Wavelet course, MVA 2014-2015

Simon Bussy, simon.bussy@gmail.com **Antoine Recanati, arecanat@ens-cachan.fr**

Dreem Challenge report (team Bussanati)

1 Description and specifics of the challenge

We worked on the challenge proposed by the start-up *Dreem*. The data consists of epochs of 3 seconds of EEG signals, sampled at 200 Hz during a normal night on 20 different subjects. These epochs are labeled as 0 or 1. A label 0 means that no Slow Oscillation has been measured during the 0.5 seconds following the 3 seconds epoch. A label 1 means that a slow oscillation occurred during the 0.5 seconds following the 3 second epoch. Figure 1 shows plots of a few of these time series. An efficient detection of slow oscillations

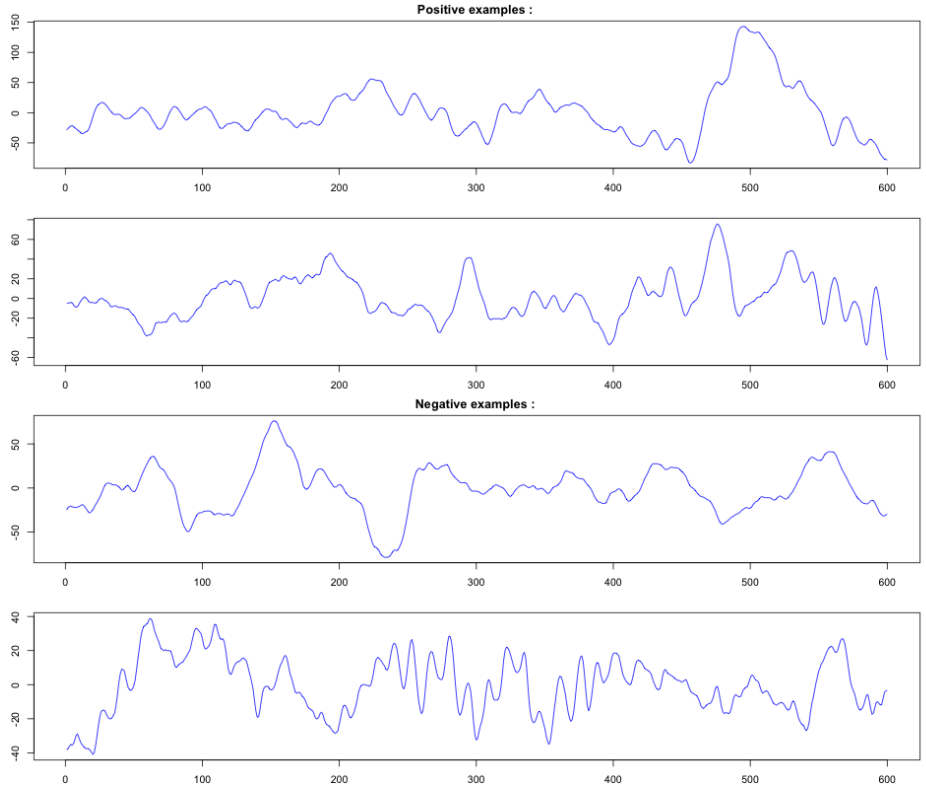


Figure 1: example of time series

would enable the start-up product to help the user switching stages of sleep by stimulating the appearance of slow oscillation.

One specificity of the dataset is that it is skewed : there is only 7% of positives. Therefore, the metric used to score the results is the AUC metric, that is the area under the ROC curve. Another interesting point

is that the data contains the IDs of the subject, allowing us to fit separately the data for each subject. In figure 2 is shown the proportion of data that comes from each subject. The different subjects happen to have different behaviors : there are substantial discrepancies in the positive rates according to the subjects, as one can see from 3.

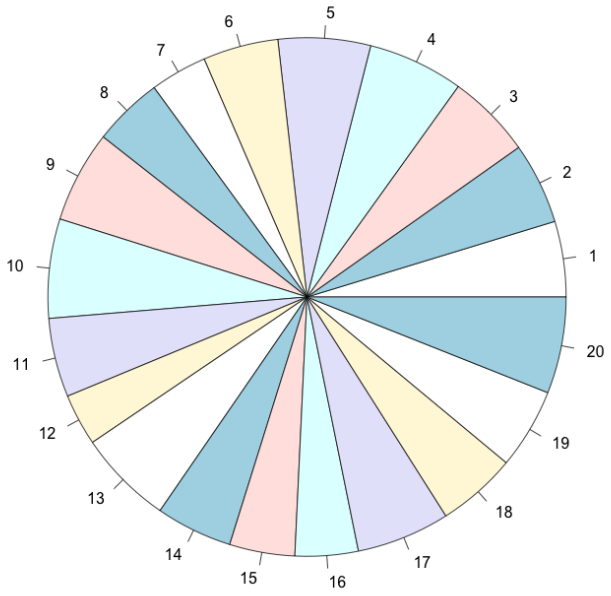


Figure 2: proportion of each subject in the training set

A first look at the data does not provide any obvious "trick" to help classifying the data. For example, We plotted boxplots of the means and ranges (5) of positives and negatives classes, and in figure 4 are plotted Fourier Transform amplitude coefficients for some positives and negatives examples. We have plotted the boxplots for each of the 20 subjects, but this does not give any useful information either.

We also tried to implement directly some classifiers (logistic regression, SVM) on the raw data, but these yielded to poor AUC scores. Scattering transforms build invariant, non-linear and little redundant representations of the signal, which can be an accurate to classify signals such as EEG ([1],[2]). "Naive" neural networks can lead to similar results, although they usually require more computation and are more like a black box.

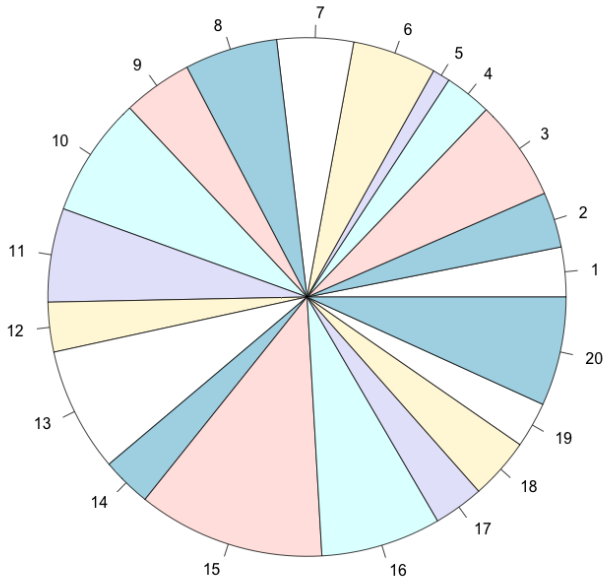


Figure 3: positive rate for each subject in the training set. each slice's thickness is proportional to the positive rate of the subject

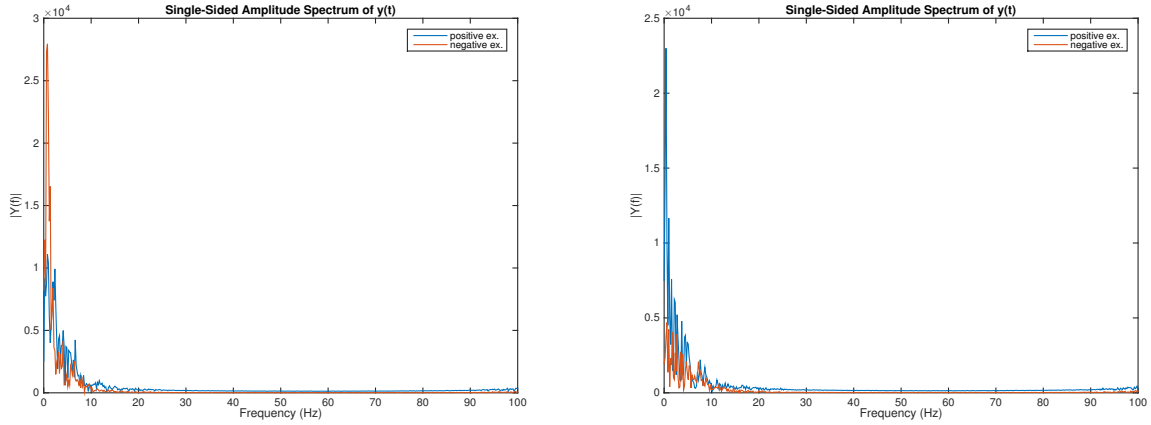
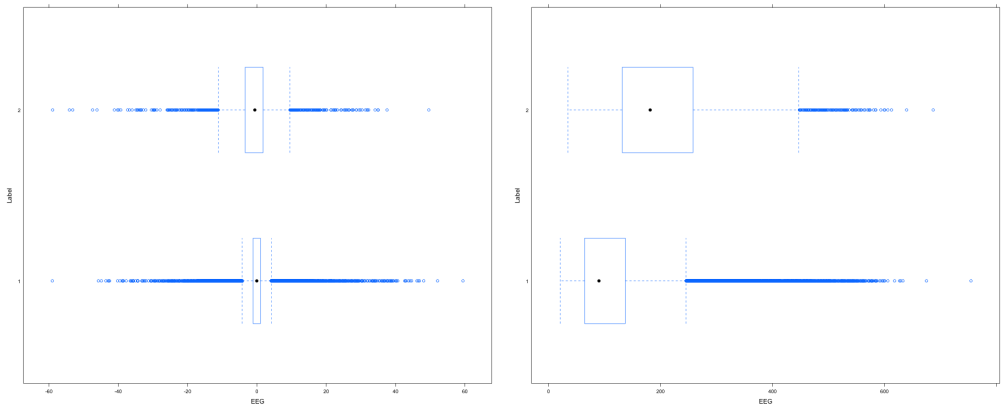


Figure 4: FFT coefficients for four examples of signals (1 positives and 1 negatives per subplot)



(a) means (b) ranges

Figure 5: boxplots for positive (height 2) and negative (height 1) classes

2 Approaches and results

Having not found any relevant feature on the data enabling us to design an algorithm specific to this problem, we tackled the challenge this way : we first computed scattering transforms of the signal. Then, we tried several algorithms to classify the data, keeping in mind that the data set is skewed and that we want to maximize the AUC score. Finally, we tried to train a classifier with a neural network, and to use the knowledge on the subjects IDs. An additional challenge we took was to classify the data in a reasonable time on our laptops.

2.1 Computation of the scattering transforms

As one can see from the Fourier Transforms (figure 4), most of the information of the signal is contained below 20 Hz. We want to take a sliding window of at least $0.05s$, that is to say of at least 10 points, which makes the next power of two being $T = 2^4$ points. We tried several values of T : $2^4, 2^5, 2^6$, and we also chosen the quality factors Q by cross-validation, for each order. We let the function `T_to_J` select the scales j_1 and j_2 (we also tried to use the computations at the third order but it did not change the score). Assuming the signal was translation-invariant, we used the built-in `wavelet_factory_1d`. The optimal values we found (with the classifiers used afterwards) were : $T = 2^5$, $Q_1 = Q_2 = 1$, $j_1 = j_2 = 5$. For these values, we show a visualization of the scattering coefficients in figure 6.

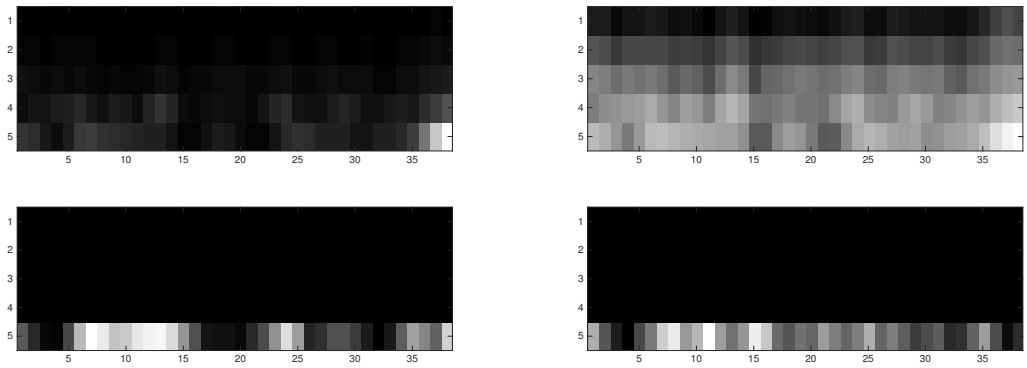


Figure 6: scattering coefficients with (right) and without (left) normalization+log on the mean of the negatives

2.1.1 heuristics

We tried to improve the score with a few *ad-hoc* techniques. Following [2], we tried to threshold the scattering coefficients in order to reduce the impact of the noise. We also tried to use only the second order coefficients, in case the difference between positives and negatives would be contained in those rather than in the first order coefficients. Finally, we tested whether renormalizing the scattering coefficients and computing

their logarithm improved the classification score. None of these heuristics systematically and substantially improved the score, so we kept the scattering coefficients in their original form.

2.1.2 Naive implementation

Our first try was to use a "usual" classifier on these features vectors, *e.g.* logistic regression. This yielded to an AUC around 0.74. A linear SVM did not score significantly better, and a radial basis function SVM took a very long time to run, andTo improve the score, we tried to tackle the fact that the data set was skewed by using algorithms specifically designed to maximize the AUC.

2.2 Classification algorithms to maximize the AUC score

2.2.1 approximated AUC gradient-descent

We implemented the gradient-descent method described by Herschtal and Raskutti in [3]. The main lines of the paper are the following. 1. We try to optimize a linear classifier. 2. The empirical AUC can be written as :

$$\hat{AUC}(\vec{\beta}) = \frac{1}{PQ} \sum_{j=1}^{P} \sum_{k=1}^{Q} g(\vec{\beta}.(x_j^+ - x_k^-)) \tag{1}$$

where $\vec{\beta}$ describes the linear classifier, x_j^+ are the features vectors of the positive examples (there are P of them), x_k^- are the negative ones, and g is the Heaviside function. 3. Replacing the Heaviside function by the "sigmoid" function makes it differentiable and does not change its value much if $\|\vec{\beta}\|$ is large enough, yielding the rank statistic :

$$R(\vec{\beta}) = \frac{1}{PQ} \sum_{j=1}^{P} \sum_{k=1}^{Q} s(\vec{\beta}.(x_j^+ - x_k^-)) \qquad , \quad s(x) = \frac{1}{1+e^{-x}} \tag{2}$$

4. An unbiased estimator of the rank statistic, with a variance that is reasonably low, is given by the following expression, making a huge computational (and memory) gain :

$$R_l(\vec{\beta}) = \frac{1}{Q} \sum_{k=1}^{Q} s(\vec{\beta}.(x_{k mod P}^+ - x_k^-)) \tag{3}$$

5. Starting from $\vec{\beta}$ with a small norm and increasing it until it is large enables us not to fall into local maxima and end up with a rank statistic close to the true empirical AUC.

We implemented this algorithm, initializing $\vec{\beta}$ with a linear regression, and the results were quite good (0.87).

2.2.2 Probabilistic Principal Component Analysis (PPCA)

Another algorithm that seemed suited to unbalanced data sets is the mixture of probabilistic components analysis, proposed by Tipping and Bishop in [4]. This model is a generative model providing a posterior probability distribution of the data given the reduced representation, considered as a latent variable. In a

nutshell, like PCA, it is accurate to represent data which lies in a low-dimensional manifold, but the latent variable and probability distribution framework enables us to build a mixture model, and more importantly here, to train a model on the positives and one of the negatives, and compute the probability to belong to each of these classes (as a sum of likelihoods). Here, the fact that the data set is skewed does not change those probabilities, except that there are more examples to fit the negative model than the positive one.
We implemented this algorithm, using mixture models and tuning the parameters (number of mixture component and dimension of subspace) for each model by cross-validation, but eventually using more than 1 mixture component lead to over-training. For some values, we could reach a score of almost 0.99 on the train set, but less than 0.6 on the test set. The results of these method were good, but not optimal : on a test set created with 20% of the train set, we found an AUC score of 0.82 (and we have no submissions left!).

2.3 Neural networks

We then tried artificial neural networks with backpropagation learning algorithm. We tried different pre-processing (fourier basis, scattering transform) and it turned out that the best practice was to directly plug the original time series data as input of the network. We tried different architectures, including those with many hidden layers known as deep learning models. Many layers allow them to compactly represent highly non-linear and highly-varying functions. So the complexity of the model increases, which often leads to higher risks to overfit, as explained in the third section.

Because neural network learning is in general not convex, the training could be very complicated, as we noticed.

More precisely, the problem of finding optimal parameters of a Neural Network is in general not convex, implying there is no guarantee of global solution. Then curse of dimensionality phenomena arise when analyzing data in high-dimensional spaces and the amount of data needed to support the result often grows exponentially with the dimensionality. So hyper-parameter values selection becomes harder with deep neural networks. Hyper-parameters associated with the neural network structure and the training algorithm need to be selected, some of them are : the initial learning rate, the learning rate schedule, the number of training iterations, number of hidden units, the non-linearity function, the preprocessing steps etc. And there is no clear theoretical solution for values of any of these hyper-parameters for the moment. Thats why we need to make a lot of cross validations in different parts of the parameters space chosen using interpretation and intuition coming from previous results.
The optimal architecture is four layers with 14,13,10,9 neurons, and 400 iterations.

2.4 Use of additional knowledge on the subject IDs

We tried to use the extra information contained in the subjects IDs in two ways :

1. train a classifier for each subject

2. train a classifier with all subjects and then add a subject-dependent term to the final classifier

The first idea did not improve the results because it lead to overfitting, as the number of positives in the training set for a single subject was often too low (order of magnitude : 100 examples, whereas there were around 600 features) to properly train the classifier.
The second idea was to use all the training set to have a robust classifier, and then compare the predicted positive rates of those classifiers to the effective ones, and artificially make the negatives that are the closest to being classified as positive, positives, so as to have the same rates. This gave a very slight increase in the performance.

3 Conclusions

To test these algorithms, we have separated our train set into a sub training set (80%) and a test set (20%). We thought the AUC score was the same with likelihood or with labels (0 and 1), but it actually is not, so we realized very late before the dead line that the gradient-descent algorithm and PPCA were good. Until then, our best scores had been obtained with the neural network. Now, we don't have enough time nor submissions to try to combine the results from different algorithms, but we had fun doing this project anyway !

References

[1] *Wavelets in Neurosciences*, Hramov, A.E., Koronovskii, A.A., Makarov, V.A., Pavlov, A.N., Sitnikova, E., **Springer**

[2] *Wavelet based EEG denoising for automatic sleep stage classification*, Edson Estrada, Homer Nazeran, Gustavo Sierra, Farideh Ebrahimi, S. Kamaledin Setarehdan

[3] *Optimising Area Under the ROC Curve Using Gradient Descent*, Alan Herschtal, Bhavani Raskutti

[4] *Mixtures of Probabilistic Principal Component Analysers*, Michael E. Tipping and Christopher M. Bishop, Neural Computation 11(2), pp 443482. MIT Press.