have also used PyWavelets (pywt in shorts) for working with Discrete Wavelet Transform with ease, allowing us to compute Daubechies coefficients in just a few lines instead of implementing a whole algorithm for it.
Our final and best experiment, which reached a 94.416% accuracy, has been done using the mean of absolute values of the signal, the minimum and maximum value of the first signal, the signal bounds, the larger distance at a given time, the biggest slope in the signal, and the first and tenth cA and cD Daubechies coefficients of the five signals.

6. Conclusion

This project was the very first step in thinking and working as Data scientists. We have learned a lot technology-wise and about how to work with data. Some methods for working with data were easy

to implement in the project, some less so. Our hardest task was to sometimes take a few steps back to be able to go further, but also to constantly challenge our initial assumptions that could be very misleading. This is something we have done a lot, but we definitely now know that it was not enough and that we could have reached a higher score if we had challenged each other's assumptions more often. We have lost a lot of time working with completely unusable variables, or obsolete ones, thinking it could be an important aspect of creating the predictions.

To summarize, given our difficulties and the obstacles we encountered, we believe we can be content with our result of 94%. Although, if we had a second go at this project, and used everything we now know, we could most likely reach a higher score than 94%.

References

Schirrmeister, R.T., Springenberg, J.T., Fiederer, L.D.J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W. and Ball, T. (2017), Deep learning with convolutional neural networks for EEG decoding and visualization. Hum. Brain Mapp., 38: 5391-5420. https://doi.org/10.1002/hbm.23730

Huang J-S, Li Y, Chen B-Q, Lin C and Yao B (2020) An Intelligent EEG Classification Methodology Based on Sparse Representation Enhanced Deep Learning Networks. Front. Neurosci. 14:808. doi: 10.3389/fnins.2020.00808

Golmohammadi M, Harati Nejad Torbati AH, Lopez de Diego S, Obeid I and Picone J (2019) Automatic Analysis of EEGs Using Big Data and Hybrid Deep Learning Architectures. Front. Hum. Neurosci. 13:76. doi: 10.3389/fnhum.2019.00076

Umut Orhan, Mahmut Hekim, Mahmut Ozer, EEG signals classification using the K-means clustering and a multilayer perceptron neural network model, Expert Systems with Applications, Volume 38, Issue 10, 2011, Pages 13475-13481, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2011.04.149.

Kumar N, Alam K, Siddiqi A. H. Wavelet Transform for Classification of EEG Signal using SVM and ANN. Biomed Pharmacol J 2017;10(4). Available from: http://biomedpharmajournal.org/?p=18132

A. E. Vijayan, D. Sen and A. P. Sudheer, "EEG-Based Emotion Recognition Using Statistical Measures and Auto-Regressive Modeling," 2015 IEEE International Conference on Computational Intelligence & Communication Technology, 2015, pp. 587-591, doi: 10.1109/CICT.2015.24.

H. Nolan, R. Whelan, R.B. Reilly, FASTER: Fully Automated Statistical Thresholding for EEG artifact Rejection, Journal of Neuroscience Methods, Volume 192, Issue 1, 2010, Pages 152-162, ISSN 0165-0270, https://doi.org/10.1016/j.jneumeth.2010.07.015.

Janir Ramos da Cruz, Vitaly Chicherov, Michael H. Herzog, Patrícia Figueiredo, an automatic pre-processing pipeline for EEG analysis (APP) based on robust statistics, Clinical Neurophysiology, Volume 129, Issue 7, 2018, Pages 1427-1437, ISSN 1388-2457, https://doi.org/10.1016/j.clinph.2018.04.600.

Gregory R. Lee, Ralf Gommers, Filip Wasilewski, Kai Wohlfahrt, Aaron O'Leary (2019). PyWavelets: A Python package for wavelet analysis. Journal of Open-Source Software, 4(36), 1237, https://doi.org/10.21105/joss.01237.

Philippa J. Karoly, Dean R. Freestone, Ray Boston, David B. Grayden, David Himes, Kent Leyde, Udaya Seneviratne, Samuel Berkovic, Terence O'Brien, Mark J. Cook, Interictal spikes and epileptic seizures: their relationship and underlying rhythmicity, Brain, Volume 139, Issue 4, April 2016, Pages 1066–1078, https://doi.org/10.1093/brain/aww019