KTH ROYAL INSTITUTE
OF TECHNOLOGY

Project overview

Dr. Sara Benouar

Biomedical Signal Processing, HL2028

Agenda

- Tasks of the project (Canvas)
- Dataset
- General steps of the project
- Additional support - Documentation

EEG-based sleep scoring

- The original recommendation presented in the R&K standard was to make use of several biosignals for sleep scoring:
- ***one EEG (*** Electroencephalogram) ***lead, two EOG (*** Electrooculogram***) leads, and an EMG (*** Electromyogram) ***lead***

Non-stationarity

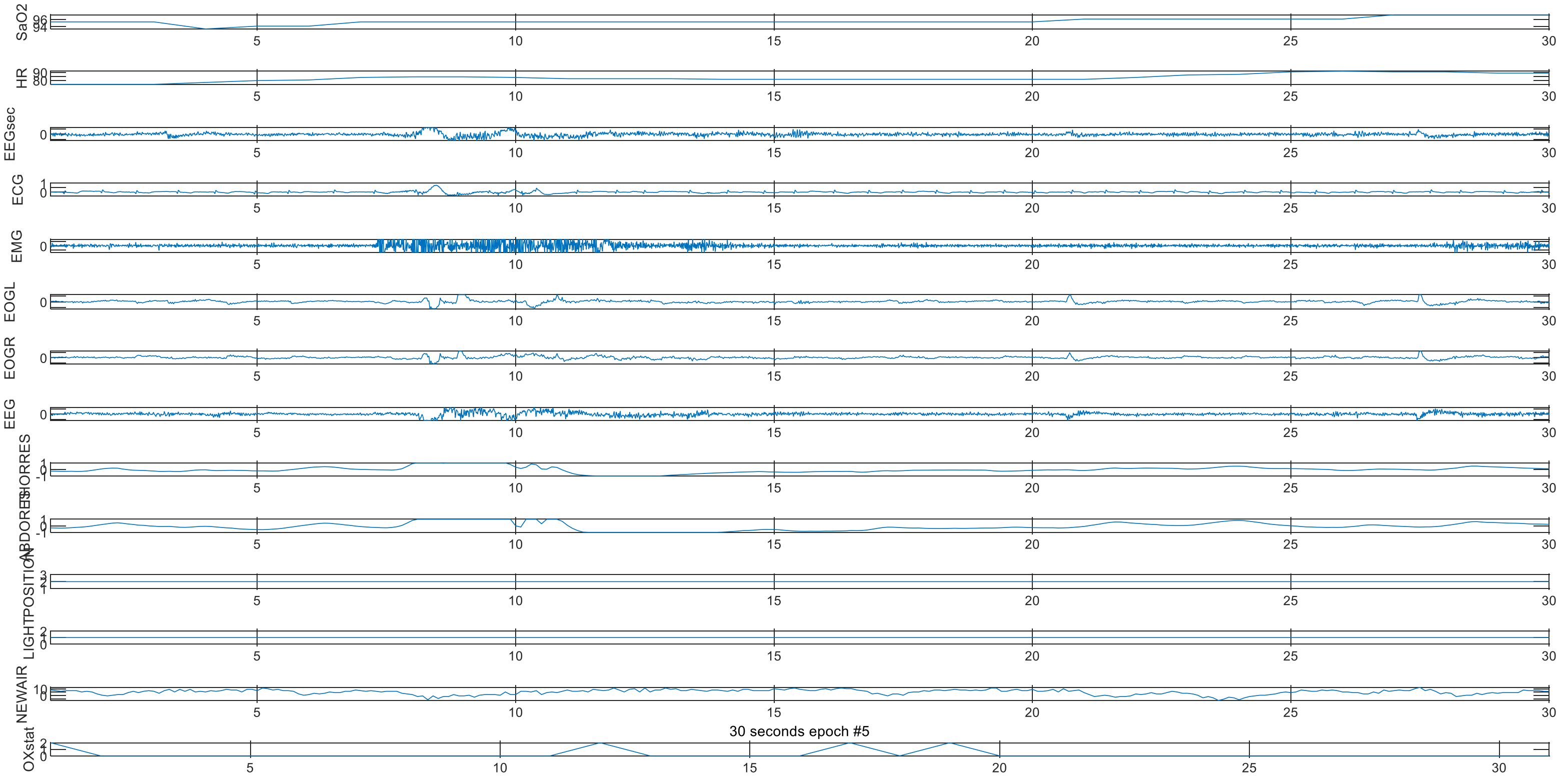
divided into multiple intervals – segments –

approximately stationary

EEG-based sleep scoring

Dataset, Signals

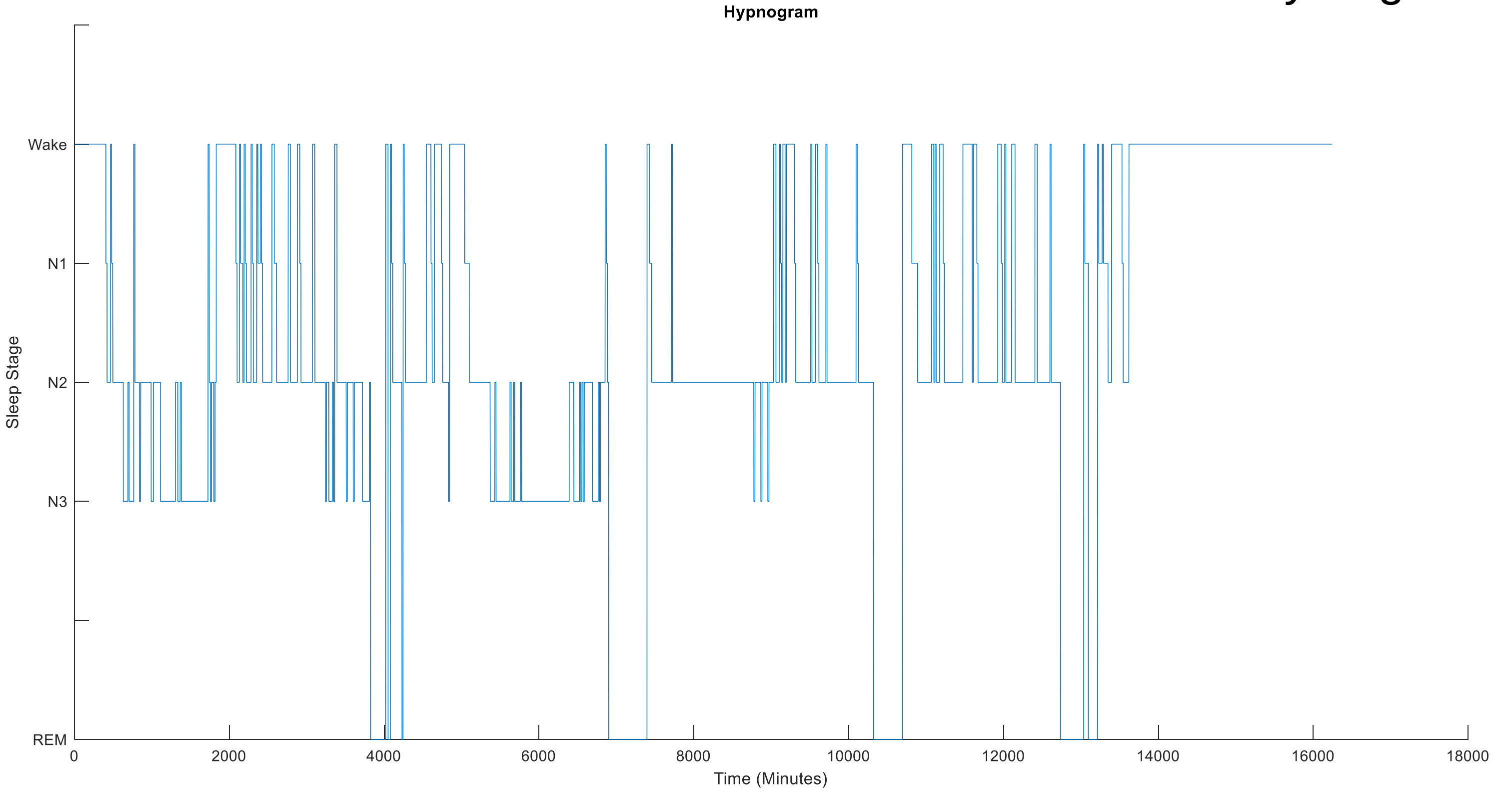
Input is 30sec epoch Matrix



EEG-based sleep scoring

Dataset, Annotations

A binary target Matrix



General steps of the project

1. General Pre-processing

Algorithm filtering According to the signal specifications

2. Specific Pre-processing

30Hz high cut filter (before calculating Hjorth parameters)

3. Temporal features extraction

statistical measures such as **mean**, **variance**, **amplitude**, skewness, kurtosis. For **sleep, scoring** calculate also **Hjorth parameters**.

General steps of the project

Transform the signal to the frequency domain, divide it into 5 - 10 frequency bands, and extract features. Use **Wavelet transform** (better than DFT). Two main features for sleep application are sleep spindles and K complexes.

4. Frequency features extraction

Feature extraction overview

Temporal features extraction

statistical measures such as **mean, variance, amplitude, skewness, kurtosis.** For **sleep**, scoring calculate also **Hjorth parameters.**

Frequency features extraction

transform the signal to the frequency domain, divide it into 5 -10 frequency bands, and extract features. Use **Wavelet transform** (better than DFT). Two main features for sleep application are sleep spindles and K complexes.

General steps of the project

5. Features selection step

use a heuristic approach based on a genetic algorithm. Or use PCA principal component analysis. To reduce the number of features: which means to apply a transformation that maps the high dimensional feature space to a low dimensional one.

6. Choose Unsupervised or supervised

Clustering or classification

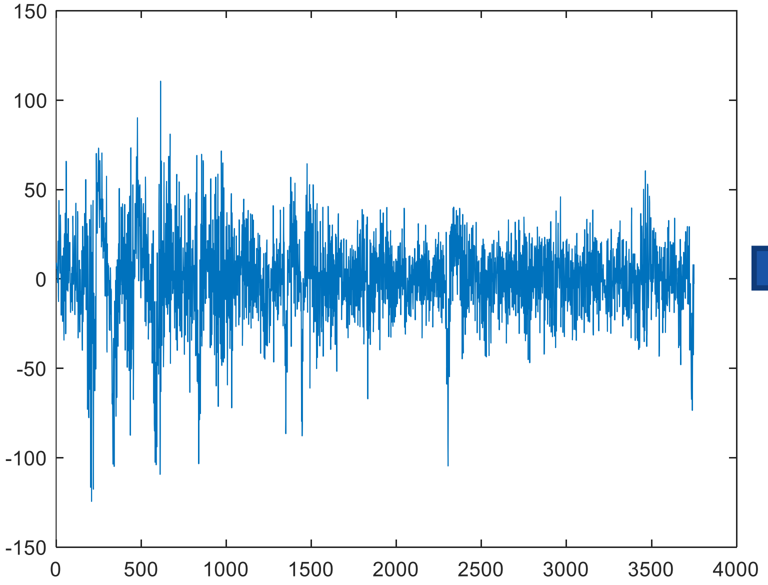
Unsupervised clustering step: wake/sleep clustering using PCA features. Or SVM classifier using the PCA Features.

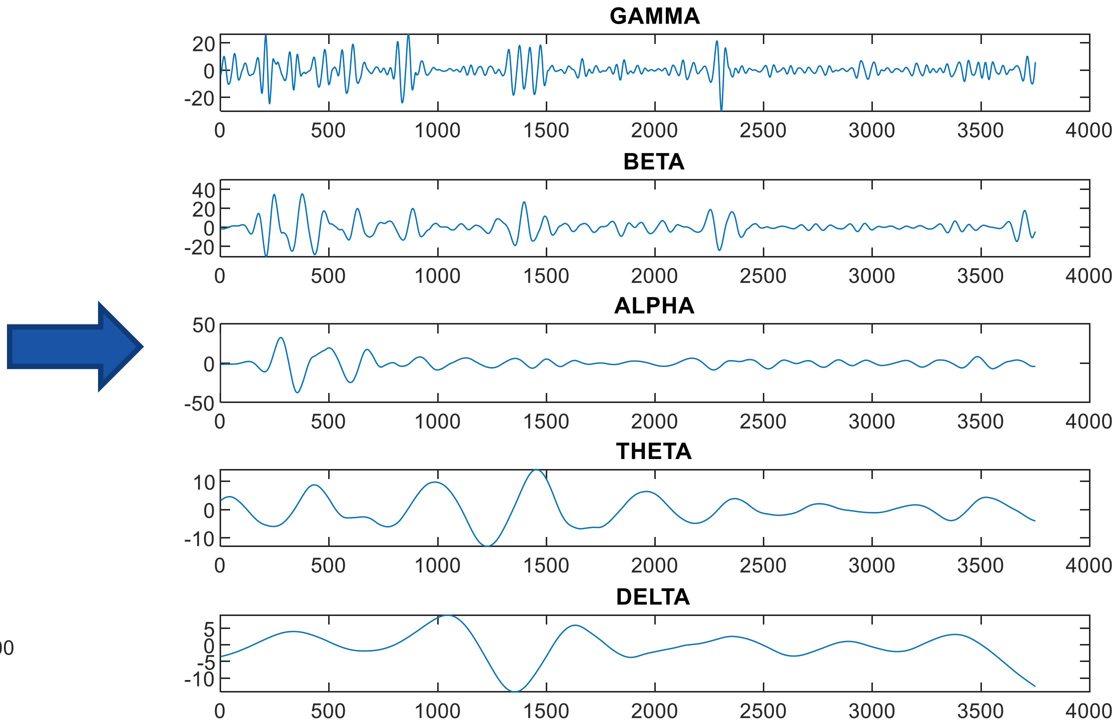
For sleep analysis, we have these **classification** possibilities: neural-net-based, cluster-based, statistical, and fuzzy classification.

General steps of the project

- Applying the machine learning workflow including training, testing, and validation. (Lecture 1- module 3)
- Different classification models and compare the accuracy.

Using Wavelet example





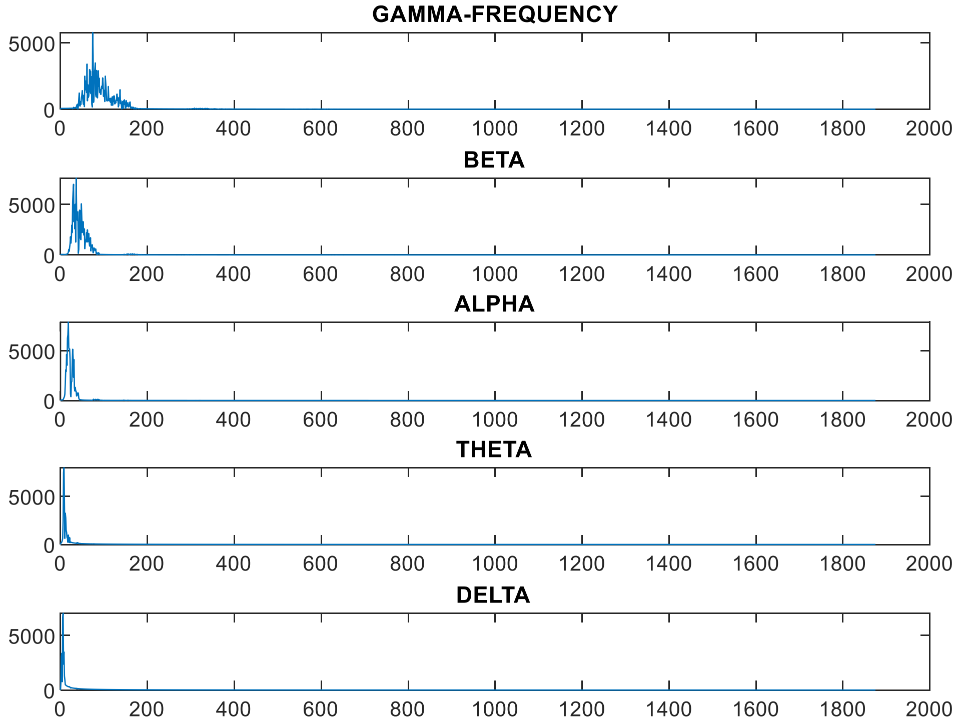
Using Wavelet example

Delta activity	< 4 Hz
Theta activity	4 – 8 Hz
Alpha activity	8 – 13 Hz
Beta activity	> 13 Hz

```
N=length(s);


waveletFunction = 'db8';
                [C,L] = wavedec(s,8,waveletFunction);

                cD1 = detcoef(C,L,1);
                cD2 = detcoef(C,L,2);
                cD3 = detcoef(C,L,3);
                cD4 = detcoef(C,L,4);
                cD5 = detcoef(C,L,5); %GAMA
                cD6 = detcoef(C,L,6); %BETA
                cD7 = detcoef(C,L,7); %ALPHA
                cD8 = detcoef(C,L,8); %THETA
                cA8 = appcoef(C,L,waveletFunction,8); %DELTA
                D1 = wrcoef('d',C,L,waveletFunction,1);
                D2 = wrcoef('d',C,L,waveletFunction,2);
                D3 = wrcoef('d',C,L,waveletFunction,3);
                D4 = wrcoef('d',C,L,waveletFunction,4);
                D5 = wrcoef('d',C,L,waveletFunction,5); %GAMMA
                D6 = wrcoef('d',C,L,waveletFunction,6); %BETA
                D7 = wrcoef('d',C,L,waveletFunction,7); %ALPHA
                D8 = wrcoef('d',C,L,waveletFunction,8); %THETA
                A8 = wrcoef('a',C,L,waveletFunction,8); %DELTA
```



Using Wavelet example

```
Gamma = D5;
            figure; subplot(5,1,1); plot(1:1:length(Gamma),Gamma);title('GAMMA');

          Beta = D6;
          subplot(5,1,2); plot(1:1:length(Beta), Beta); title('BETA');


          Alpha = D7;
          subplot(5,1,3); plot(1:1:length(Alpha),Alpha); title('ALPHA');

          Theta = D8;
          subplot(5,1,4); plot(1:1:length(Theta),Theta);title('THETA');
          D8 = detrend(D8,0);

           Delta = A8;
          %figure, plot(0:1/fs:1,Delta);
          subplot(5,1,5);plot(1:1:length(Delta),Delta);title('DELTA');

D5 = detrend(D5,0);
xdft = fft(D5);
freq = 0:N/length(D5):N/2;
xdft = xdft(1:length(D5)/2+1);
figure;subplot(511);plot(freq,abs(xdft));title('GAMMA-FREQUENCY');
[~,I] = max(abs(xdft));
fprintf('Gamma:Maximum occurs at %3.2f Hz.\n',freq(I));

D6 = detrend(D6,0);
xdft2 = fft(D6);
freq2 = 0:N/length(D6):N/2;
xdft2 = xdft2(1:length(D6)/2+1);
% figure;
subplot(512);plot(freq2,abs(xdft2));title('BETA');
[~,I] = max(abs(xdft2));
fprintf('Beta:Maximum occurs at %3.2f Hz.\n',freq2(I));
```

```
D7 = detrend(D7,0);
xdft3 = fft(D7);
freq3 = 0:N/length(D7):N/2;
xdft3 = xdft3(1:length(D7)/2+1);
% figure;
subplot(513);plot(freq3,abs(xdft3));title('ALPHA');
[~,I] = max(abs(xdft3));
fprintf('Alpha:Maximum occurs at %f Hz.\n',freq3(I));


 xdft4 = fft(D8);
freq4 = 0:N/length(D8):N/2;
xdft4 = xdft4(1:length(D8)/2+1);
% figure;
subplot(514);plot(freq4,abs(xdft4));title('THETA');
[~,I] = max(abs(xdft4));
fprintf('Theta:Maximum occurs at %f Hz.\n',freq4(I));

A8 = detrend(A8,0);
xdft5 = fft(A8);
freq5 = 0:N/length(A8):N/2;
xdft5 = xdft5(1:length(A8)/2+1);
% figure;
subplot(515);plot(freq3,abs(xdft5));title('DELTA');
[~,I] = max(abs(xdft5));
fprintf('Delta:Maximum occurs at %f Hz.\n',freq5(I));
```

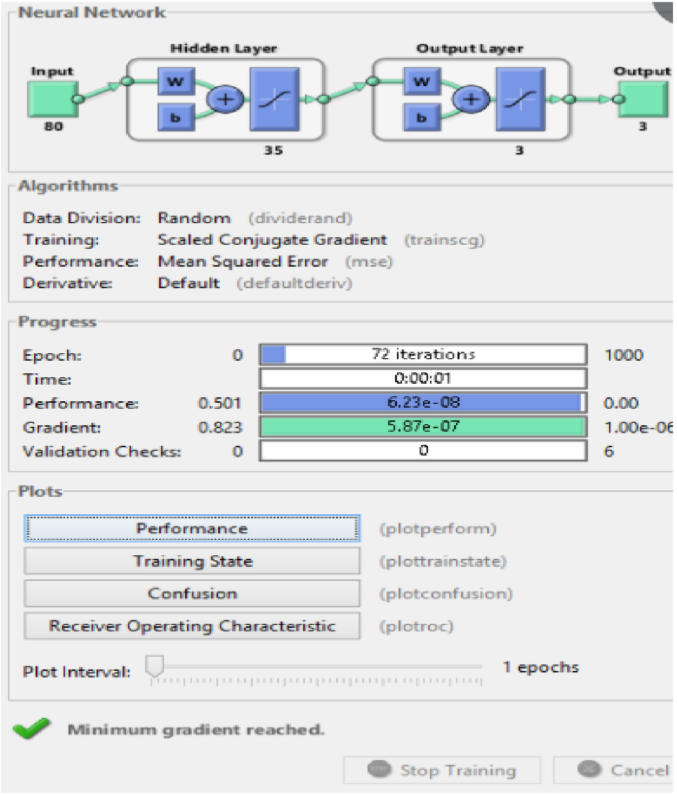
Feature Selection

Features selection resume

Use a heuristic approach based on a genetic algorithm. Or use PCA principal component analysis. To reduce the number of features: which means to apply a **transformation** that **maps** the **high dimensional feature space to a low dimensional one**.

Artificial neural network (use Pattern recognition on Matlab) (will be presented in the lab)

- Build the target format using the annotation data.
- Build the input matrix format.
- Train the network, test, and validate.
- construct the deployable version together with the extraction of the customized code for future test and use on other signals.



Additional support

- Source code The source code for the implementation including evaluation code can be found at:

https://github.com/Sebelino/hypnoscorer/

- Codes and methods used in the Laboratory sessions using supervised learning and Artificial neural networks.
- Documentation using unsupervised learning


QUESTIONS


Tack!