BSPM Tutorial II

Nogay Küpelioglu

June 9, 2020

Abstract In this tutorial we have covered all the basic and necessary steps in dealing with the EEG data. These steps include the preprocessing of the data, epoching/segmentation of the data, feature extraction using different methods and feature evaluation using fisher's scores. The extracted features were then used to train classifiers that would separate the error-related potentials from the baselines. Different methods of extracting the features and different modes of classification were tried.

Keywords Brain–computer interface · Data preprocessing · Feature extraction · Feature classification · Modelling

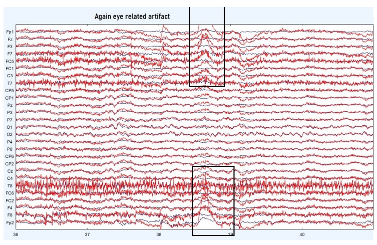
1 Introduction

The data that we used during the tutorial was generated through an experiment. In this experiment the subject by pressing arrow keys tried to move the cursor to a target stimulus that was presented on the screen. However with about a third chance the cursor moved to the wrong direction. This would generate error-related potentials in the subjects brain. The experiment was conducted in order to generate this data, which contained both a baseline and error-related potentials, so that they can be observed and their differences compared. As for error-related potentials, we have seen in the previous presentation assignment that they have a wide-variety of use in the rehabilitation of paralysed patients (P300) to the personalization of autonomous driving. If we can analyse the underlying structure of the error-related potentials and are able to separate them from the baselines with a good success rate it is a contribution to the higher-level use cases of the error related potentials.

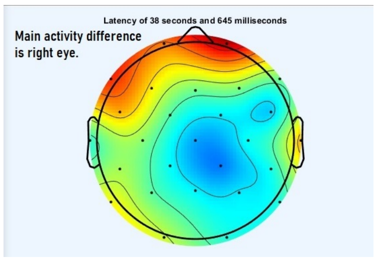
Author: Nogay Küpelioglu

2 Exploring the data

In the first part of the tutorial we were to experiment on two datasets one of which was contaminated with artifacts, whilst the other was already cleaned. The purpose of this to introduce us to the controls of EEGLab Matlab plugin and at the same time learn to recognize how artifacts look like and how they contaminate the data. Through the data I found several artifacts, but visually going through the data I realized that the most common cause for artifacts was eye-related activities. One such example can be seen in figure 1. The activity in the fronto-central channels were contaminated with the EOG activity, which caused large spikes on those channels. This is one of the reasons why we need to preprocess the data, because such activities are not brain-related but related to the muscles that control the eyes.
P.S. This is the only part in which Matlab was used.



(a) Artifact in time domain


Latency of 38 seconds and 645 milliseconds
Main activity difference is right eye.

(b) Artifact in the topoplot

Fig. 1: Example eye-related artifact

3 Data preprocessing

Although we have continued the work from the preprocessed data, when we did our qualitative comparison of the clean and unclean data the need for preprocessing was obvious. This is caused by the senstivity of the EEG amplifiers. In order to get a signal from an object that is behind a very thick bone structure the electrodes have to be very sensitive as well. This makes them prone to the external interferences. That means that EEG-signals are contaminated by non-brain signals as was shown in figure 1 and they need to be cleaned out of the data. Finally EEG-signals are not generated by single neurons. Local field potentials are being measured. Given the positioning of the electrodes on the skull the field potentials coincide between the channels making the data highly correlated, which also has to be taken care of.

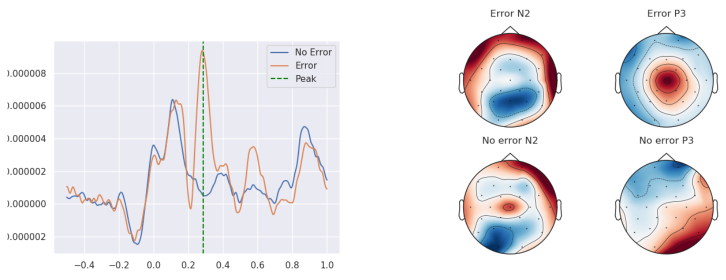
Preprocessing is done in several steps each step contributing to it:

– **Bandpass filtering** Is done to get rid of the dc-drifts and various types of noise. EEG signals have a very low signal to noise ratio. Some muscular artifacts (such as caused by clenching) have very high frequency and by lowpass filtering we get rid of them as well. Eye-related artifacts (such as horizontal eye movements and eye blinks) can also be removed by a highpass.

- **Occular artifact removal** Done by analysing the EOG channels. Eye related artifacts can be seen very clearly in EOG-channels and by decorellating the EEG channels by EOG channels we can get rid of them.
- **Channel/Segment rejection** Used to get rid of the parts of the data that is contaminated by artifacts. This would be done after rejecting the artifact components after we do an independent component analysis and remove the artifact components from the data in the case that rejecting artifact components is not fruitful in cleaning the portions.
- **Common average re-referencing** Would help clean the artifacts which are contaminating all the channels.
- **Downsampling** Is done to reduce the computational load in further processing the data.

4 Data segmentation

As we are dealing with event-related potentials we only need the EEG-data corresponding to those events. Therefore, by epoching we segment the only the data portions that we require. This can be seen in the following reproduced figures. These figures were generated on the channel *Cz* and by averaging all the error and no-error events separately.



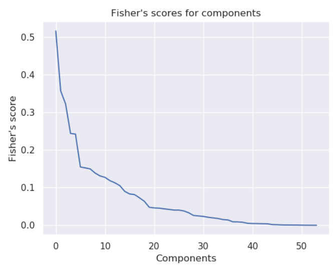
(a) In time domain (b) In topoplot

Fig. 2: Error related potential

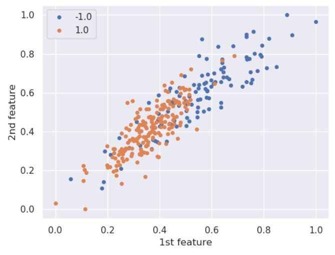
The area between 200-300 ms after the events show the peak of the error-related potential and is the main difference between the data of error and no-error events. As the course of the classification task is to discriminate between the error and no-error events this are and another subsequent peak at around 500 ms after the event interest us in the subsequent feature extraction.

5 Feature Extraction and evaluation

In the tutorial we extract 2 features per channel that are located 210.9 ms and 285.2 ms after the event markers, because they show the most prevalent difference between the error and no-error events. In total we gather $27 \times 2 = 54$ features from each event. In the tutorial text it was explained that this was not sufficient enough to generalize for all such data and why. Therefore in the cross-comparison of methods in my report I suggest an alternative. We then evaluate our features using fisher's rank. The following two figures are the reproduction.



(a) Fisher's rank



(b) First and second features

Fig. 3: Feature evaluation

6 Modelling and classification

In this part of the tutorial we use a Shrink-LDA classifier to classify the features to the labels of the events. We also use K-Folds cross-validation. That is we divide the data into 10 separate folds and use 9 of those folds the train the classifier and use 1 of the folds to test the classifier we trained. We do it 10 times and allow each of the folds to be tested, while use the remaining 9 to train again. At every step we take into account the misclassification rate. In my case I used the $1 - MCR(fold)$ the classification accuracy. At the end I averaged the classification accuracy of each fold to generate a result. The Shrink-LDA classifier mean accuracy in 10-Folds 10-Crossvalidation test yielded the following result: 82.33%.

Although this seems successful enough, the features that we extracted are fairly limited. The peak of the main difference may not exactly be on the same timepoint at every channel and if we look at the figure 2b the error-related potential may not be shown in every channel. Also the difference between error and no-error events are not limited only at the timepoints in question but can be observed at an interval. Therefore, I suggest an alternative feature extraction method in the next part of the report. Finally, I have already employed a min-max scaling for all of the trials, after I noticed that without it I could not have plotted the figure 3b, because the numbers were too small and the features in fishers rank could not be plotted without scaling.

7 Cross-comparison of methods

If we extract only two time-points from every channel, I believed it would not be sufficient to separate error and no-error events. In order to back-up my theories

I have plotted error and no-error segments for every channel and to see exactly between what time-points there is a difference between error and no-error segments and on which channels we can see this difference. If we visually inspect this figure

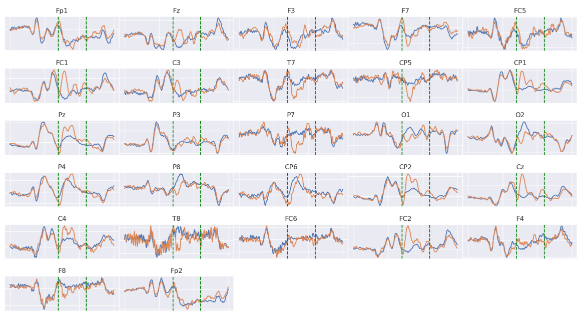


Fig. 4: Error vs no error events for all channels

we can see that the main difference between the error and no-error events reside in the time points between 0.2 and 0.6 seconds, which are shown by green dashed vertical lines. If we inspect the channels properly this difference is expressed in the 7 channels the most. These channels are F_z, FC_1, CP_1, CP_2, C_z, C_4, FC_2. The amount of timepoints for a channel corresponding to the interval between 0.2 and 0.6 seconds in the downsampled data is 14. Instead of the 54 features that we have used in the original tutorial, I have used the $7 \times 14 = 98$ features for each event and used the same Shrink-LDA classifier with the same shrinkage parameter (0.4) and used the same 10-Folds 10-times crossvalidation test the same in the original tutorial. The mean classification accuracy was 88.33% compared to the original which was at 82.33%. For the test on the recall.set lda.mat corresponds to the original tutorial method, while the extension corresponds to the ldaextra.mat.

8 Conclusion

In this tutorial we have learnt about and applied every aspect of EEG data processing, which is very important to brain-computer interfaces. These aspects include data preprocessing, feature extraction, feature evaluation and feature classification. There are different ways of applying each of these aspects and all of them are equally important. We learnt why/how the error-related potentials are generated and how we can apply these aspects of the EEG-processing to the data we were given. We have seen how we can improve them and the reasoning behind these methods and extensions.

9 Requirements

Except for the very first part the code was written in python using jupyter notebook. I do not include a requirements.txt, because I used an anaconda-python executable. Here is the code to set up an anaconda environment.

```
conda install -c conda-forge mne
conda install -c intel scikit-learn
conda install -c anaconda pandas
conda install -c pchrapka scikit-feature
```