EEG and Chaotic Time Series Analysis

Contents

Abstract

How do we build a prosthetic limb that moves according to the thoughts of a wearer? Likewise, how do we communicate instructions to a computer using thoughts alone? Both questions are apart of the developing field of Brain-Computer Interfacing (BCI).

Electroencephalography (EEG) is one of the leading and affordable types of analysis of brain activity. We utilize a number of existing methods to filter and express these data. Furthermore, we illustrate approaches towards the categorization and prediction of these data through Artificial Neural Networks (ANNs). Finally, we implement a Nonlinear Autoregressive Exogenous (NARX) model for this prediction, and analyze the dynamics of the system.

1 Introduction

The human brain is an intricate and interwoven network of communicating cells, structures, and substructures. As such, understanding and interpreting information from this network is a primary research focus of many dedicated professionals. We aim to synthesize and present current understandings as succinctly as possible, for focus on how they may be utilized. To begin, we must define our operating principals.

1.1 Electroencephalography (EEG)

Traditionally, *Electroencephalography* is the measurement of neuronal function given by the *ionic gradient* of neuronal membranes [5]; typically given in microvolts and amplified for analysis. Electrical potentials are determined by individual *neurons* consisting of four main parts: the *Dendrites* or signal receptors, *Soma* - main body of the cell, *Axon* - for relaying signals, and the *Terminal Bouton* - for transmitting signals via the synaptic gap. For the purpose of brevity, signals can be interpreted as either *Excitatory Postsynaptic Potentials* (ESP) or *Inhibitory Postsynaptic Potentials* (ISP), based on the influx of either sodium or potassium ions, respectively. The dendrites of the neuron receive both excitatory and inhibitory signals from adjacent neurons, which diffuse across the soma and trigger an *action potential* based on the either positive or negative concentration of ions in the cell. This results in the propagation of signals across the axon, resulting in a consequent signal released by the terminal bouton into the synaptic gap. These events, known as *synaptic potentials* are ultimately what are being measured by an EEG device.

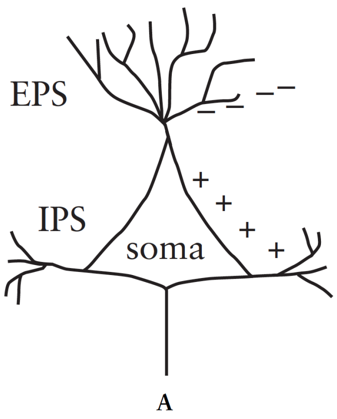

Figure 1: Neuron with inhibitory and excitatory signals - (where a is the axon)

We used the OpenBCI Ultracortex Mark IV headset for its open source design, 8 channels, and modifiable 3D-printed apparatus. The controlling Cyton board links via bluetooth interface with the OpenBCI USB Dongle, although the hardware allows for an insertable SD card for writing the data. The board samples at 250 Hz, which can be modified in the Cyton's Arduino firmware. Likewise, the scale factor of the received data is given as 0.022 35 µV, which is modifiable given the gain (default set to maximum) in the firmware or with live connection through development drivers such as the Python-MNE.

1.2 Artificial Neural Networks (ANNs)

An *Artificial Neural Network* is an information processing system based on the same principles as biological networks like the brain [3]. The consist of a set of *perceptrons* which work concurrently given activations on directed connections with each other. Much like neurons, the perceptrons send both excitatory and inhibitory signals, which influence the subsequent layers.

Formally, we express the neural network using a *directed graph*.

Definition 1. A *Directed Graph*, G, consists of a set of *vertices*, $V(G)$, and *edges* ,$E(G)$. A *directed edge* or *arc*, $e = (u, v) \in E(G)$ represents a connection from the vertex u to the vertex v. We define the *predecessors* of u, *pred*(u), as $\{v \in V(G)|(v, u) \in V(G)\}$, and the *succesors* of u, *succ*(u), as $\{v \in V(G)|(u, v) \in V(G)\}$.

Which allows us to define the neural network as,

Definition 2. A *neural network* is a directed graph, G, where the vertices are redefined as *neurons*,

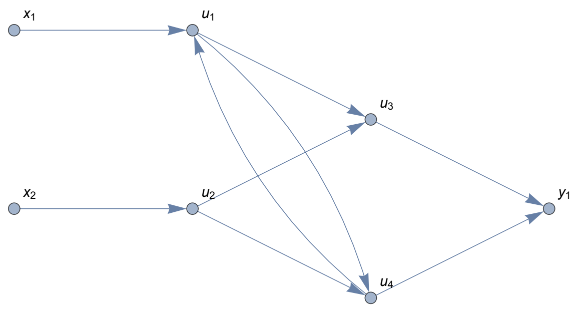

Figure 2: Example Network

$$\begin{array}{c|ccccccc}
 & x_1 & x_2 & u_1 & u_2 & u_3 & u_4 & y_1 \\
x_1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
x_2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
u_1 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\
u_2 & 0 & 0 & 0 & 0 & 3 & 3 & 0 \\
u_3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
u_4 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
y_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Figure 3: Example Network Structure

$V(G) = N(G)$, and edges redefined as *connections*, $E(G) = C(G)$. The set $N(G)$ is partitioned into $N(G) = N_{in} \cup N_{out} \cup N_{hid}$, where $N_{in} \neq \emptyset$, $N_{out} \neq \emptyset$, and $N_{hid} \cap (N_{in} \cup N_{out}) = \emptyset$.

Where each connection, uv, carries a weight, w_{uv}, and each neuron, u, possesses four quantities: network input, net_u, activation, act_u, output, out_u, and external input, ext_u. Likewise, each neuron has three activation functions: $f^u_{net} : \mathbb{R}^{2|pred(u)|+\kappa_1(u)} \to \mathbb{R}$, $f^u_{act} : \mathbb{R}^{\kappa_2(u)} \to \mathbb{R}$, and $f^u_{out} : \mathbb{R} \to \mathbb{R}$, where κ_1 and κ_2 are dependent on the number of inputs.

The *adjacency matrix* or *network structure* of G is given as $M = \{w_{ij} | i, j \in [n(G)]\}$. If G contains no self-loops or directed cyles, then G is a *feed forward* network, else G is a *recurrent* network.

2 Digital Signal Processing and Sampling

A near constant problem in the transmission of information is the inevitable presence of *noise*. While a field of study in its own right, we utilize a few methods to aid in our signal processing [1] [6] [2].

2.1 Time Domain

We receive data from the Cyton board roughly once a millisecond (as there can be a minor offset from system time to Cyton board time), the position of each node is given as,

	θ	r	name
1	-39	0.333	$F3$
2	0	0.256	Fz
3	39	0.333	$F4$
4	-90	0.256	$C3$
5	90	0	Cz
6	90	0.256	$C4$
7	-141	0.33	$P3$
8	141	0.333	$P4$

.

Eight channels give us row vectors of the form,

$$f(t) = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8),$$

followed by the translation to phase space.

$$x_j = x(t + j\tau),$$

where t is the start time, τ is our sampling rate, and j is the channel index.

2.2 Frequency Domain

Utilizing the *Discrete Fourier Transform* (DFT),

$$F_n = \sum_{k=0}^{N-1} X_k \cdot e^{-2\pi i n k/N},$$

and *Inverse Discrete Fourier Transform*,

$$f_k = \frac{1}{N}\sum_{k}^{N-1} F_n \cdot e^{2\pi i n k/N},$$

we move between time and frequency domains. It's here that we utilize our digital filter, and claim multiplication in the frequency domain is equivalent to convolution in the time domain [1], and its complexity can be reduced from $O(n^2)$ to $O(n \log n)$.

Proof. Let $f_1(t)$ and $f_2(t)$ be arbitrary vectors of $f(t)$. We define convolution as
$[f_1 * f_2](t) = \sum_{v=-\infty}^{\infty} f_1(\tau) f_2(t-\tau)$
$= \sum_{v=-\infty}^{\infty} f_1(t-\tau) f_2(\tau)$.
This implies $[f_1 * f_2](t) = \sum_{v=-\infty}^{\infty} f_1(\tau) f_2(t-\tau)$
$= \sum_{v=-\infty}^{\infty} f_1(\tau) \sum_{v=-\infty}^{\infty} F_2(v) \cdot e^{2\pi i v(t-\tau)}$
$= \sum_{v=-\infty}^{\infty} F_2(v) \sum_{v=-\infty}^{\infty} (f_1(\tau) \cdot e^{-2\pi i v \tau}) e^{2\pi i v t}$,
$= \sum_{v=\infty}^{\infty} F_2(v) F_1(v) e^{2\pi i v t}$,
which is really,
$= F_\tau^{-1}[F(\tau)F(\tau)](t)$
and so by applying F to both sides, $F[f_1 * f_2] = F(f_2)F(f_1)$

Q.E.D

(For the non-discrete convolution theorem)

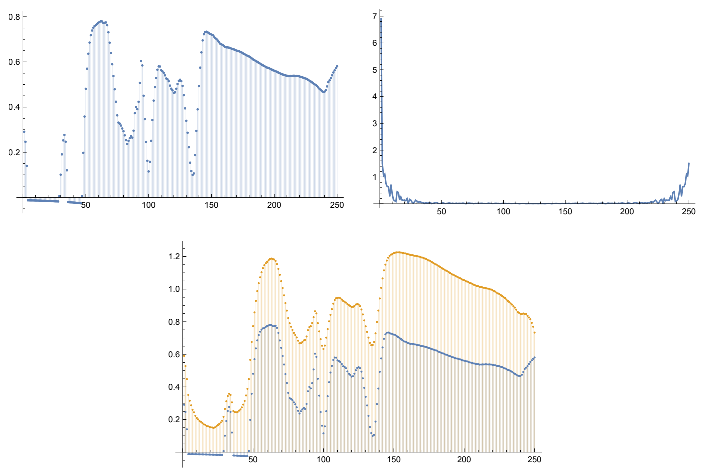

Figure 4: Top Left: Unfiltered data, Top Right: Translation to frequency domain, Bottom: Filter applied and translation back to the time domain

Besides the benefit of more efficient filtering time, frequency information combined with the channel locations allows for some interesting visualizations 5 6.

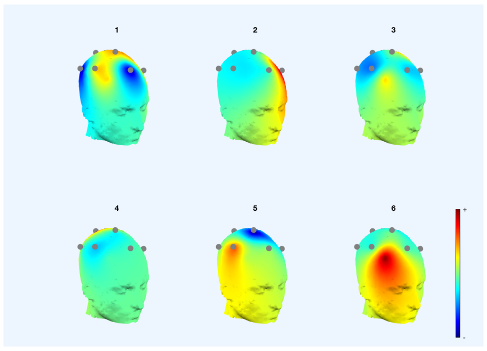

Figure 5: Head Plots showing activation

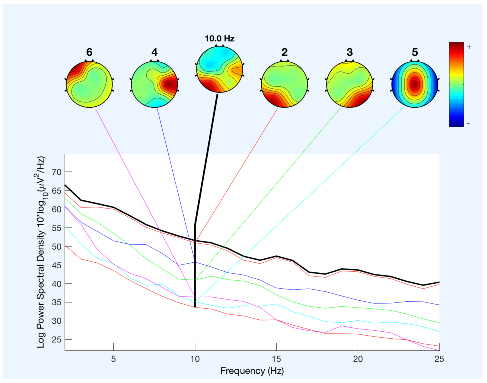

Figure 6: Map of six channels given a random 25% of the data

2.3 Time-Frequency Domain

While technically out of the scope of this project, a wavelet transform may be implemented in the future. This is due to more efficient computation time with regard to the DFT, as the computation of the Fourier matrix is $O(n \log n)$ [4] whereas the Wavelet matrix is $O(n)$. The wavelet functions can be given as,

$$\psi^{a,b}(x) = |a|^{-1/2}\,\psi\left(\frac{x-b}{a}\right)$$

$$W_\psi(f)(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty} f(t)\psi\left(\frac{t-b}{a}\right)dt,$$

for some $\psi(x)$, for example, the Haar function,

$$\psi(x) = \begin{cases} -1 & \frac{1}{2} < x < 1 \\ 1 & 0 < x < \frac{1}{2} \\ 0 & \text{else} \end{cases} .$$

Translation into the time-frequency domain affords us the ability to see potential activations at any one particular time interval. Of particular interest is the frequencies from 8 Hz − 12 Hz, or more commonly refered to as the *alpha* band [8].

3 Artifact Rejection

The presence of *artifacts* in EEG data presents a serious problem for data interpretation. A leftover of transient electromyographic signals, fluctuations in the muscles beneath the scalp are interpreted and recorded by EEG devices. This produces a notable spike in both measured potentials as well as frequencies, which can be problematic when we're looking to define features of the given inputs.

3.1 Preprocessing

Following the recording of these data, we import (see Matlab subsection) to a toolbox where we can filter and run an *independent component analysis*. This produces a maximally linearly independent set of vectors, which also smooths data spikes such as eye blinks [4].

Now that we have filtered data with minimal artifacts, we can start to consider our computational intelligence approach. Features will be found in these newly processed data sets, all that remains is to convert them to a json file format for use by our neural network. However, it's necessary to train the given network given a set of inputs and expected outputs. The implication then is that we must design an experimentation framework to record different expected outputs.

Once this is achieved, we can begin to train two ANNs against both processed and unprocessed data. The first network must be trained to extract artifacts, and return data similar to the processed data, and the second must analyze said processed data and make predictions regarding future input. Both processes must also have a predecessor data filter.

We're currently training a *Nonlinear Autoregressive Exogenous* model to accurately predict and respond to blinks, simply for our own edification [7].

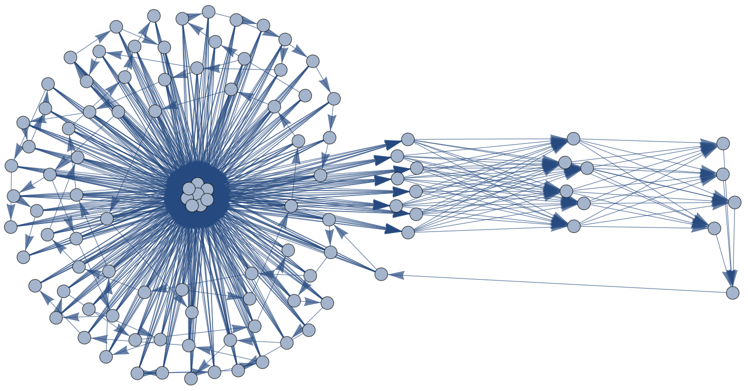

Figure 7: Our NARX

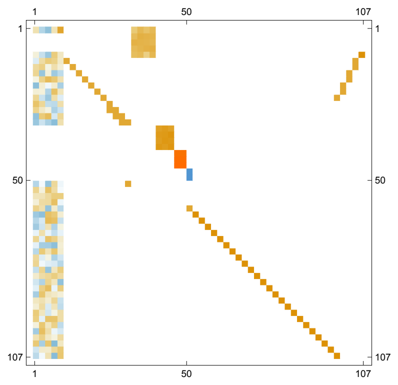

Figure 8: Network Structure

4 Implementations

For the current project code, please see this repository. We're currently working to establish a database, using MongoDB, for the storage of EEG data. Being as these data can be sensitive, we also intend to implement safe storage procedures to secure the privacy of any volunteers helping us collect data.

4.1 Matlab/Octave

This function was written to automate the preprocessing of recorded data using the OpenBCI GUI. It applies the necessary filters, followed by the independent component analysis, before outputing the data as a set file. These data must be converted back to micro volts before the ANN is trained, as well as converted to the appropriate json format.

```

function EEG =  OpenBCIDATA(file ,writefile)
```

```
    [ALLEEG EEG CURRENTSET ALLCOM] = eeglab;
    RAWDATA = dlmread(file,",");
    Data = RAWDATA(6:rows(RAWDATA),2:9);
    idx = NA(size(Data(:,:)));
    Data{:,:}(idx) = 0;
    MyArr = transpose(table2array(Data));;
    EEG=pop_importdata('setname','test','data',MyArr,'dataformat', ...
                       'array','pnts',height(Data),'chanlocs',<
    channel_location_file>,'nbchan',8, ...
                  'xmin',0,'srate',250);

    [ALLEEG EEG CURRENTSET ] = eeg_store(ALLEEG, EEG);
    EEG = pop_eegfilt( EEG, 1, 0, [], [0]);

nevents = length(EEG.event);
for index = 1 : nevents
  if ischar(EEG.event(index).type) && strcmpi(EEG.event(index).type, 'square')
    EEG.event(end+1) = EEG.event(index);
    EEG.event(end).latency = EEG.event(index).latency - 0.1*EEG.srate;
    EEG.event(end).type = 'cue'; '% Make the type of the new event '
  end;
end;

EEG = eeg_checkset(EEG, 'eventconsistency');
[ALLEEG EEG CURRENTSET] = eeg_store(ALLEEG, EEG, CURRENTSET);

[ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, CURRENTSET, 'setname', '
    Continuous EEG Data');
EEG = pop_reref( EEG, [], 'refstate',0);

EEG = pop_rmbase( EEG, [0 1000]);
[ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, CURRENTSET, 'setname', '
    Continuous EEG Data epochs', 'overwrite', 'on');


EEG = pop_runica(EEG,'chanind',[1 2 3 4 5 6 7 8]);

[ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, CURRENTSET, ...
                                        'setname', writefile, 'overwrite', 'on');

pop_saveset(EEG,'filename',writefile,'filepath',<data_path>);

[ALLEEG EEG CURRENTSET] = eeg_store(ALLEEG, EEG, CURRENTSET);
```

Listing 1: Matlab/Octave Dataset Function

5 Proposal for Future Work

The developing field of robotics/bionics promises to provide affordable, customized prosthetic equipment to those in need. We'd like to formally request the following materials in order to contribute what we can. A link to the project website can be found here.

- 1 x servo

- 1 x dyneema fishing line

- 1 x nylon fishing line

- 1 x pulleys

- 1 x sponge tape

- 1 x adhesive-tape

- 1 x safety grip tape

- 1 x velcro

- 1 x wooden dowel

- 1 x arm sleeve

- 1 x 4mm silicone sheet

- 1 x 5mm silicone sheet

In addition, we'd like to request ready access to a 3D-printer along with,

- 1 x NinjaFlex TPU filament

- 1 x MatterHackers ABS filament

And finally, the following from OpenBCI,

- 1 x wifi-shield

- 1 x Myoware-muscle-sensor

- 1 x pulse-sensor

- 1 x Cyton-Daisy Biosensing board

References

[1] Sophocles J. Orfanidis. *Introduction to signal processing*. Prentice Hall signal processing series. Prentice Hall, Englewood Cliffs, N.J, 1996.

[2] Germn Rodrguez-Bermdez and Pedro J. Garcia-Laencina. Analysis of EEG signals using non-linear dynamics and chaos: a review. *Applied Mathematics & Information Sciences*, 9(5):2309, 2015.

[3] Kruse Rudolf, Borgelt Christian, Braune Christian, and Mostaghim Sanaz. *Computational Intelligence: A Methodological Introduction*. Springer.

[4] Sarit Shwartz, Michael Zibulevsky, and Yoav Y. Schechner. ICA using kernel entropy estimation with NlogN complexity. *Lecture notes in computer science*, pages 422–429, 2004.

[5] William O Tatum. *Handbook of EEG interpretation*. Demos Medical Pub., New York [N.Y., 2013. OCLC: 841495449.

[6] J. Ulbikas, A. Cenys, and O. P. Sulimova. Chaos parameters for EEG analysis. *Nonlinear Analysis: Modelling and Control*, 3:1–8, 1998.

[7] Yan Xiu and Wei Zhang. Multivariate Chaotic Time Series Prediction Based on NARX Neural Networks.

[8] Longhao Yuan and Jianting Cao. Patients EEG Data Analysis via Spectrogram Image with a Convolution Neural Network. In Ireneusz Czarnowski, Robert J. Howlett, and Lakhmi C. Jain, editors, *Intelligent Decision Technologies 2017*, volume 72, pages 13–21. Springer International Publishing, Cham, 2018. DOI: 10.1007/978-3-319-59421-7_2.