BCI-4-ALS- Mid Semester Assignment Report

Assaf Peleg
Dor Zazon

July 9, 2021

Contents

Chapter 1

Introduction

1.1 Rationale

Our goal in this assignment is to record 120 trials of motor imagery a day for three days and then train an offline machine learning model. Our code will be based on Asaf's Harel SSVEP offline model trainer which has been modified by Eden and Adi for motor imagery. As our recording with the Open BCI system was not successful, we will use the original recording by Eden's and Adi's as other recordings that we received were of very low quality. These recordings include 60 trials a day for three days of motor imagery training.

1.2 Method

We used Eden's and Adi's recordings with all the electrodes, pre-processed them(using our own set of artifacts removal code that is closely related to the original), segmented them, extracted 5 features using the original motor imagery features and feature selection code, given to us by Eden's and Adi's(but we did alter the file to support infinite concatenation of the previous days recordings and some other changes we will elaborate on later) and then finally we trained a LDA and SVM RBF(for multi class) classifiers. The changes to the files consisted of:

1. **Preprocess**- We removed the general removal of 1,2,3 electrodes and added parameters to control these removal. However, in our tests of the classifers, we did not remove any electrode. In addition, we added options to choose between low-high pass 0.5hz-40hz filter and high-pass-notch 0.5hz-40hz filter, which is different from the original code which included a low-high-notch pass filter between 0.5hz-30hz. **Next, we changed the notch filter to be on all the channels and not on just the first and second electrodes.** Additionally, we added resampling and average referencing to the set of artifacts removal functions. Additional miscellanies changes were to allow several plots. **In conclusion, in our tests we did a high-pass filter of 0.5hz and a notch filter of around 50hz on all the electrodes, average referencing and resampling**.

2. **Segmentation**- We kept the original code but fixed the numChans assignment which gave wrong number of channels, and another change for the loading of EEG_event which caused an error on our version of matlab(R2018b). **In conclusion, the main change we did was to fix the numChans assignment**.

3. **Feature Extraction**- We retained the original feature extraction code but changed the end to allow for infinite concatenation of previous days recording. In addition, instead of testing only on the last day, we sampled the test sample from all of the data using 5-fold-cross-validation instead of using a test sample from the last day. Additional miscellanies changes were to allow several plots. Please see chapter 4 for an in-depth look at the best features. **In conclusion, the main changes were to allow infinite previous day concatenation and the test sample to spread across days using 5-fold-cross-validation.**

4. **Model Learning**- First, we altered this file to work with the changes that were made to the feature extraction file. In addition, we added the option to use a SVM RBF classifier. Additionally, we added support for 5-fold-cross-validation for both classifiers. It's important to note that the best features selection was done on one un-related and completely different 0.8 to 0.2 fold, and thus there might be a data leak from the test samples to the train samples for some folds. **In conclusion, the main change here was to add another classifier and cross-validation**.

5. **Main Script- We created our own main script to use all of the functions and train our model.**

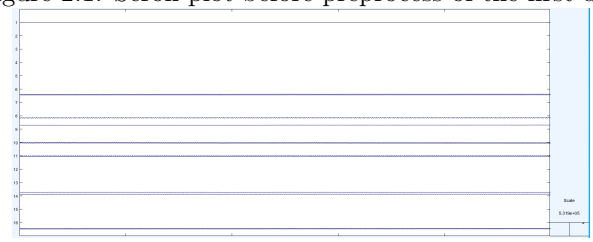
Chapter 2

LDA Results

2.1 First Day- Preprocessing

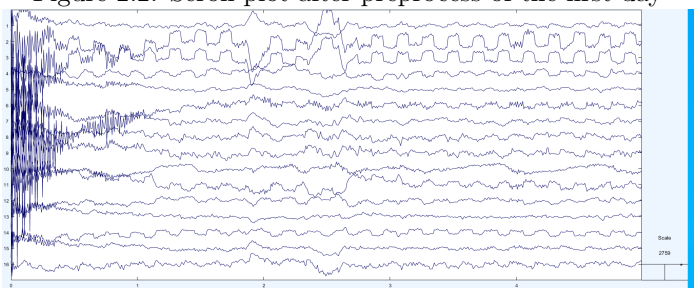
The preprocessing of the data is very important. If you try to scroll through the data **of the first day**, you'll see that the scale of each electrode is vastly different, and therefore you will get something like this after adjusting the scale:

Figure 2.1: Scroll plot before preprocess of the first day



After we preprocess the data keeping all of the electrodes, as described before, using a high-pass of 0.5hz and a notch filter of around 50hz, average re-reference and re-sampling we'll get something like this:

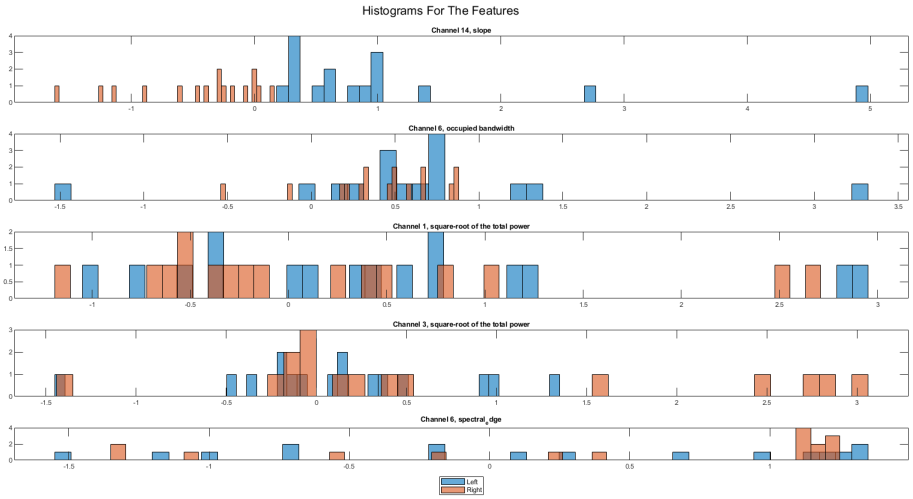
Figure 2.2: Scroll plot after preprocess of the first day



2.2 First Day- Extracting Features

After we preprocess our data, we'll use the default code for segmenting it(with some bug fixes alterations) and preparing it for feature extraction. As described before, we did not alter the features but we did add support for concatenation of infinite previous days. In addition, we are taking our test sample from all of the days using 5-fold-cross-validation, instead only the last one as was in the basic code. As we did not remove any electrode, our total sum of features is over 200. We will use, as in the original code, an automatic feature selection algorithm called "neighborhood component analysis" which will allow us to choose only 5 features. After we created our features and choose the best first 5, we plotted a histogram to aid us in understanding of the chosen features:

Figure 2.3: Features Histogram



The chosen features are, in order of the histogram plot:

1. Channel 14, slope
2. Channel 6, occupied bandwidth
3. Channel 1, square root of total power
4. Channel 3, square root of total power
5. Channel 6, spectral edge

From the histogram plot, it's pretty clear why the first and second feature were chosen. Regarding the others, it's a bit hard to see why they would be useful.

2.3 First Day- Training The LDA Model

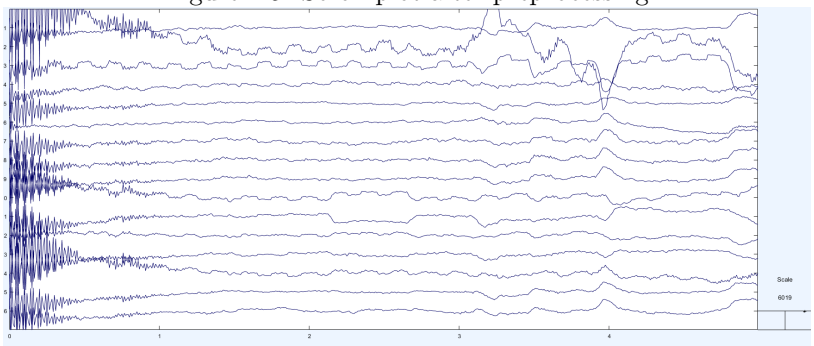
After training our model, we got **91.6667%** accuracy.

2.4 Second Day- Preprocessing

Next, we preprocessed separately the second recording day. The before and after plots are shown below:

Figure 2.4: Scroll plot before preprocessing

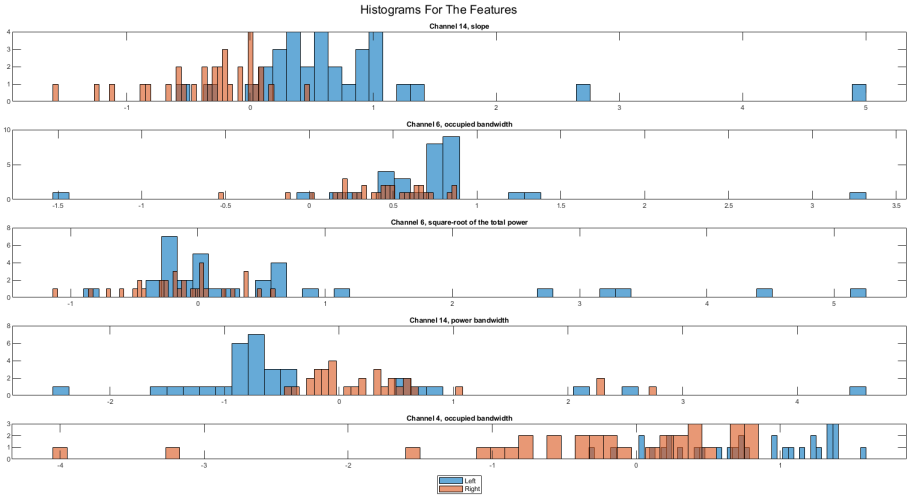
Figure 2.5: Scroll plot after preprocessing



2.5 First&Second Day- Extracting Features

We'll now concatenate the first & second day and extract the same features as before. We are using the same train/test from the first day but adding to it train/test from the second day.

Figure 2.6: Features Histogram



The chosen features are, in order of the histogram plot:

1. Channel 14, slope
2. Channel 6, occupied bandwidth
3. Channel 6, square root of total power
4. Channel 14, power bandwidth
5. Channel 4, occupied bandwidth

The first 2 features are the same, but it seems that in order to distinguish between left & right from the data added, other features were chosen in order to do so.

2.6 First&Second Day- Training The LDA Model

After training our model, we got **88.3333%** accuracy which is understandable, as we know that the brain is non-stationary.

2.7 Third Day- Preprocessing

Next, we preprocessed separately the third recording day. The before and after plots are shown below:

Figure 2.7: Scroll plot before preprocessing

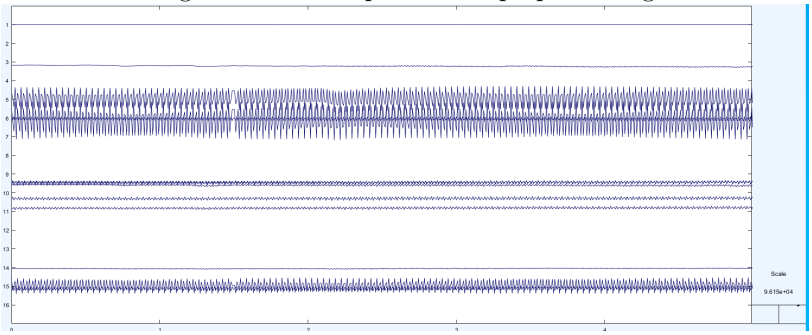
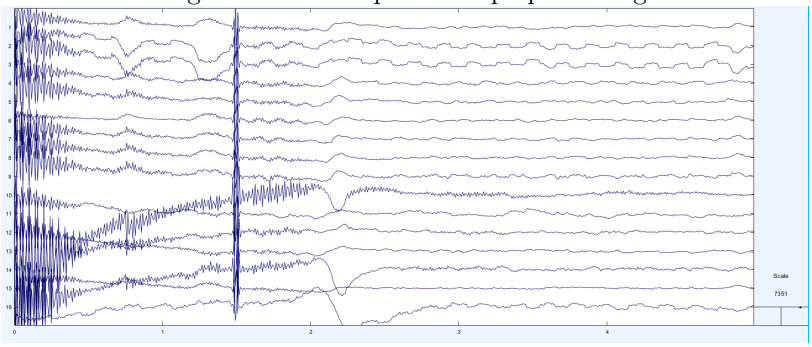


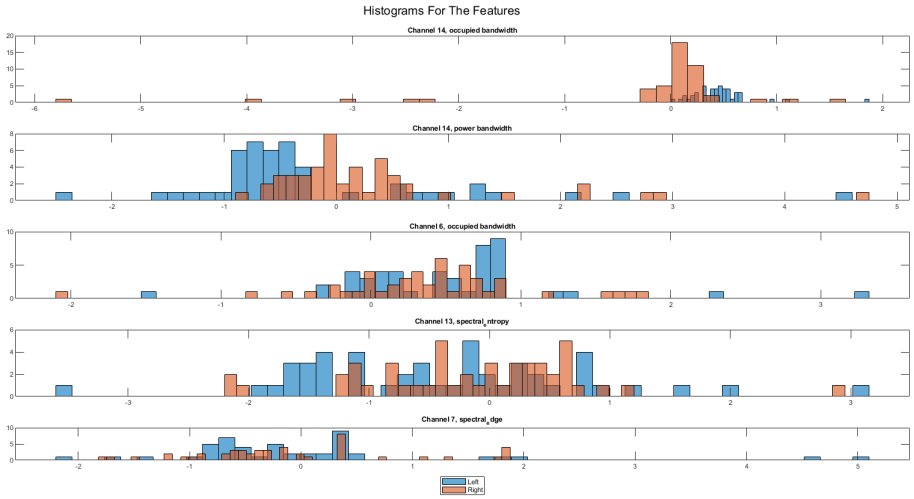
Figure 2.8: Scroll plot after preprocessing



2.8 First&Second&Third Day- Extracting Features

We'll now concatenate the first, second & third day and extract the same features as before. We are using the same train/test from the first and second day but adding to it train/test from the third day.

Figure 2.9: Features Histogram



The chosen features are, in order of the histogram plot:

1. Channel 14, occupied bandwidth
2. Channel 14, power bandwidth
3. Channel 6, occupied bandwidth
4. Channel 13, spectral entropy
5. Channel 7, spectral edge

Now, we got mostly new features but the channel 6 occupied bandwidth seems to be good throughout the days.

2.9 First&Second&Third Day- Training The LDA Model

After training our model, we got **77.2222%** accuracy which is again, understandable, as we know that the brain is non-stationary. Nevertheless, it seems we got pretty good results all in all.

Chapter 3

SVM RBF Results

Our final order of business will be to see if we can get better results with another machine learning model. Our chosen model is the SVM with the RBF kernel(using onevsone for multi-class).

3.1 Preprocessing

We will do the same preprocessing as we did before for the LDA model.

3.2 First&Second&Third Day- Extracting Features

As in the preprocessing section, we will do the same here as we did before. Therefore, the features will be the same. In addition, we used the rng function at the start of our script to ensure the same train/test separation and other random aspects to be the same as with the LDA model.

3.3 Training The SVM RBF Model

We trained the SVM RBF model and got the following results:

1. First day- We got 93.3333% accuracy.
2. First&Second day- Surprisingly, we got 92.5% which is somewhat the same as the first day.
3. First&Second&Third day- Alas, unfortunately, we went all the way down to 82.7778%.

Chapter 4

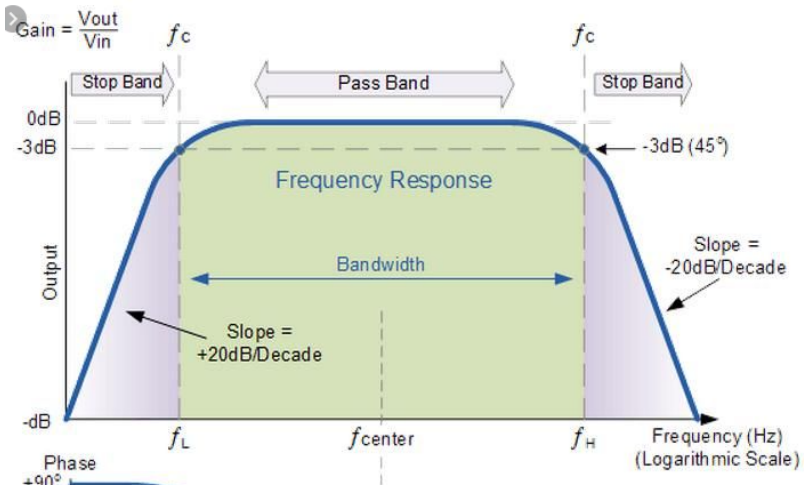
Best Features

In the final chapter, let's try and dive deeper into the best features chosen.

4.1 Slope

The slope of a segment of data is calculated by polyfitting on the transposed PSD estimate frequency vector, and then taking the first coefficient. **See Figure 4.1**

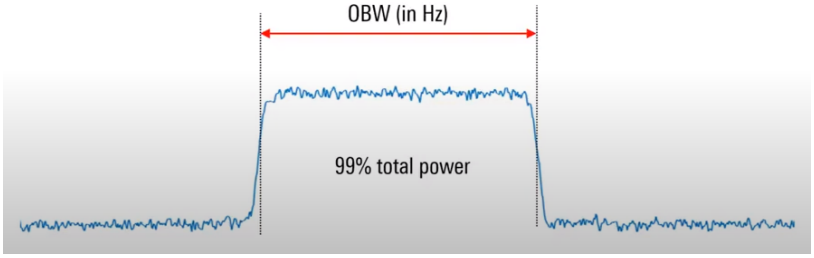
Figure 4.1: Slope



4.2 Occupied Bandwidth

The occupied bandwidth of a segment of data is the bandwidth that contains 99% of the total PSD signal power. **See Figure 4.2**

Figure 4.2: Occupied Bandwidth



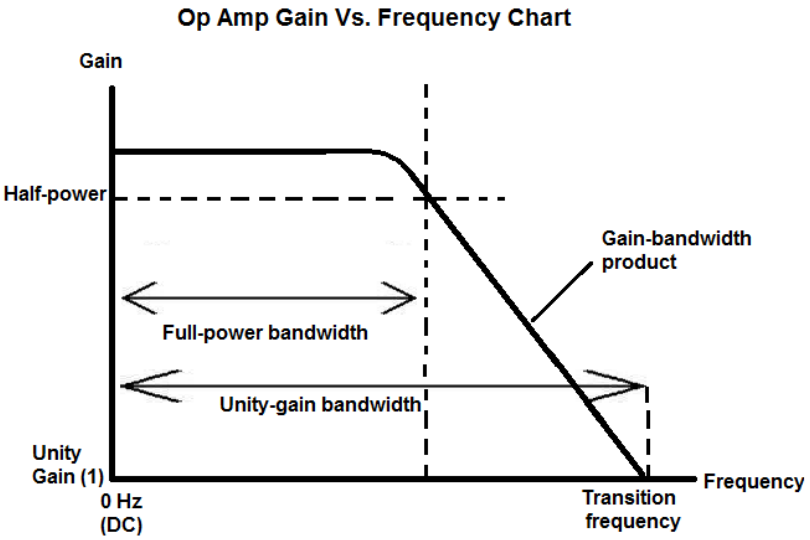
4.3 Square Root Of Total Power

The square root of total power of a segment of data is calculated by calculating the root of the sum of the PSD estimate total power.

4.4 Power Bandwidth

The power bandwidth of a segment of data is the 3-dB bandwidth of the normalized PSD estimate. For example, the full power bandwidth can be seen in **Figure 4.3**.

Figure 4.3: Full Power Bandwidth



4.5 Spectral Entropy

The spectral entropy describes the complexity of a system. It is calculated as the entropy on the normalized PSD estimate. The steps to calculate it are as follows:

1. Normalize the PSD

$$p_i = \frac{P(\omega_i)}{\sum_i P(\omega_i)}$$

2. Calculate the entropy

$$-\sum_{i=1}^{n} p_i \ln p_i$$

4.6 Spectral Edge

The spectral edge frequency of a segment of data is the frequency in which 90% of total power resides below it and 10% of the total power resides above it.

Chapter 5

Appendix- Related Work

In the appendix, we'll go over the code that we didn't write but based our work on.

5.1 Segmentation

The segmentation code written by Asaf Harel, segments the data from each channel into segments that start a little bit after each marker and a little bit before the the end of the trial. This is done by first locating the start of each trial and using the function called sortElectrodes to segment the data.

5.2 Features Extraction and Selection

The feature extraction code and selection written by Asaf Harel and modified by Eden and Adi and also by us calculates the PSD estimate of each trial and channel and then transforms them to features(in addition to the band powers). We dived deeper into the best features characteristics in chapter 4 but we didn't go over how these features were chosen so let's go over this now.

5.2.1 Neighborhood Component Analysis[1]

Neighborhood Component Analysis is a non-parametric method for selecting features in which the goal is to maximize an objective function that in-effect calculates the optimal number of neighbors to use in a knn classification algorithm. In practice, by maximizing the objective function we can extract a weight vector which can be used to understand which features give us the biggest amount of insight into how to classify our data.

Bibliography

[1] Goldberger, Jacob Roweis, Sam Hinton, Geoffrey Salakhutdinov, Ruslan. (2004). Neighbourhood Components Analysis.. in Advances in Neural Information Processing Systems (NIPS 2004) Vancouver Canada: Dec.. 17.