AI for Medical Time Series

Lecture 4 exercises

March 13, 2024

Introduction

In this exercise set, you will explore concepts covered in the forth lecture. You will get hands on experience on how wavelet transformation work and how to apply it to time series data. This week we will be working with an Electroencephalography(EEG) based Brain-Computer Interface (BCI) dataset (described here: https://ieeexplore.ieee.org/document/1300799). This dataset contains EEG data of motor imagery for feet. The data are already epoched and can be found as a 3D numpy array "bci_eeg_dataset.npy" (epochs x channels x time-samples). There is also file named "bci_eeg_dataset_times.npy" which contains time data corresponding each time-sample. The 3rd file named "bci_eeg_testData.npy" includes 1D data of 1 channel of 1 epoch that you will use as a test dataset. All files are available on Ilias. The sampling rate is 160 Hz.

Important: The purpose of this exercise is to let you explore how a wavelet transform is implemented. For this we ask you to solve them writing your own code, i.e. **without** using built-in wavelet transform functions available in existing python libraries.

These exercises require the Numpy, SciPy, and Matplotlib libraries and Python. Please use **only** these libraries for this exercise set. Solutions with other packages will **not** be accepted, and you will not get any points. Please use comments to indicate which sub-task your are answering (# Exercise 1a, etc.). The exercise will be marked as PASSED if you get **22 / 28** points or more. Points are only awarded for exercises where your code produces the expected result, and where you provide comments describing what the code does.

The exercises should be handed in by a group of two students. Copying code or solutions of individuals outside the group (e.g. submitting the code of other individuals as your own) will result in 0 points.

If you have any questions please e-mail *pinar.goektepe@unibe.ch*. You should describe what you have done so far, and what issues you have encountered. The solutions must be handed in via **ILIAS**. Deliver your submission as a compressed file (zip) containing one .py or jupyter notebook file. Please make sure to name the zip file as follows:

HW_homeworkNumber_GroupID_surname1_name1_surname2_name2.zip.

Deadline: 14:00, March 20

Exercises

1. **Time-frequency analysis** .... 28 points

   (a) **Preparing EEG data** .... 1 points

   Import the data files "bci_eeg_dataset.npy" and "bci_eeg_dataset_times.npy". Print the number of epochs, number of channels and number of time samples in the data.

   (b) **Generating Morlet wavelets** .... 10 points

   Write a function that computes a wavelet transform. This function should take as input the 1-dimensional time domain signal on which you want to apply the wavelet transform, the number of cycles for the wavelet function, sampling rate of the data and an array of 40 evenly spaced frequencies ranging from 1 to 40 Hz (1 per wavelet that you want to create) (e.g. via np.linspace(1,40,40)). You can use [-1,1] as the time range of the wavelets (e.g. via np.arange(-1,1,1/samplingRate).

   Note: you can use the following formula to generate wavelets:

   $$\psi = e^{2j\pi f_0 t} * e^{\frac{-t^2}{2\sigma_t^2}} \tag{1}$$

   Hint: f_0 is the center frequency for the motherwavelet, t is time, σ_t is the standard deviation in the time domain.

   (c) **Wavelet transform** .... 7 points

   Within your wavelet transform function convolve a wavelet with the 1-dimensional signal given as an input to the function and repeat this convolution for every wavelet with a different center frequency. The output should have the shape of number of center frequencies x number of data points.

   Hint-1: Consider applying the equivalent of convolution in the frequency domain. Remember that the result of the convolution will have the same length as the data length + kernel length - 1. You will need to cut out half of the wavelet length from the beginning and the end to have the same length as the original data.

   Hint-2: Scale the amplitudes of each wavelet by dividing with the maximum value of the wavelet so that the maximum will be 1 in frequency domain before computing the wavelet transform of the data. This will allow you to have your final results in the same unit as the original data.

Hint-3: Reminder: After a multiplication in the frequency domain, as an equivalent of convolution in the time domain, you will need to convert the data back to the time-domain.

You can compute the power of the wavelet transform as the square of the amplitudes of frequency components. Return the power of the wavelet transform as the output of your function.

Import file named "bci_eeg_testData.npy". This test dataset includes 1 trial of 1 channel with 961 time points. Use this 1D dataset to test your function. Use 6 cycles for the wavelets. The output of the function should be a 2D array with dimensions: number of frequencies x number of time samples. Plot this output as 2D image. You should obtain something similar to the exemplar figure 1 below.

Hint: You can use matplotlib.pyplot.imshow function from matplotlib library to plot 2D image.

(d) **Applying wavelet transform on single trials** .... 5 points

Choose one channel from the BCI data to apply wavelet transform with Morlet wavelet. Indicate which channel you chose to work on and use the same channel for the rest of the assignment.

Apply the wavelet transform function that you wrote in step (c) to each trial of the chosen channel using a loop and then store the results in a 3D numpy array with dimensions: number of epochs x number of center frequencies x number of time samples. Use 6 cycles for the wavelets.

You can also normalize the output by baseline. To do this, you should identify which are the baseline time points which are before the event onset (< 0 ms). Get the wavelet transformed data at these baseline time points. Then, compute the mean of the wavelet transform over the baseline.

Apply baseline normalization at every time point in the wavelet transformed data as value_new = (value - baseline_mean)/baseline_mean.

(e) **Generating time-frequency plots** .... 2 points

Average the wavelet transform results that you obtained in step (d) over all epochs and visualize the time-frequency plot as a 2D image in which the x-axis represents time, the y-axis represents frequencies and the plotted values represent the baseline-normalized power of the frequencies over time. You should obtain something similar to the exemplar figure 1:

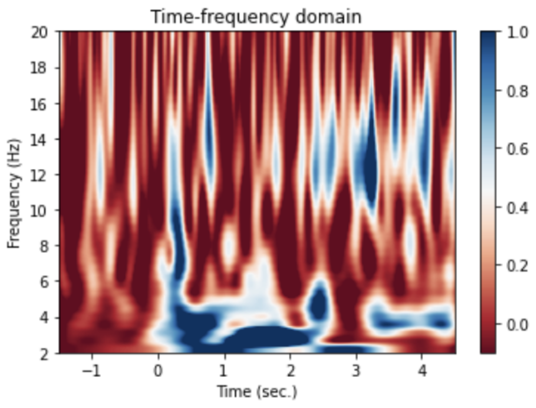

Figure 1: Example time-frequency plot

Hint: You can use matplotlib.pyplot.imshow function from matplotlib library to plot 2D image.

(f) **Exploring the effects of number of cycles in wavelets** .... 5 points

Repeat the previous step with the wavelets that have 3, 8, 24 cycles while keeping the center frequencies the same (always starting from the initial epoched data that you loaded in step (a)).

Describe what you observe when using different number of cycles for the wavelets. Discuss a strategy to better capture the time-frequency properties of a signal based on your observations.