MACHINE LEARNING ENGINEER NANODEGREE

Grasp-and-Lift EEG Detection
Capstone Project

ANDREI UNGUREANU

2020

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 DEFINITION

1.1 Project overview

Patients who have lost hand function due to amputation or neurological disabilities live a life that is deprived of the ability to perform basic tasks that are trivial to a healty person. Using a *brain-computer interface* (BCI) would help in restoring a patient's capabilities by enabling them to use prosthetics that provide similar functionality to the one that has been lost. The development of such an interface starts from gathering and analysing *electroencephalogram* (EEG) data. This method of recording the electrical activity of the brain uses electrodes that are attached to the scalp in various positions and configurations and measures voltage fluctuations (with an amplitude of about 100 µV).

1.2 Problem statement

The problem is defined in the kaggle Grasp-and-Lift competition:[1] we are challanged to identify when a hand is performing grasp-and-lift actions on an object based on EEG recordings taken from healthy subjects. The data comes from 12 different subjects which have completed a series of 10 trials where each trial represents around 30 performed actions. During these trials, the object's weight, surface friction, or both were changed. One trial is defined as a series of sequence actions from the first one to the sixth. They are described below:

- HandStart = reach for object
- FirstDigitTouch = grasp object with the thumb and index finger
- BothStartLoadPhase = lift object and hold it for a couple of seconds
- LiftOff = put object back on the support surface
- Replace = replace the object
- BothReleased = return the hand to a designated rest position

One trial data consists in 32 EEG channels from the total number of channels arranged according to the Brainproduct EasyCap-M1 montage (Figure 1). The sampling rate used is 500 Hz. For each frame we are provided with six values of either 0 or 1 that represent whether each of the six actions is occurring or not, with a precision of ± 75 frames. This means that two or more actions can overlap in the events files provided.

1.3 Metrics

The competition task is to predict the occurance of each of the six actions on a set of test data. The predicted outcome is evaluated by calculating the mean of the individual *areas under the ROC curve* [2] for each predicted action.
The ROC curve is created by plotting the *true positive rate* (TPR) against the *false positive rate* (FPR) at various threshold settings, where:

$$\mathrm{TruePositiveRate} = \frac{\mathrm{TruePositives}}{\mathrm{TruePositives} + \mathrm{FalseNegatives}} \tag{1}$$

and

$$\mathrm{FalsePositiveRate} = \frac{\mathrm{FalsePositives}}{\mathrm{FalsePositives} + \mathrm{TrueNegatives}} \tag{2}$$

1 http://www.kaggle.com/c/grasp-and-lift-eeg-detection
2 https://en.wikipedia.org/wiki/Receiver_operating_characteristic

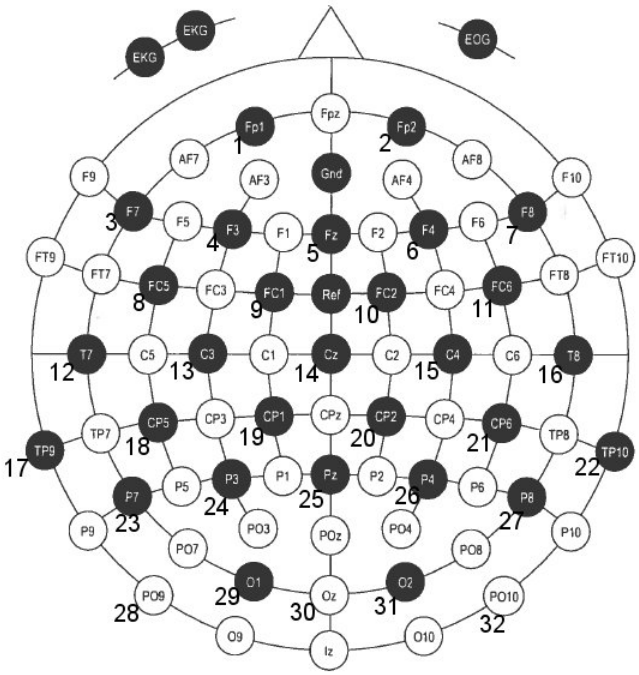

Figure 1: Diagram of the Brainproduct EasyCap-M1 montage and the numbered 32 channels used to collect the data (from https://www.kaggle.com/c/grasp-and-lift-eeg-detection/data).

2 ANALYSIS

2.1 EEG Bands

Before going into the analysis of the data a short introduction about the EEG signal frequency bands is required. They are grouped in the following major categories:

- **Delta**: has a frequency of **3 Hz** or below. It tends to be the highest in amplitude and the slowest waves. It is normal as the dominant rhythm in infants up to one year and in stages 3 and 4 of sleep.

- **Theta**: has a frequency of **3.5 to 7.5 Hz** and is classified as "slow" activity. It is perfectly normal in children up to 13 years and in sleep but abnormal in awake adults.

- **Mu**: has a frequency between **7.5 and 13 Hz**. Is usually best seen in the posterior regions of the head on each side, being higher in amplitude on the dominant side.

- **Beta**: considered a "fast" activity. It has a frequency between **14 and 31 Hz** . It is usually seen on both sides in symmetrical distribution and is most evident frontally.

- **Gamma**: represent binding of different populations of neurons together into a network for the purpose of carrying out a certain cognitive or motor function. It has a frequency between **32 and 100 Hz**.

As mentioned by [2] the bands that present interest for the motor activity we are interested in analysing are extracted from the mu (about 8-12 Hz) and the beta (about 16-24 Hz) frequency bands.

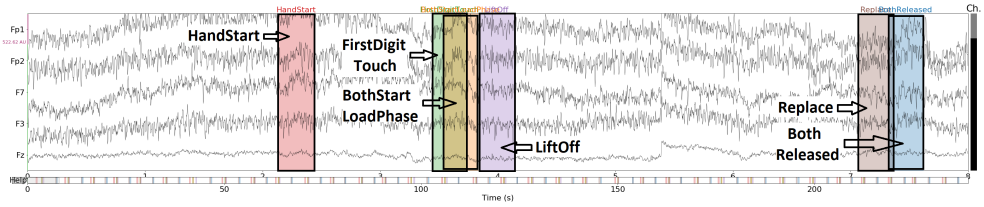

Figure 2: Raw data with events for subject 1, series 1, first 5 channels, first 8 seconds

2.2 Raw data exploration and visualisation

The de facto library for analysing and visualising EEG data in python is called **MNE** [3] and it is also what we use in this analysis step.
We start by creating an `info` object which contains the EEG channel names, the sampling frequency and the aforementioned montage (the spatial arangement of the electrodes on the scalp). We then load the data for `subject1`, `series1` as a `pandas` dataframe and then transform it to a `numpy` array which We use in conjunction with the `info` file created earlier to create a `raw` object. This object is the backbone of the MNE library and will be used throught this project. .
The next step is to add the `events` data to the `raw` object. For this we created a helper function called `get_column_annotations` which takes each of the 6 events columns names and the events pandas dataframe as parameters and determines the starting frames end ending frames for the events that happened in a particular column. Adding the `annotation` object created using the helper function to the `raw` object and ploting the raw object for the first 5 channels and the first 8 seconds results in Figure 2

The time windows for the 6 events are marked in the figure and, as mentioned in the competition documentation, one can see that the events can overlap.

2.3 Exploring the data using the CSP (Common Spatial Pattern) method

One interesting method of evaluating EEG data is called **CSP (Common Spatial Pattern)**. This method designs spatial filters in such a way that the variances in the filtered time series data are optimal (in the least squares sense) for discrimination [1]. The downside of using this method is that it can only discriminate between two classes of actions. Therefore it is not strictly useful for designing a model that can detect the six gestures we presented above but it produces some intersting results that can influence further research.
We start by loading the data files for a particular `subject1` and using the aforementioned **MNE** library to create a `raw` object that contains the 32 data channels and the corresponding 6 binary signals that denote an activation for each of the actions defined in the experiment. We then filter the data according to the **mu** and **beta** bands and select time windows of -2 seconds to -0.5 seconds before the **Replace** event and 0.5 seconds to 2 seconds after the **Replace** event. This effectively selects time windows where the hand was actively moving versus time windows when it was not and adds them to the `epochs_all` object and tagged appropriately. After normalization the **CSP** algorithm is applied to the data and the results are plotted against a topographic map of the brain (Figure 3). This is an indication of the fact that around electrode **C3** the signal for the two bands denotes an activity for the hand movement case versus the non-movement case.

3 https://mne.tools/dev/index.html

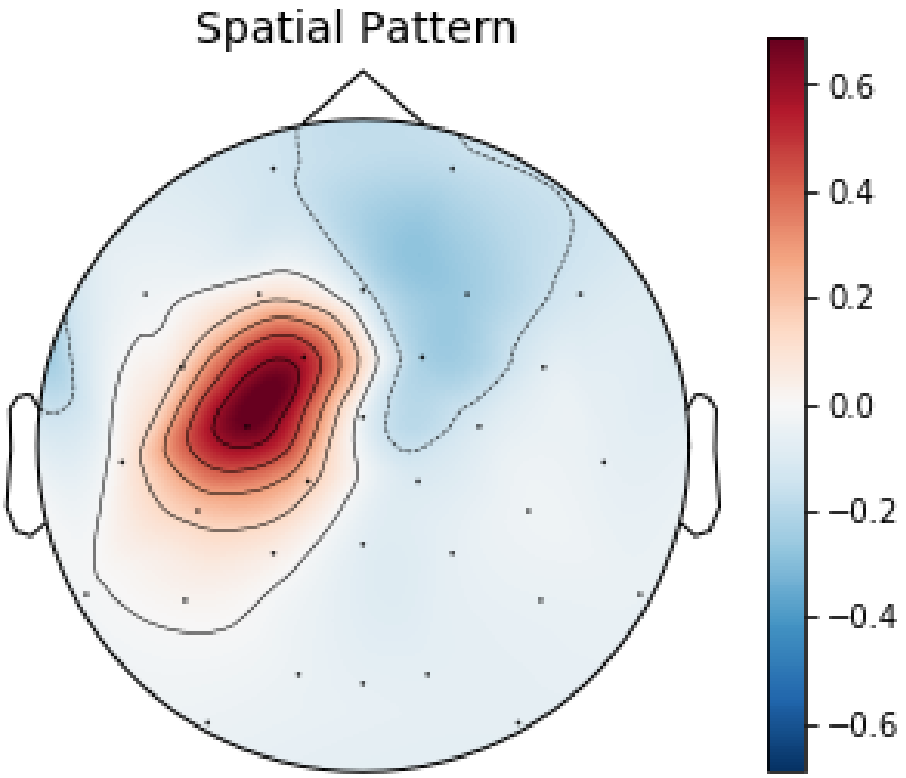

Figure 3: Left motor cortex activation corresponding to a right hand movement for subject 2

3 DEEP LEARNING MODEL

The data we use for training and validation can be found at the following link: https://www.kaggle.com/c/grasp-and-lift-eeg-detection/data. We start off by loading all the data into a `glob_object` and calculating two types of metrics for normalization, mean plus standard deviation and maximum plus minimum.
We then proceed to arranging the data in batches in order to feed it to the neural network. We chose to use series 1 through 6 to build the training set and series 7 through 8 to build the validation set. Since the events we need to predict have a duration of 150 samples this is the value we chose to use for our window length. We also chose a `batch_size` of 32 meaning that each batch in our data will consist of 32 windows of 150 samples of the 32 recorded channels. The `generate_data` function loads each file, cuts the excess data using a threshold in order to have the size needed to be batched, normalizes it (using the MaxMin approach) and calls the `reshape_data` which rearranges it. Each window containing the 32 channels of 150 samples will have a one 6-dimentional binary event vector as an output, which will denote whether or not each of the six actions has occured. This output is considered to be the last event that occured at the end of the respective window.
After proceeding with these steps our training data has the following shapes:

```
X_train: (3046, 32, 32, 150)
Y_train: (3046, 32, 6)
```

We repeat the same procedure for the validation data and get the following shapes:

```
X_valid: (646, 32, 32, 150)
Y_valid: (646, 32, 6)
```

The network that we chose uses 8 one dimensional convolution layers, 4 max pooling layers, 2 dropout layers and 3 linear layers. The detailed architecture is provided below:

```
OneDimensionalConvolution(
  (c1): Conv1d(32, 16, kernel_size=(3,), stride=(1,))
  (c2): Conv1d(16, 8, kernel_size=(3,), stride=(1,))
  (m1): MaxPool1d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (c3): Conv1d(8, 32, kernel_size=(3,), stride=(1,))
  (c4): Conv1d(32, 16, kernel_size=(3,), stride=(1,))
  (m2): MaxPool1d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (c5): Conv1d(16, 64, kernel_size=(3,), stride=(1,))
  (c6): Conv1d(64, 32, kernel_size=(3,), stride=(1,))
  (m3): MaxPool1d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (c7): Conv1d(32, 64, kernel_size=(3,), stride=(1,))
  (c8): Conv1d(64, 32, kernel_size=(3,), stride=(1,))
  (m4): MaxPool1d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (d1): Dropout(p=0.1, inplace=False)
  (l1): Linear(in_features=128, out_features=64, bias=True)
  (d2): Dropout(p=0.1, inplace=False)
  (l2): Linear(in_features=64, out_features=64, bias=True)
  (l3): Linear(in_features=64, out_features=6, bias=True)
)
```

Since one window can contain multiple events the `BCELoss` was chosen and for the optimizer the `SGD` with `lr = 0.01` and a momentum of `0.9`

REFERENCES

[1] Herbert Ramoserr at all. Optimal Spatial Filtering of Single Trial EEG DuringImagined Hand Movement, IEEE Transactions on rehabilitation engineering, Vol. 8, No. 4, December 2000, pp 441-446.

[2] Fabien Lotte. A Tutorial on EEG Signal Processing Techniques for Mental State Recognition in Brain-Computer Interfaces. https://hal.inria.fr/hal-01055103, 2014, HAL Id: hal-01055103.

[3] Hendrik Weideman. Kaggle's Grasp and Lift EEG Detection Competition. https://hjweide.github.io/kaggle-grasp-and-lifte/, 2015. [Online; accessed 15-January-2020].