Automatic Sleep Stages Classification Using EEG Entropy Features and Unsupervised Pattern Analysis Techniques

Sasha Collin
collin.sasha@gmail.com

Clement Nguyen
clement.ngn@gmail.com

Abstract

Sleep stage classification is a useful tool to better understand the sleep process. However, it is usually done manually. In this project, we implemented from scratch an automatic sleep classification method based on unsupervised entropy feature classification as proposed in [3], and compared its performances to the ones obtained from custom features, or obtained from a Convolutional Dictionary Learning method for classification.

1. Introduction

Sleep has a very important biological role and has been related to plasticity and reorganization of memory. Unfortunately, sleep disorders are a very common issue that affect a significant number of people worldwide, and have a serious impact on people's health. That's why trying to better understand how sleep works and trying to detect and characterize sleep stages is a hot research topic.

There are 5 different sleep stages: wake (W), drowsiness (N1), light sleep (N2), deep sleep (N3), and rapid eye movement sleep (REM). The classification of such stages, when done manually, is difficult, time consuming, and can lead to annotation errors. To tackle this issue, the authors of [3] propose an unsupervised classification scheme for sleep stage prediction, using entropy features computed from electroencephalograms (EEGs), features relevance analysis, and unsupervised clustering.

We implemented from scratch this algorithm and, to assess the relevance of entropy features, we designed custom features and compared the performances of the algorithm on both sets of features. Finally, we implemented a semi-supervised classification method based on Convolutional Dictionary Learning and compared the classification performances of both algorithms.

2. Methods

In this section, we first present the algorithm described in [3] to classify sleep stages in an unsupervised way. This algorithm follows these steps, which are all detailed in the next subsections:

1. Processing of the raw EEG channels (input)
2. Extraction of entropy features
3. Projection and selection of features($Q-\alpha$ algorithm)
4. Unsupervised classification (J-means clustering)
5. Prediction of sleep stages

We then describe the computation of custom features, and the Convolutional Dictionary Learning method for classification.

2.1. Data processing

The EEG signals used were taken from the SC Sleep-EDF Database [Expanded] [2]. We used the channels Fpz-Cz and Pz-Oz of the EEG signals, sampled at $100Hz$. Signals were first cropped to reduce the number of sleep stage 1 (awake). Then, signals were split into epochs of 30 seconds. Sleep stages 3 and 4 were merged in a single stage of "deep sleep". In the following, a **sample** refers to the processed signal corresponding to a single night of a participant.

2.2. Features extraction

Given an input discrete time series $x = \{x_1, x_2, ..., x_N\}$ of length N, which corresponds to one epoch of a channel of a given sample, we computed the following entropy features:

Fractal dimension. Fractal dimension is a measure of the signal complexity and its scale invariance. The fractal dimension corresponds to the slope of the straight line fitting the sequence of points $(\ln(L), (S(L)/L))$, where L is the size of a box and $S(L)$ the number of boxes, which can be computed as followed: $S(L = n\Delta t) = \sum_{i=1}^{\text{mod } N/n} |\max(\Delta x_i) - \min(\Delta x_i)|$. In this report, **FD** denotes the fractal dimension feature. We also computed two variants of this feature: the Higuchi FD and the Katz FD.

Detrended fluctuation analysis (DFA). DFA is a measure of the time series long-range self-correlations. The cumulative profile X of the normalized time series is first computed as: $X_t = \sum_{i=1}^{t}(x_i - \bar{x}_i)$. X is divided into N/L boxes of length L and, for each box, a straight line Y is fitted by minimising the squared errors within each box. The fluctuation is defined as:

$$f(L) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(X_k - Y_k)^2}.$$

$f(L)$ is computed for several values of L and the DFA is defined as the slope of the linear fitting between $(\log(L), \log(f(L)))$. We computed the DFA for $4 \leq L \leq 16$ and $16 \leq L \leq 64$ and we refer respectively to these features as **DFA-$\mathbf{a_1}$** and **DFA-$\mathbf{a_2}$**.

Shannon entropy. It is a measure of the data spread, and was computed using the following formula: $H(x) = -\sum_i p(x_i)\log(p(x_i))$, where $p(x_i)$ denotes the probability of x_i. In practice, we partitioned the values of the x_i in $n = 100$ bins and computed the entropy from the resulting histogram. The Shannon entropy is denoted by **H**.

Approximate entropy. The approximate entropy measures the regularity of the time series based on the existence of repetitive patterns. Let m be the length of the pattern and r the error threshold. Subsequences $X_1, X_2, ..., X_{N-m+1}$ of length m are extracted from x and for each subsequence X_j, we count the number of subsequences $C_j^m(r)$ differing from X_j up to r: $C_j^m(r) =$ (number of X_i such that $||X_i - X_j||_\infty < r$). The approximate entropy is defined by:

$$ApEn(m, r) = \Phi^m(r) - \Phi^{m+1}(r)$$

where $\Phi^m(r) = (N - m + 1)^{-1}\sum_{i=1}^{N-m+1}\log(C_i^m(r))$. We computed the approximate entropy for $m = 1, 2$ and r taken as 0.1 times the variance of the time series. We call these features **ApE-m1** and **ApE-m2**.

Sample entropy. Sample entropy solves the bias due to counting subsequence self-matches. $C_j^m(r)$ is modified as follows: $C_j^m(r) =$ (number of X_j, $j \neq i$ such that $||X_i - X_j||_\infty < r$). The sample entropy is defined by:

$$SampEn(m, r) = \log(\frac{\Phi^m(r)}{\Phi^{m+1}(r)})$$

where $\Phi^m(r) = \sum_{i=1}^{N-m+1} C_i^m(r)$. Similarly, we computed the sample entropy for $m = 1, 2$ and r taken as 0.1 times the variance of the time series, and we call these features **SampEn-m1** and **SampEn-m2**.

Multiscale entropy. The multiscale entropy extend the sample entropy to higher time scales. Let τ denote the time scale. The multiscale entropy is obtained by computing the sample entropy on the coarse-grained time series $y_n = \sum_{i=(n-1)\tau+1}^{n\tau}$. We computed the multiscale entropy for $\tau = 2, 3, 4, 5, 6$, resulting in the features **MSE-tau{2-6}**.

We computed the features above on the Fpz-Cz and Pz-Oz EEG signal channels and we use the respective suffixes **-Fpz** and **-Pz**.

2.3. Feature Relevance Analysis

To project the computed features and select the best ones, we implemented the $Q - \alpha$ algorithm, which is described in 1. We also implemented a similar algorithm where the principal components are used to assess the importance of the original features. We then keep only the most important ones to perform the classification.

Algorithm 1 $Q - \alpha$ algorithm as described in [5]

Input: $n \times p$ matrix X (EEG features)
$M \leftarrow (X^T X)^2$
$\alpha \leftarrow$ eigenvector corresponding the the largest eigenvalue of M
$W \leftarrow \text{diag}(\alpha)$
$\tilde{X} \leftarrow XW$: weight original data
$V \leftarrow$ PCA of $\tilde{X}$
$\tilde{V} \leftarrow V[:, n_{rf}, :]$: only keep the n_{rf} first principal components explaining 98% of the features variance
$Y \leftarrow \tilde{X}\tilde{V}$: projection of the data
Output: New set of features Y

2.4. J-Means clustering algorithm

As done in [3] we used a J-means clustering algorithm [1] for unsupervised sleep stages classification. J-means is an heuristic that searches an improved clustering solution by moving a cluster center to an "unoccupied" element and re-compute the clusters according to the minimum sum-of-squares clustering. The steps of J-means algorithm can be described as follows:

- *Initialization.* An initial partition is computed with a standard clustering algorithm, e.g. a K-means algorithm.
- *Find unoccupied elements.* A point is considered as an unoccupied element if its distance to the centroid of its cluster is higher than a tolerance threshold. A typical value for this threshold is the standard deviation of the distances to the centroid within the cluster.
- *Jump to the neighbors.* Given an unoccupied element x and a cluster centroid $\bar{x}$, a jump to the neighbors is done by relocating $\bar{x}$ at x and update the partition. Find

the unoccupied element and the cluster centroid giving the best updated partition associated to the jump.

- *Repeat or end.* If the updated partition doesn't improve the clustering, stop and return the previous partition. Otherwise, refine the updated partition with K-means and repeat the process with this new partition, beginning at step 2.

Instead of initializing with K-means in step 1, [4] proposed to initialize the centroids randomly. We investigate both strategies in the results section.

2.5. Custom features

We also computed custom features to compare the classification performances of our algorithms using these features and the ones detailed previously. To do so, we reused the work done in the practical session about features selection (TP2). We computed:

- *distribution features:* mean, standard deviation, min, max, kurtosis, skew, 25%, 50% and 75% percentiles
- *auto-correlation features:* auto-correlation values up to a maximum lag of 500, maximum lag in *Hz* and maximum auto-correlation
- *spectral features:* sum of square Fourier coefficients over frequency bins ($0 - 50$ Hz, $n_{bins} = 100$)

As the number of features was very high (1206), we implemented a first feature selection by discarding the features that had a variance below 0.2. This enabled us to reduce the number of features to roughly 200 per sample.

2.6. Convolutional Dictionary Learning (CDL)

Finally, we implemented a last method of classification using convolutional dictionary learning, by adapting the work done during the first practical session about dictionary learning (TP1). For each sleep stage and each channel, we selected a period of 1 minute (i.e. two epochs) and learnt a dictionary of atoms by solving the following task:

$$\min_{(\mathbf{d}_k)_k,(\mathbf{z}_k)_k \|\mathbf{d}_k\|^2 \leq 1} \quad \left\| \mathbf{x} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{d}_k \right\|^2 \quad + \quad \lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_1$$

where $\mathbf{d}_k \in \mathbb{R}^L$ are the K dictionary atoms (patterns), $\mathbf{z}_k \in \mathbb{R}^{N-L+1}$ are activation signals, and $\lambda > 0$ is the sparsity constraint. We used 8 atoms with a length of 200 each, and a penalty of 4. Then, using the learnt atom dictionaries, we tried to reconstruct each epoch of 30 seconds. For each reconstruction, we computed the MSE for both channels and summed them. The final predicted label for a given epoch was the one corresponding to the lowest MSE.

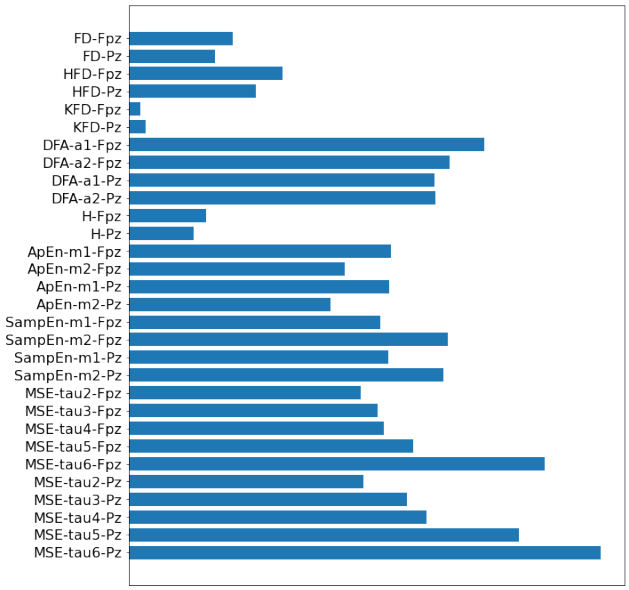

Figure 1. Feature relevance defined by the contribution to the principal components.

Remark: contrary to the previous methods, this one is not completely unsupervised, but the idea behind is to show that with only one minute labeled for each sleep stage, we are able to predict the labeling of the rest of the epochs of the sample. We will also show that the learnt dictionary can also be used for the labeling of new samples.

2.7. Metrics

To evaluate the performances of the different algorithms, we computed the accuracy, the recall and the precision of our predictions.

3. Experiments

3.1. Feature relevance

In our experiments, we looked at the relevance of the features. This relevance is obtained using the $Q - \alpha$ algorithm, but it generates projected features. To assess the relevance and thus directly select original features, we use the vector $\tilde{V}$ obtained in the algorithm $Q - \alpha$, which is composed of the principal components explaining 98% of the features variance. Then, for each original feature, we sum its absolute contribution to each of the principal components, which gives its overall importance. Finally, we normalize the obtained vector of importance weights.

3.2. Sleep stage clustering results

Sleep stages clustering was performed on each sample independently, for 40 samples. Three sets of features derived from the entropy features described in Section 2.2 are used: every features, the 8 most relevant features according

	Accuracy	Avg precision	Avg recall
All features	0.571	0.558	0.557
Rel. features	0.541	0.537	0.538
Proj. features	0.553	0.534	0.536

Table 1. Sleep stage J-means clustering results with entropy features. The mean of the accuracy, the average precision and the average recall is taken over 40 samples. Three sets of features are compared: all the features, 8 most relevant features and projected features obtained by $Q - \alpha$ algorithm.

Sleep stage	Precision	Recall
W	0.854	0.604
N1	0.116	0.157
N2	0.719	0.504
N3	0.607	0.802
REM	0.378	0.615

Table 2. Per-stage J-means clustering results with projected entropy features. The mean of the accuracy, the average precision and the average recall is taken over 40 samples.

to the previous experiment, and projected features obtained by the $Q - \alpha$ algorithm described in Section 2.3. Table 1 shows the mean taken over the 40 subjects of the accuracy, the average precision and the average recall obtained after J-means clustering. Keeping all the features gave the best results as all the information is preserved. No noticeable performance gap is observed using the two lower dimension sets of features, i.e. the relevant and the projected features.

Table 2 shows the mean taken over the 40 samples of the precision and the recall for each sleep stage using the projected features only and J-means as the clustering algorithm. The results highly vary from a sleep stage to another, meaning that some sleep stages are harder to detect with the clustering algorithm. For instance, the method performs poorly at classifying sleep stage N1. [3] also reported poor results on the classification of stage N1. They explain this by the known similarity of the EEG activity between stages N1 and REM. Indeed, as the confusion matrix shows stage N1 is misclassified with REM, but also with stage N2. These performance gaps may also be a consequence of the class imbalance in the dataset. Notably, epochs corresponding to sleep stage N1 are far less common than with other sleep stages.

3.3. Comparison of clustering algorithms

In this experiment, we compare several clustering strategies: J-means algorithm initialized with K-means as described in Section 2.4, J-means with a random initialization as proposed by [4], and K-means algorithm. Table 3 shows the mean taken over 40 samples of the accuracy, the average precision and the average recall using the projected entropy features, for the different clustering strategies. Overall, J-means approaches yielded slightly better results than the standard K-means algorithm. Randomly initializing J-means algorithm can be useful to find better local optimum since it was the best performing approach.

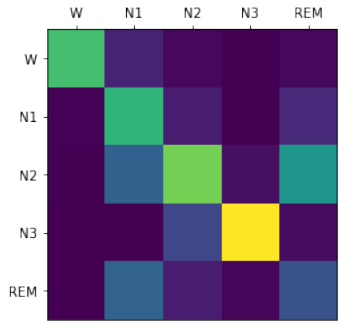

Figure 2. Confusion matrix obtained with the label assignment by J-means clustering.

	Accuracy	Avg precision	Avg recall
J-means (K-means init)	0.554	0.534	0.536
J-means (random init)	0.555	0.543	0.542
K-means	0.553	0.534	0.536

Table 3. Sleep stage clustering results with projected entropy features. The mean of the accuracy, the average precision and the average recall is taken over 40 samples. Three clustering strategies are compared: J-means initialized with K-means, J-means with a random initialization, and K-means algorithm.

3.4. Comparison with custom features

Results obtained with the custom features (Tab 4) outperformed by a large margin (e.g. 10% in accuracy) the ones obtained with entropy features. While being lower than the best ones obtained by the authors of [3], they compete with the performances they obtained using entropy features and PCA (average accuracy of 70% against 67% for us), which seems very promising.

	Accuracy	Avg precision	Avg recall
All features	0.669	0.645	0.645
Rel. features	0.658	0.621	0.627
Proj. features	0.670	0.638	0.654

Table 4. Sleep stage k-means clustering results with custom features. The mean of the accuracy, the average precision and the average recall is taken over 40 samples. Three sets of features are compared: all the features, most relevant features and projected features obtained by $Q - \alpha$ algorithm.

3.5. Comparison with CDL classification scheme

The performances of the different algorithms for one sample are shown on Tab 5. The sample was also used to compute the atom dictionary of CDL method. The CDL algorithm achieves an accuracy of 0.475, which is below the accuracy of 0.728 and 0.688 respectively achieved with the entropy and custom features, but the results are promising, especially when we compared them with a random labeling achieving an accuracy of 0.20. This shows that with very few labeled data (less that 0.5% in average), we are able to retrieve nearly half of the labels overall.

Then, we decided to apply the CDL algorithm to a new sample while keeping the same atom dictionary. Results are shown on Tab 6. The accuracy drops for CDL by 8%, but we are still able to correctly predict nearly 40% of the labels.

The results obtained with CDL are very encouraging. To improve them, more time should be allocated to the tuning of the parameters of this method, especially the number of atoms learnt, the length of such atoms and the sparsity constraint λ.

	Accuracy	Avg precision	Avg recall
Entropy features	0.688	0.7026	0.6826
Custom features	0.7065	0.7405	0.7112
CDL	0.4750	0.5489	0.4235

Table 5. Performances for a single sample, using entropy features, custom features, or CVL method. One minute of each sleep stage of the sample was used to learn the atom dictionary.

	Accuracy	Avg precision	Avg recall
Entropy features	0.633	0.662	0.596
Custom features	0.728	0.709	0.736
CDL	0.3955	0.5489	0.4844

Table 6. Performances for a single sample, using entropy features, custom features, or CVL method. The atom dictionary was previous learnt from **another sample**.

4. Conclusion

In conclusion, we have shown that EEG features are a very powerful tool to understand sleep dynamics. Using the algorithm and features described in [3], we obtained promising results for sleep stage classification. However these results were outperformed while using custom features. This can be explained by the fact that the authors of [3] did not share how they preprocessed the inputted EEGs (did they try to denoise the signal?), which can drastically impact the values of the entropy features, and thus the performances. Also, many hyperparameters are necessary for the computation of the entropy features, and their values were not shared too.

In any case, custom features seems to be very promising, all the more as they are much quicker to compute than entropy features. It would be interesting to combine them with entropy features to see if it increases the overall performances. The CDL algorithm also delivers good results, even for samples on which the atom dictionary was not trained. Thus, it could be interesting to use the predictions of the CDL on new samples as an additional custom features.

5. Code sharing

All our code, including a jupyter notebook explaining how to obtain our results, is available at the following address `https://github.com/cosasha97/automatic-sleep-stages-classification.git`. We coded all the methods presented in the original paper [3] from scratch. For CDL and custom features extraction parts, we adapted the code used during the practical sessions with Charles Truong.

6. Work distribution

Entropy features were coded by **both students**. Clement Nguyen coded the J-means method and was in charge of the overall training and prediction processes using entropy features. Sasha Collin coded the $Q - \alpha$ method, and was in charge of the generation of custom features and of the CDL algorithm.

References

[1] Pierre Hansen and Nenad Mladenović. J-means: a new local search heuristic for minimum sum of squares clustering. *Pattern recognition*, 34(2):405–413, 2001. 2

[2] Bob Kemp, Aeilko H Zwinderman, Bert Tuk, Hilbert AC Kamphuisen, and Josefien JL Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000. 1

[3] Jose Luis Rodríguez-Sotelo, Alejandro Osorio-Forero, Alejandro Jiménez-Rodríguez, David Cuesta-Frau, Eva Cirugeda-Roldán, and Diego Peluffo. Automatic sleep stages classification using eeg entropy features and unsupervised pattern analysis techniques. *Entropy*, 16(12):6573–6589, 2014. 1, 2, 4, 5

[4] José Luis Rodríguez-Sotelo, Alejandro Osorio-Forero, Alejandro Jiménez-Rodríguez, F Restrepo-de Mejía, Diego Hernán Peluffo-Ordóñez, and J Serrano. Sleep stages clustering using time and spectral features of eeg signals. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 444–455. Springer, 2017. 3, 4

[5] José Luis Rodríguez-Sotelo, D Peluffo-Ordonez, David Cuesta-Frau, and Germán Castellanos-Domínguez. Unsupervised feature relevance analysis applied to improve ecg heartbeat clustering. *Computer methods and programs in biomedicine*, 108(1):250–261, 2012. 2