Amanda Solis

EEG Signal Processing & Exploratory Data Analysis

Introduction

The brain's electrical activity can be recorded with electrodes placed on the scalp to produce an electroencephalograph (EEG). EEGs are most often used to provide information about brain abnormalities or in the domain of brain computer interfaces- where software and/or hardware applications are controlled solely with a user's brain activity. The data collected from an EEG recording is very noisy due to the complexity of the underlying brain activity but can be filtered and transformed to extract features of interest. There are many methods by which to "clean" EEG data and this paper demonstrates the signal processing techniques and exploratory data analyses I did on a selected raw EEG data set.

Data

The data set I explored contained multichannel EEG recordings from three different subjects. During the experiment, the subjects' brain activity was recorded with 32 electrodes at a sampling rate of 512 Hz while they did one of three tasks. The tasks were to imagine moving their left hand, imagine moving their right hand, or generate a word beginning with the same random letter. Every 15 seconds at the researcher's request a subject would randomly switch between tasks. This lasted for a total of four minutes and each subject did this three times. Since the data was recorded continuously there was no trial structure separating the three tasks from one another. Each raw data file contained approximately 122k samples and 32 features (corresponding to the 32 electrodes).

Methods

The first thing I did with the raw data was plot each electrode's voltage as a function of time. These plots showed that the voltage drifted slightly over the duration of the recording, as can be seen by the overlapping/drifting lines in figure 1 below. One cause for this could be that the metals of the electrode are reacting with the subject's skin causing very low frequency voltage changes. One way to get rid of these low frequency oscillations is to apply a high pass filter that attenuates the lower frequencies (fremoves frequencies below 0.01 Hz).

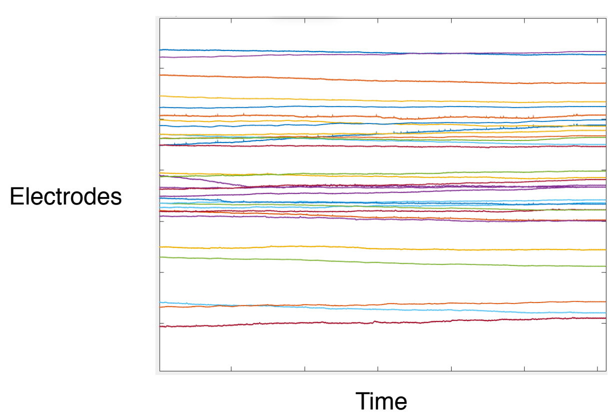

Figure 1 Voltage Drift Over Time

I also plotted the periodogram power spectral density estimate of each channel. These plots show the power of the signal distributed over different frequencies. In figure 2 we can clearly see a spike in the power at 50 Hz due to the line noise from recording in Europe. To get rid of this 50 Hz technical artifact we can apply a notch filter that removes a narrow frequency range.

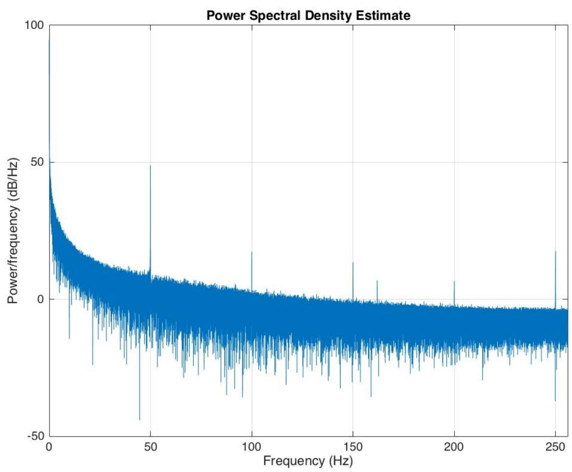

Figure 2 Line Noise at 50Hz, Electrode Fp1

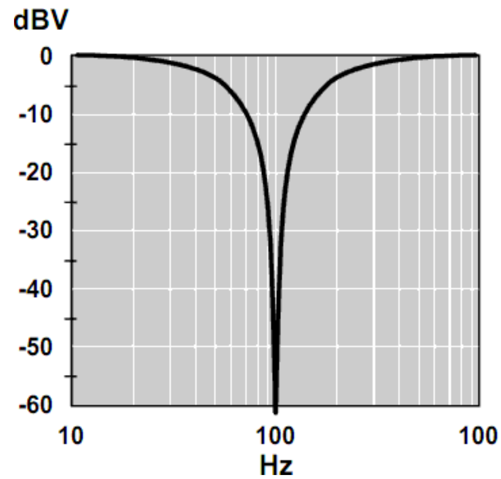

Figure 3 Notch Filter 100Hz

Though I experimented with different types of notch and high-pass filters I ended up applying only a single bandpass filter to remove both of the unwanted frequency intervals at once. I bandpass filtered the raw data between 7-30 Hz using a 20th order IIR Butterworth filter. There were many filters to choose from and many filter options (like window size, segment length) to adjust, but I chose to use the Butterworth filter with MATLAB's default parameters. I used the IIR Butterworth because it is the default filter used in a program called EEGLAB created by UCSD's own Swartz Center for Computational Neuroscience. This filter provides a smooth monotonic frequency response that is maximally flat in the passband, but sacrifices rolloff steepness for flatness (Mathworks). Figure 3 shows the filter I created in MATLAB and figure 4 shows the power spectral density estimate after filtering was applied to electrode Fp1.

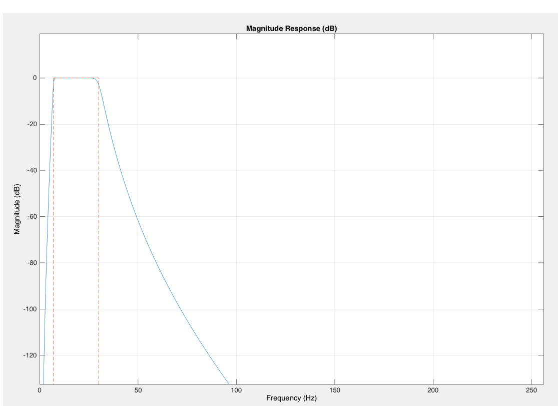

Figure 3 IIR Butterworth filter with passband 7-30Hz

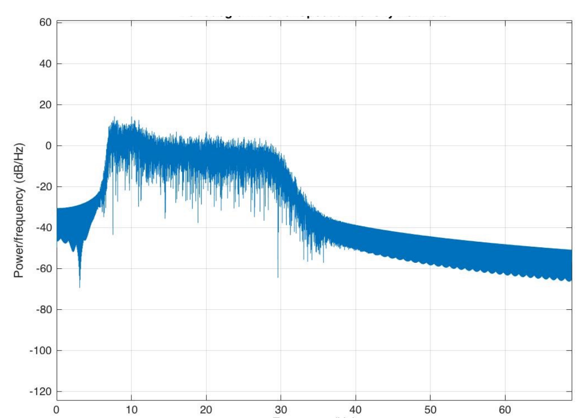

Figure 4 Power Spectral Density estimate of filtered signal, Electrode Fp1

Once the data was filtered I wanted to see how the power for different frequencies differed depending on the task the subject was performing. I divided each single data set into three smaller ones separating the three tasks. I then used a function in MATLAB called bandpower which takes the power spectral density estimates and a specified frequency interval and then returns the average power in that interval. Using this function I could find the power in a small frequency interval, say 7-12 Hz, and divide that by the total power in the 7-30 Hz interval to find the percentage of total power in 7-12 Hz, in MATLAB that looks like this for left hand imagery:

```
>L_power_subband = bandpower(psdEstim_Left(:,channel), f_Left, [7 12] ,'psd');
>L_power_total = bandpower(psdEstim_Left(:,channel), f_Left, 'psd');
>per_power_L = 100*(L_pband/L_ptot);

>per_power_L =

  44.8217
```

The input, "f_Left", is a vector of frequencies corresponding to the PSD estimates in "psdEstim_Left". These commands return the percentage of total power in the 7 Hz to 12 Hz interval. I did this in really small subband sizes (instead of intervals like 7-12 Hz, I used intervals with a width of 0.1 giving me intervals like 7.0-7.1 Hz). I then plotted the percent of power over frequency for each electrode. Figure 5 shows a large spike in channel C4 around 10 Hz when subject 1 is performing right hand imagery, whereas in Channel T8 the right hand imagery power is more broadly distributed over the frequencies.

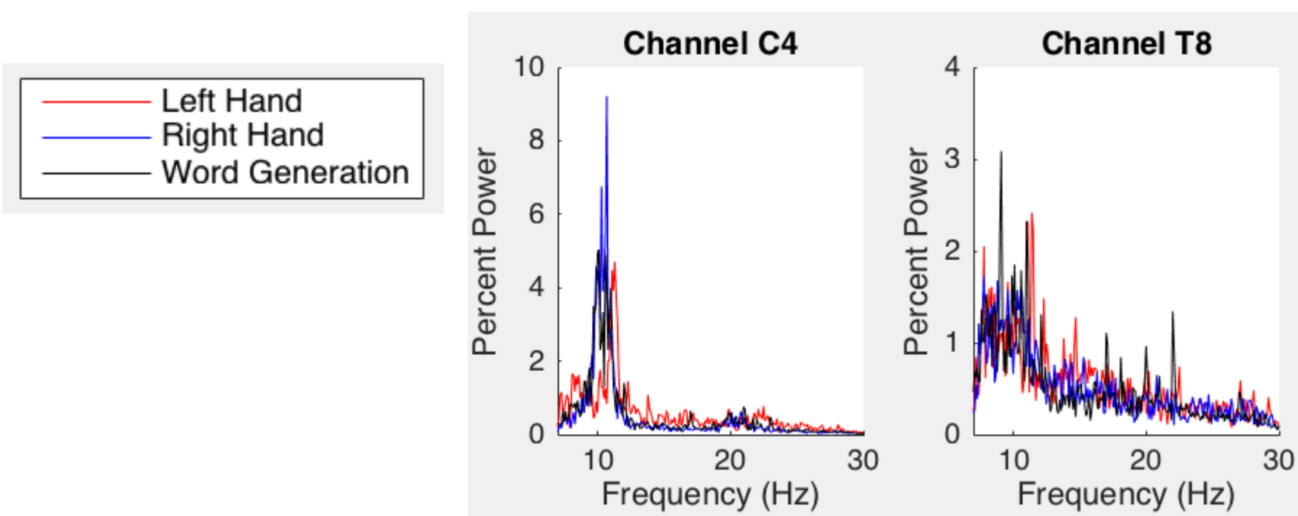

Figure 5 Percentage of power plotted against frequency for different tasks

Conclusions

My main focus was to explore some of the neural signal processing techniques I've researched by applying them to a raw EEG data set. Though I met my objective, I also developed many follow-up questions. Since I only performed the above signal processing on one subject's data from one recording session, I would have also liked to perform objective statistical analyses by using data from all three sessions of the experiment (to understand between-subject effects) and across all subjects (to explore within-subject effects). I would have also liked to have had a better understanding of how different filters and filter options can be used to distort signals.

References

Kass, Robert E., Uri T. Eden, and E. N. Brown. *Analysis of Neural Data.* Print.

Collaborative Research in Computational Neuroscience, CRCNS, **https://crcns.org/data-sets**

MATLAB, **https://www.mathworks.com/**

EEG Data Set, **http://www.bbci.de/competition/iii/**, Data Set V, Millán, J. del R.. On the need for on-line learning in brain-computer interfaces Proc. Int. Joint Conf. on Neural Networks., 2004.

EEGLAB, Swartz Center for Computational Neuroscience, **http://sccn.ucsd.edu/wiki/EEGLAB_Wiki**

Notch Filter, **http://www.learningelectronics.net/circuits/hq-notch-filter-without-close-tolerance.html**