EEG Signal Analysis: Color Recognition Using Support Vector Machine

A graduate project in partial fulfillment of the requirements

For the degree of Master of Science in Electrical Engineering

By

Jorge Heredia

May 2015
The graduate project of Jorge Heredia is approved:

______________________________________________  ______________
Professor Benjamin F. Mallard  Date

______________________________________________  ______________
Dr. Sharlene Katz  Date

______________________________________________  ______________
Dr. Xiyi Hang, Chair  Date

California State University, Northridge
# Table of Contents

SIGNATURE PAGE ........................................................................................................... ii
LIST OF TABLES ............................................................................................................... iv
LIST OF FIGURES ........................................................................................................... v
ABSTRACT ........................................................................................................................ vii

**SECTION 1: INTRODUCTION** ....................................................................................... 1
  - BCI Interface .................................................................................................................. 2
  - Purpose ............................................................................................................................. 3

**SECTION 2: ACQUISITION** ......................................................................................... 4
  - Acquisition device .......................................................................................................... 4
    - Hardware ....................................................................................................................... 5
    - Software ...................................................................................................................... 7
  - Artifacts ............................................................................................................................ 8
  - Precautions .................................................................................................................... 12

**SECTION 3: FILTERING** ............................................................................................ 13
  - Surface Laplacian ......................................................................................................... 13
  - Power Line Interference ............................................................................................... 14

**SECTION 4: DATA TRANSFORMATION** ................................................................... 15
  - Frequency Transformation ............................................................................................. 15
  - Feature Extraction ......................................................................................................... 15
    - What are features? ....................................................................................................... 15
    - Feature Reduction ....................................................................................................... 16

**SECTION 5: MACHINE LEARNING** .......................................................................... 22
  - What is machine learning? ............................................................................................ 22
    - Support Vector Machines .......................................................................................... 23

**ANALYSIS** .................................................................................................................. 26
  - Test Setup ..................................................................................................................... 26

**CONCLUSION** ............................................................................................................ 49

**REFERENCES** ............................................................................................................. 50

**APPENDIX: MATLAB CODE** .................................................................................. 51
**LIST OF TABLES**

Table 1: Emotiv EPOC Specifications [4] .............................................................................. 5
Table 2: Internal Artifacts ........................................................................................................ 12
Table 3: External Artifacts ....................................................................................................... 12
Table 4: Channel Neighbors used for Laplacian filtering ....................................................... 14
Table 5: Eigenvector CEOD percentages ............................................................................... 19
Table 6: EEG Frequency Ranges ............................................................................................ 35
Table 7: CEOD values for a sample of the actual data ............................................................. 43
Table 8: Classification Results ............................................................................................... 45
LIST OF FIGURES

Figure 1: Human Brain [2] ........................................................................................................ 2
Figure 2: Emotiv EPOC headset [4] ........................................................................................... 6
Figure 3: International 10-20 locations [5] ................................................................................. 6
Figure 4: Emotiv Testbench Time Domain ............................................................................... 7
Figure 5: Emotiv Testbench Frequency Domain ....................................................................... 8
Figure 6: Blinking Eyes .............................................................................................................. 9
Figure 7: Swallowing Saliva ..................................................................................................... 9
Figure 8: Clenching Jaw Muscles ............................................................................................. 10
Figure 9: Electrode Popping On and Off .................................................................................. 10
Figure 10: Turning Left and then Right ..................................................................................... 11
Figure 11: Movement Around the Headset ............................................................................. 11
Figure 12: Small Laplacian and Large Laplacian Relevant Channel Locations ...................... 13
Figure 13: Surface Laplacian Filter Channel Example .............................................................. 14
Figure 14: 3D Feature Space showing Red and Blue ................................................................. 18
Figure 15: Eigenvector results after performing PCA ............................................................... 19
Figure 16: Dataset on new axis ................................................................................................. 20
Figure 17: 2D reconstruction of original data .......................................................................... 21
Figure 19: Hyper plane separating two classes in the feature space ....................................... 24
Figure 20: Hyper plane in optimal position ............................................................................ 25
Figure 21: Raw Data AF3 ........................................................................................................ 27
Figure 22 Raw Data AF4 ........................................................................................................... 28
Figure 23 Raw Data F3 ............................................................................................................ 28
Figure 24 Raw Data F4 ............................................................................................................ 28
Figure 25 Raw Data F7 ............................................................................................................ 29
Figure 26 Raw Data F8 ............................................................................................................ 29
Figure 27 Raw Data FC5 ......................................................................................................... 30
Figure 28 Raw Data FC6 ......................................................................................................... 30
Figure 29 Raw Data T7 ........................................................................................................... 31
Figure 30 Raw Data T8 ........................................................................................................... 31
Figure 31 Raw Data P7 .......................................................................................................... 32
Figure 32 Raw Data P8 .......................................................................................................... 32
Figure 33 Raw Data O1 .......................................................................................................... 33
Figure 34 Raw Data O2 .......................................................................................................... 33
Figure 35: Frequency Response of 60Hz Notch Filter ............................................................ 34
Figure 36: Filtered and Unfiltered Spectra .............................................................................. 35
Figure 37: AF3 Average Power Spectrum ............................................................................. 36
Figure 38: AF4 Average Power Spectrum ............................................................................. 36
Figure 39: F3 Average Power Spectrum ................................................................................ 37
Figure 40: F4 Average Power Spectrum ................................................................................ 37
Figure 41: F7 Average Power Spectrum ................................................................................ 38
Figure 42: F8 Average Power Spectrum ................................................................................ 38
Figure 43: FC5 Average Power Spectrum ................................................................. 39
Figure 44: FC6 Average Power Spectrum ................................................................. 39
Figure 45: T7 Average Power Spectrum ................................................................. 40
Figure 46: T8 Average Power Spectrum ................................................................. 40
Figure 47: P7 Average Power Spectrum ................................................................. 41
Figure 48: P8 Average Power Spectrum ................................................................. 41
Figure 49: O1 Average Power Spectrum ................................................................. 42
Figure 50: O2 Average Power Spectrum ................................................................. 42
ABSTRACT

EEG Signal Analysis: Color Recognition Using Support Vector Machine

By
Jorge Heredia

Master of Science in Electrical Engineering

This project is a biomedical application of pattern recognition. The objective of this project is to be able to accurately predict which of two colors, red or blue, is seen by the subject by analyzing the EEG signals and patterns present for each color. Techniques utilized in this paper include the Surface Laplacian spatial filter, power line rejection notch filter, spectral analysis using the FFT, PCA for feature extraction, and SVM for machine learning. The data acquisition device used for this project was the Emotiv EPOC headset, which follows the international 10-20 channel placement locations and has a sampling rate of 128Hz. The project was successfully able to predict the colors.
SECTION 1: INTRODUCTION

An electroencephalogram (EEG) is a measurement of signal that carries information about what the brain is doing. Different situations, conditions and thoughts going on in a person’s mind will all produce a different EEG. They are measured by placing electrodes over different locations of the patient’s scalp. Some applications of EEGs are examining what parts of the brain are active for certain activities, comparing mental conditions between a healthy person and someone with a brain disease, or for controlling prosthetics. Section 2 of this paper shows some sample EEGs from common activities performed by the individual.

There are 5 main frequency bands in which most of the observed EEG signals for humans lie: Delta (0.5Hz-4Hz), Theta (4Hz-7.5Hz), Alpha (8Hz-13Hz), Beta (14Hz-26Hz), Gamma (30Hz-45Hz). The delta waves are more prominent in situations when a person is experiencing deep sleep. The wave disappears when the person opens their eyes again. Theta waves are prevalent when the person is drowsy. This impaired awareness state might be found in cases when the person is meditating. Alpha waves are most active when the person is in an idle or waiting state. They appear when the person closes their eyes and also when they’re not concentrating on anything. This state can be reduced by surprising the person and making them break out of the idle state. Beta waves show up when the person is engaged in an activity where they are thinking or concentrating on something. This will show up when the person is trying to solve a math problem. Gamma waves believed to tie the sensory information to movement. It shows up before a movement is performed. [1]
The brain is separated into three main parts: cerebrum, cerebellum, and brainstem as shown in Figure 1. The functions controlled in the cerebrum include initiating actions, processing of active and conscious sensations, problem solving, emotions, and behavior. The functions controlled in the cerebellum are voluntary muscle control and maintaining balance. The functions in the brainstem are the involuntary controls that go on behind the scenes in the keeping the human body running. These functions include keeping the heart beating and respiration. [1]

![Figure 1: Human Brain](image)

**BCI Interface**

Brain computer interfaces, as the name implies are systems that connect the mind to the computer. The brain acts as the source of the signals, which are then sent to the computer for processing. The processing consists of, but not limited to, filtering, spectral
analysis, control signal extraction and machine learning. The widest applications of BCI are those involving the use of the brain to provide control signals to move the arms of paralytics and prosthetics for amputees. BCI serves as an alternate path for the brain signals to communicate around the damaged parts of the body and bring the information to the device under control. [1]

**Purpose**

The purpose of this project is to be able to predict the color that I see by analyzing the electroencephalogram (EEG) signals that occurs in my brain at the moment that the color is being observed. The two colors that were observed were red and blue. The procedure that was followed can be summarized as first acquiring the EEG data while observing a color, running this raw data through a series of filters, extracting features from the filtered data, running the features (for both red and blue) through a machine learning algorithm to develop a predication model, and finally testing the prediction model and recording its accuracy for different implementations of the learning algorithm.
SECTION 2: ACQUISITION

Acquisition device

There are three main methods to measure the electrical signals generated by the brain: implant, ECoG, and EEG. They are listed in the order from the most invasive to the least invasive. As expected, the signal quality is greater with the more invasive procedure but it poses a greater risk to the patient. The implant is a direct connection between the electrode and the brain of the subject, where the electrode is actually penetrating the cerebral cortex. ECoG is another implant that is inside the skull, but this does not penetrate the brain. Instead, it is an electrode net that is placed over the cerebral cortex surface. Unlike the previous two methods that were discussed, EEG does not require surgically implanting any foreign object inside the body, making it the only method that provides a non-invasive measurement. [3] An EEG acquisition device was chosen for this project since it is the safest to use on myself and the signal quality is sufficient for my needs. This device is the Emotiv EPOC. The specifications for this headset are listed below in Table 1.
<table>
<thead>
<tr>
<th></th>
<th>EEG HEADSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels</td>
<td>14 (plus CMS/DRL references, P3/P4 locations)</td>
</tr>
<tr>
<td>Channel names</td>
<td>AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4</td>
</tr>
<tr>
<td>Sampling method</td>
<td>Sequential sampling. Single ADC</td>
</tr>
<tr>
<td>Sampling rate</td>
<td>128 SPS (2048 Hz internal)</td>
</tr>
<tr>
<td>Resolution</td>
<td>14 bits 1 LSB = 0.51μV (16 bit ADC, 2 bits instrumental noise floor discarded)</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>0.2 - 45Hz, digital notch filters at 50Hz and 60Hz</td>
</tr>
<tr>
<td>Filtering</td>
<td>Built in digital 5th order Sinc filter</td>
</tr>
<tr>
<td>Dynamic range</td>
<td>8400μV (pp)</td>
</tr>
<tr>
<td>Coupling mode</td>
<td>AC coupled</td>
</tr>
<tr>
<td>Connectivity</td>
<td>Proprietary wireless, 2.4GHz band</td>
</tr>
<tr>
<td>Power</td>
<td>LiPoly</td>
</tr>
<tr>
<td>Battery life (typical)</td>
<td>12 hours</td>
</tr>
<tr>
<td>Impedance Measurement</td>
<td>Real-time contact quality using patented system</td>
</tr>
</tbody>
</table>

Table 1: Emotiv EPOC Specifications [4]

**Hardware**

The Emotiv EPOC headset is shown below in Figure 2. As the specifications mention, it follows the International 10-20 locations for the channel locations on the scalp. The 10-20 gets its name from the electrode placement at distances based on 10% and 20% of the total distance between certain landmarks on the skull. This can be described better with the illustration in Figure 3. [5]
Figure 2: Emotiv EPOC headset [4]

Figure 3: International 10-20 locations [5]

The sampling rate of 128 SPS (or Hz) limits the acquisition to handle any signals below 64 Hz without any aliasing effects showing up. Since the bandwidth of the headset is from 0.2 Hz – 45 Hz, there should be little to no aliasing effects from anything above 64 Hz.
Software

Emotiv provides software that captures the raw data from the headset and displays it on-screen. This data is updated in real-time and the option exists to view it in either the time domain or the frequency domain. Both Figure 4 and Figure 5 show the user interface provided by the software. Both are recordings when there is no user input, i.e., the headset is not on the user’s head. The top-left image shows where the channels should be located on the head. The color of the circles holds information on the connection quality. Green indicates the best connection, followed by yellow, orange, red, and finally black (no connection). The software returns the data in .edf format, but there is an option to convert this to .csv.

Figure 4: Emotiv Testbench Time Domain
Figure 5: Emotiv Testbench Frequency Domain

Artifacts

When measuring EEGs, the headset is not only picking up the activity that is occurring in the brain. The brain is measuring the effects of anything that induces a potential difference and this could be coming from either inside or outside the body. Examples of some internal artifacts include the subject moving their eyes (Figure 6), moving their tongue around inside their mouth (Figure 7), muscle activity (Figure 8), ECG signals, breathing and sweat. Examples of some external artifacts include 60 Hz power line noise, the electrode losing contact with the scalp (Figure 9), turning to a direction (Figure 10), and movement around the subject (Figure 11) [6].
Figure 6: Blinking Eyes

Figure 7: Swallowing Saliva
Figure 8: Clenching Jaw Muscles

Figure 9: Electrode Popping On and Off
Figure 10: Turning Left and then Right

Figure 11: Movement Around the Headset
Precautions

In order to mitigate the effect of the artifacts mentioned above, certain precautions were taken. The artifacts and the precautions associated with each have been organized in the table below.

<table>
<thead>
<tr>
<th>Internal Artifact</th>
<th>Precaution taken</th>
<th>Extra note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving Eyes</td>
<td>Tried to avoid blinking excessively and sudden eye movement</td>
<td>Blinking was unavoidable, some eye movement was present</td>
</tr>
<tr>
<td>Moving Tongue</td>
<td>Kept tongue as motionless as possible, pressed up to the top of my mouth</td>
<td>Moved tongue to swallow saliva several times during the recording</td>
</tr>
<tr>
<td>Muscle Activity</td>
<td>Tried to remain in a relaxed position without clenching teeth</td>
<td>Keeping tongue motionless might produce artifact here</td>
</tr>
<tr>
<td>ECG Activity</td>
<td>None</td>
<td>No real control on this</td>
</tr>
<tr>
<td>Breathing</td>
<td>None</td>
<td>No real control on this</td>
</tr>
<tr>
<td>Sweat</td>
<td>Washed head and hair before test</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 2: Internal Artifacts

<table>
<thead>
<tr>
<th>External Artifact</th>
<th>Precautions taken</th>
<th>Extra note</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 Hz power line noise</td>
<td>Plan on using a 60 Hz Notch filter</td>
<td>None</td>
</tr>
<tr>
<td>Electrode disconnection</td>
<td>Avoid sudden head movements</td>
<td>None</td>
</tr>
<tr>
<td>Movement around the subject</td>
<td>Performed the measurements in a room alone, with no moving parts around</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3: External Artifacts
SECTION 3: FILTERING

Surface Laplacian

The Surface Laplacian filter is used to reduce the amount of interference that is caused by adjacent channels. The filtering was performed as described by the equation below. Two filters of this type are the Small Laplacian and the Large Laplacian. The Small Laplacian subtracts a scaled contribution from the channels that are directly adjacent to the channel of interest, while the Large Laplacian subtracts the channel that is two steps away from the channel of interest [7]. The setup in Figure 12 shows the channel to be measured in red and the adjacent channels in yellow.

\[ x_{\text{filt}}[n] = x_{\text{orig}}[n] - \frac{1}{N} \sum_{\text{adj}=1}^{N} x_{\text{adj}}[n] \quad [8] \]

Since the headset that was used did not have always have adjacent channels available that lie one location apart (according to the 10-20 system), the Laplacian used was a mixture of both the Small and the Large Laplacians. The diagram below shows an example for a set of yellow channels that are considered adjacent to one red channel. Here the channel that will be filtered is F3 (in red) and the adjacent channels which are used in the filtering of F3 are AF3, FC5 and F4 (in yellow).
The list of the channels and their adjacent contributors are shown in Table 4.

<table>
<thead>
<tr>
<th>Left Side</th>
<th>Right Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td>Adjacent CHs</td>
</tr>
<tr>
<td>AF3</td>
<td>F7, F3, FC5</td>
</tr>
<tr>
<td>F7</td>
<td>AF3, F3, FC5</td>
</tr>
<tr>
<td>F3</td>
<td>AF3, FC5, F4</td>
</tr>
<tr>
<td>FC5</td>
<td>F7, F3, T7</td>
</tr>
<tr>
<td>T7</td>
<td>FC5, P7</td>
</tr>
<tr>
<td>P7</td>
<td>T7, O1</td>
</tr>
<tr>
<td>O1</td>
<td>P7, O2</td>
</tr>
</tbody>
</table>

Table 4: Channel Neighbors used for Laplacian filtering

Power Line Interference

The next set of filtering involved removal of the interference from the power line. This was nothing more than filtering the raw time domain data using a notch filter. The notch filter was centered at 60 Hz.
SECTION 4: DATA TRANSFORMATION

Frequency Transformation

The time domain data was converted to the frequency domain so that any underlying periodic components could be identified. This could identify which of the frequency bands are active during the exposure to the visual stimulus of a color. The code used to generate the power plots in the following figures has been added to the Appendix. It was calculated by first taking the average of every time domain sample and then moving to the frequency domain using the equation below.

\[ PSF(f) = \frac{1}{F_s \cdot N} |FFT(x[n])|^2 \]

Feature Extraction

What are features?

A feature is a noticeable characteristic, pattern or trend in data that can be used to describe some characteristic of the data. They are useful in describing data because in many cases, a large set of data can be described using less. For example, a large set of data containing points that form a line can contain millions of points, but the same data can be represented more compactly by only noting the slope and an offset. In this case, the process of feature extraction is the process which is performed to get the slope and the bias from the data. In order to make the data processing more efficient, it is better to process the most relevant features of the data instead of the raw data itself to reduce the number of computations.
Feature Reduction

The data at this point is still large. There are 1,000 points of data per channel, summing up to 14,000 data points when looking at all of the channels. In order to avoid using every point of data, it is convenient to extract only the features that are important in describing the pattern that is observed in the data. The approach taken for this purpose was to apply the principal component analysis (PCA) algorithm to the frequency data and use the result of PCA as the features.

Principal Component Analysis can be used for data compression, feature extraction and filtering. In data compression, PCA serves as a method to apply dimension reduction by discarding certain dimensions that might hold small amount of data in exchange for occupying less memory. Feature extraction uses the fact that PCA provides a transformation matrix based on the original data along the areas of highest variability, which can be used as features. PCA’s application to filtering can be used to filter EOG, ECG, and EMG. The procedure for the PCA analysis is illustrated below [9].

1. Subtract the channel mean from each of the channel data.
   \[ x_i = x_i - E[x_i] \quad \forall \ i \in \text{Channels} \]

2. Calculate the channel to channel covariance matrix.
   \[ C = Cov[X] \]
   where \( X = [ x_0 \ \cdots \ x_n ] \)

3. Solve for the eigenvectors of the covariance matrix to obtain the direction of the principal components.
   \[ det(C - \lambda I) = 0 \]
This method returns a matrix whose values reflect how the channels are interacting with each other along the newly generated unit eigenvector-defined axis. The matrix, from left to right, goes from most important vector to the least important. Importance, here, refers to how much a vector characterizes the data. It can then be reduced in size by removing the less important eigenvectors which lie on the rightmost columns. Unfortunately, the high dimensionality of the data that is used for this project cannot be visually inspected to show this procedure at work, so the three dimensional visual example below should make this entire method clearer.

Consider the data below and assume that the pattern for blue and red are scattered in the feature space like in Figure 14. Note that this data is not from an actual data source and the pattern has been exaggerated. It is only provided to visualize the method used. The figure shows two very different slopes that have evident pattern differences to distinguish between the two classes. Both of the diagrams show the same datasets, with the only difference being the point of view to show the three dimensionality clearer.
Figure 14: 3D Feature Space showing Red and Blue

After performing PCA on the dataset from above, the matrix of eigenvectors was obtained. These eigenvectors are all unit vectors that are each orthonormal to each other. The eigenvector with the largest eigenvalue will point along the direction with the highest variation in the data, while the remaining eigenvectors will sequentially (from most important to least important) point towards directions at a progressively decreasing magnitude of variation. Since there are three features, three eigenvectors are obtained (Figure 15). By looking at where these eigenvectors point, we can determine which class the data corresponds to.
As mentioned earlier, the importance of each eigenvector is determined by analyzing its eigenvalue, which is also obtained by performing PCA. For the dataset describing the red data, the Contribution to Expressing the Original Data (CEOD) that can be expressed by each eigenvector (EV) along that vector-defined-axis is shown below.

<table>
<thead>
<tr>
<th>Eigenvector</th>
<th>Eigenvalue</th>
<th>CEOD</th>
<th>Cumulative CEOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV1</td>
<td>598.6623</td>
<td>93.81%</td>
<td>93.81%</td>
</tr>
<tr>
<td>EV2</td>
<td>25.5099</td>
<td>4%</td>
<td>97.81%</td>
</tr>
<tr>
<td>EV3</td>
<td>14.0018</td>
<td>2.19%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 5: Eigenvector CEOD percentages

The results clearly show that the majority of the contribution is in the direction pointed to by the most important eigenvector (EV1). Since the cumulative CEOD is 100% when all the eigenvectors are used to represent the data, then the data should be completely reconstructable. The only difference being an apparent rotation from the original feature axis to the new axis that is built from the eigenvectors. The reconstructed dataset is obtained by multiplying the eigenvector matrix by the original
data to rotate the original data. Both the red and the blue data were multiplied by the red eigenvector matrix and are plotted on the new axis (Figure 16).

![Figure 16: Dataset on new axis](image)

Since the CEOD from the first two EVs can express about 98% of the entire data, discarding the last EV should only result in a 2% loss of the variability from the original data. By discarding EV3 and attempting to reconstruct the original data, the 3rd dimension is lost (along with some data due to the variability that is lost) and the data can be described in only two dimensions. This results in less features to deal with at the price of some data loss. The two dimensional reconstruction is shown in Figure 17.
Figure 17: 2D reconstruction of original data
What is machine learning?

Machine learning is the process that is taken when teaching some system to learn from data. Pattern recognition is an application of machine learning which can be applied to detect human faces, characters, and distinguish between cancerous and non-cancerous cells by having a computer learn and generate a model based on certain patterns in a dataset. Learning is used when the following statements hold:

1. A pattern exists
2. The pattern cannot be expressed mathematically
3. Data exists on the subject to be learned

Without the first statement, there is no benefit to trying to learn. If there is no pattern, then there is no way to predict how some event will lead to another event. If the second statement doesn’t hold, then learning may not be the most efficient method to find a model for the pattern. Without the last statement, there is no way to learn since there is no data to learn from. A machine learning algorithm only needs data to exist for it to run, but if no pattern exists, it will try to learn and fail to provide a correct model. If there is a mathematical expression that can be used to model an existing pattern, then that would provide the most efficient way arrive at the model. This mathematical relationship and the parameters, however, are rarely known exactly, so learning may be used to find that relation, especially for data with high dimensionality. [10]

Machine learning can be divided into a classification problem or a regression problem. The goal of classification is to categorize data into separate states, thereby applying a label to it so that it belongs to a certain class. It can be further classified into
various subcategories: supervised learning and unsupervised learning. Supervised learning uses data that contains both the features of the samples and the classifications associated with each sample. Unsupervised learning does not have the labels that show which class the samples belong to. The goal of this type of learning is to group the data into sections, where the data can be shown to belong to one class even though the type of class is not known. Regression in machine learning is the same idea as regression in mathematics, where a model that describes the pattern in the data is generated by some initial data and then the output of a data point not in the initial data is predicted based on the model.

**Support Vector Machines**

Support vector machines (SVM) is a tool of machine learning that follows a mathematical based algorithm to learn a pattern that exists between two classes. This classification problem uses red and blue as the two labels for the two classes of data. The model that is used to classify the data is known as the objective function and in its simplest form it takes the form of the linear model shown below.

\[ f(x) = w^T x - b \]

The intuitive meaning of the weight vector, \( w \), can be interpreted as showing how significant a particular feature is in making the decision, such that a larger weight means a larger influence on the decision. It scales its inputs by multiplying the input feature vector by a weight vector and a bias term, \( b \), is used to provide a shift. The geometrical meaning of the equation is that it describes a hyper plane (hyper when dealing with more than three dimensions) that separates between two classes. The way this hyper plane is used when applying the hard margin technique is by having
everything on one side of the hyper plane classified as belonging to one class, while everything on the other side belongs to the other class (Figure 18). Soft margin is similar except that some members of opposite classes may lie on the other side of the hyper plane. Mathematically, this is found by checking the sign of the objective function. If it is negative, it is on one side of the hyper plane and if it is positive, it is on the other side [11].

![Figure 18: Hyper plane separating two classes in the feature space](image)

The goal of support vector machines is to find the best hyper plane that maximizes the distance between the hyper plane and some points that are called the support vectors. The support vectors are the points that provide a reference to measure the distance from the hyper plane (Figure 19). Finding the optimal values for both the weight vector and the bias term can be achieved by maximizing the margin between the hyper plane and the support vectors. After some math, maximizing the margin translates to minimizing the quantity shown below with respect to the weight and bias parameters.
Besides the basic linear combination objective function mentioned earlier, a kernel transformation could be applied to the input data to increase the dimensionality of the data when it cannot be separated by a hyper plane in the original feature space. The objective function in this case becomes the equation below, where $\phi(x)$ is the kernel used to transform the original data into a higher dimensional space.

$$f(x) = w^T \phi(x) - b$$
ANALYSIS

Test Setup

Before starting the measurements, the headset needs to be charged and properly set up. The setup begins by applying saline solution to the pads; this will provide the interface between the pads on the headset and the subject’s scalp. After this has been applied, the electrodes/pad assembly is attached to the headset by rotating the assembly it locks. Next, the headset is put on the subject’s head. The headset is adjusted so that the two electrodes in the front, AF3 and AF4, are above the subject’s eyebrows by the distance of three fingers when the hand is placed in a horizontal orientation right above the eyebrows. AF3 did not have an electrode that made contact with the subject’s scalp, so it was floating. Also, place the reference electrodes, P3 and P4 (also known as CMS and DRL), touching the bone behind both ears.

After the headset is set up the experimental setup is configured in the following manner. A chair was set facing a TV screen which was set to display a single color at a time. With the headset on, the subject sat on the chair facing the colored TV screen and the all other sources of light were out of sight. This was done for one color at a time with a 5 minute rest period with all lights on. The subject sat looking at the screen for about 13 minutes, which was enough to collect above 100,000 samples at a sampling rate of 128 Hz.

The function called csv2mat was used to convert the csv data file acquired from the headset into a MATLAB object named Jorge, remove the DC component and divide the entire dataset into 100 smaller samples of 1000 points each.
The raw data that was collected for each channel for the time is presented below in Figure 20 through Figure 33. It has been truncated so that it is exactly 100,000 points of data, which is enough to provide 100 samples, each containing 1000 points.

Figure 20: Raw Data AF3
Figure 21 Raw Data AF4

Figure 22 Raw Data F3

Figure 23 Raw Data F4
Figure 24 Raw Data F7

Figure 25 Raw Data F8
Figure 26 Raw Data FC5

Figure 27 Raw Data FC6
Figure 28 Raw Data T7

Figure 29 Raw Data T8
Figure 30 Raw Data P7

Figure 31 Raw Data P8
Figure 32 Raw Data O1

Figure 33 Raw Data O2
This data was filtered first using the Laplacian filter. The function SL_Filter was used to apply the Laplacian filter for every channel. Then the data was put through an IIR notch filter to remove the 60Hz component. The frequency response for this particular notch filter is shown below in Figure 34, along with sample of an unfiltered and a filtered spectra of channel AF3 when the electrode was not touching the subject’s head (i.e. it was floating above the head) in Figure 35.

Figure 34: Frequency Response of 60Hz Notch Filter
The peak in power that existed at 60Hz has been brought down to -80dB. This is much lower than anything else in the spectrum and since the value is so close to zero, reducing it any further wouldn’t have much use.

After filtering the data, the PSF was calculated for each of the channels and both the red and the blue were plotted side by side for comparison. Using the script `PlotMeanSpectra`. The results of calling this function are presented below. For convenience, the frequency ranges of the waves are shown in the table below.

<table>
<thead>
<tr>
<th>%</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta</td>
<td>0.5 Hz -  4 Hz</td>
</tr>
<tr>
<td>Theta</td>
<td>4 Hz -  7.5 Hz</td>
</tr>
<tr>
<td>Alpha</td>
<td>8 Hz - 13 Hz</td>
</tr>
<tr>
<td>Beta</td>
<td>14 Hz - 26 Hz</td>
</tr>
<tr>
<td>Gamma</td>
<td>30 Hz - 45 Hz</td>
</tr>
</tbody>
</table>

Table 6: EEG Frequency Ranges
Figure 36: AF3 Average Power Spectrum

Figure 37: AF4 Average Power Spectrum
Figure 38: F3 Average Power Spectrum

Figure 39: F4 Average Power Spectrum
Figure 40: F7 Average Power Spectrum

Figure 41: F8 Average Power Spectrum
Figure 42: FC5 Average Power Spectrum

Figure 43: FC6 Average Power Spectrum
Figure 44: T7 Average Power Spectrum

Figure 45: T8 Average Power Spectrum
Figure 46: P7 Average Power Spectrum

Figure 47: P8 Average Power Spectrum
There is definitely a noticeable difference in how the brain responds when it is processing a visual stimulus from red and blue. Blue tends to be lower in power for most of the channels. The most notable exceptions to this are channels FC5 and F8 where blue
is higher in power throughout the spectrum. The following frequencies are seen consistently throughout the red samples: 2Hz, 3Hz, 5Hz and 7Hz. This shows that the Delta, Theta, and Alpha bands are consistently used. Channels F7 and F8 handle signals that show some activity in the Gamma Band.

This data was run through the PCA process mentioned in the theory, where the initial features (analogous to x1, x2, and x3) were the frequency components for all 14 channels after performing an FFT on the data filtered by the Surface Laplacian and the Power Line filters.

The final result from the code above results in a new vector that has either all or a reduced amount of the eigenvectors which will be used as the features of the data in the machine learning. Performing a CEOD calculation for the eigenvector matrix on one of the 100 samples resulted in the numbers below Table 7.

<table>
<thead>
<tr>
<th>Eigenvector</th>
<th>Eigenvalue</th>
<th>CEOD</th>
<th>Cumulative CEOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV1</td>
<td>4.1378 E6</td>
<td>67.1479%</td>
<td>67.15%</td>
</tr>
<tr>
<td>EV2</td>
<td>0.4926 E6</td>
<td>7.9943%</td>
<td>75.1422%</td>
</tr>
<tr>
<td>EV3</td>
<td>0.3805 E6</td>
<td>6.1743%</td>
<td>81.3165%</td>
</tr>
<tr>
<td>EV4</td>
<td>0.3304 E6</td>
<td>5.3616%</td>
<td>86.6781%</td>
</tr>
<tr>
<td>EV5</td>
<td>0.1676 E6</td>
<td>2.7195%</td>
<td>89.3976%</td>
</tr>
<tr>
<td>EV6</td>
<td>0.1491 E6</td>
<td>2.4203%</td>
<td>91.8179%</td>
</tr>
<tr>
<td>EV7</td>
<td>0.1209 E6</td>
<td>1.9619%</td>
<td>93.7798%</td>
</tr>
<tr>
<td>EV8</td>
<td>0.1112 E6</td>
<td>1.8046%</td>
<td>95.5844%</td>
</tr>
<tr>
<td>EV9</td>
<td>0.0852 E6</td>
<td>1.3824%</td>
<td>96.9667%</td>
</tr>
<tr>
<td>EV10</td>
<td>0.0642 E6</td>
<td>1.0414%</td>
<td>98.0081%</td>
</tr>
<tr>
<td>EV11</td>
<td>0.0512 E6</td>
<td>0.8305%</td>
<td>98.8386%</td>
</tr>
<tr>
<td>EV12</td>
<td>0.0354 E6</td>
<td>0.5748%</td>
<td>99.4134%</td>
</tr>
<tr>
<td>EV13</td>
<td>0.0316 E6</td>
<td>0.5136%</td>
<td>99.9270%</td>
</tr>
<tr>
<td>EV14</td>
<td>0.0045 E6</td>
<td>0.0730%</td>
<td>100.0000%</td>
</tr>
</tbody>
</table>

*Table 7: CEOD values for a sample of the actual data*
The PCA process used above generated the features needed by the SVM learning and classification algorithm. This allowed the pattern recognition model to learn which eigenvectors described the mental state of viewing the color red vs blue. The training was performed using the first set of processed data, which was three times larger than the testing data. The training size of the data was 75 samples (1000 data points per sample), while the testing data was 25 samples. The results for using different parameters (kernels, number of features, and c values) is shown below.

<table>
<thead>
<tr>
<th>Kernel Used</th>
<th>PCA components used</th>
<th>Training Accuracy</th>
<th>Testing Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linear</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>Quadratic</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>100</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>100</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
The kernel that performed best for the conditions tested above was the polynomial third order kernel, which had almost perfect classification across the feature.

<table>
<thead>
<tr>
<th>Polynomial (Third order)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MLP</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>89.33</td>
<td>78</td>
</tr>
<tr>
<td>2</td>
<td>78</td>
<td>92</td>
</tr>
<tr>
<td>3</td>
<td>97.33</td>
<td>94</td>
</tr>
<tr>
<td>4</td>
<td>98.67</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>99.33</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>98.67</td>
<td>98</td>
</tr>
<tr>
<td>7</td>
<td>98</td>
<td>92</td>
</tr>
<tr>
<td>8</td>
<td>98</td>
<td>86</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>96</td>
</tr>
<tr>
<td>10</td>
<td>99.33</td>
<td>90</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>92</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>88</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RBF</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>72</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>52</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 8: Classification Results
number sweep, while the RBF had the worst performance. The number of PCA components used translates to 196 features and 14 features used for the case when it was set to 14 and 1, respectively.

<table>
<thead>
<tr>
<th>C</th>
<th>Kernel Used</th>
<th>PCA Comp’s used</th>
<th>Training Accuracy</th>
<th>Testing Accuracy</th>
<th>PCA Comp’s used</th>
<th>Training Accuracy</th>
<th>Testing Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00 E-03</td>
<td>rbf</td>
<td>1</td>
<td>100 %</td>
<td>58 %</td>
<td>14</td>
<td>100 %</td>
<td>50 %</td>
</tr>
<tr>
<td>1.00 E-02</td>
<td>rbf</td>
<td>1</td>
<td>100 %</td>
<td>58 %</td>
<td>14</td>
<td>100 %</td>
<td>50 %</td>
</tr>
<tr>
<td>1.00 E-01</td>
<td>rbf</td>
<td>1</td>
<td>100 %</td>
<td>72 %</td>
<td>14</td>
<td>100 %</td>
<td>50 %</td>
</tr>
<tr>
<td>1.00 E+00</td>
<td>rbf</td>
<td>1</td>
<td>100 %</td>
<td>74 %</td>
<td>14</td>
<td>100 %</td>
<td>50 %</td>
</tr>
<tr>
<td>1.00 E+01</td>
<td>rbf</td>
<td>1</td>
<td>100 %</td>
<td>74 %</td>
<td>14</td>
<td>100 %</td>
<td>50 %</td>
</tr>
<tr>
<td>1.00 E+02</td>
<td>rbf</td>
<td>1</td>
<td>100 %</td>
<td>74 %</td>
<td>14</td>
<td>100 %</td>
<td>50 %</td>
</tr>
<tr>
<td>1.00 E+03</td>
<td>rbf</td>
<td>1</td>
<td>100 %</td>
<td>74 %</td>
<td>14</td>
<td>100 %</td>
<td>50 %</td>
</tr>
<tr>
<td>1.00 E-03</td>
<td>mlp</td>
<td>1</td>
<td>97.33 %</td>
<td>84 %</td>
<td>14</td>
<td>50 %</td>
<td>50 %</td>
</tr>
<tr>
<td>1.00 E-02</td>
<td>mlp</td>
<td>1</td>
<td>98 %</td>
<td>82 %</td>
<td>14</td>
<td>50 %</td>
<td>50 %</td>
</tr>
<tr>
<td>1.00 E-01</td>
<td>mlp</td>
<td>1</td>
<td>94 %</td>
<td>78 %</td>
<td>14</td>
<td>94 %</td>
<td>76 %</td>
</tr>
<tr>
<td>1.00 E+00</td>
<td>mlp</td>
<td>1</td>
<td>89.33 %</td>
<td>78 %</td>
<td>14</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>1.00 E+01</td>
<td>mlp</td>
<td>1</td>
<td>88.67 %</td>
<td>80 %</td>
<td>14</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>1.00 E+02</td>
<td>mlp</td>
<td>1</td>
<td>89.33 %</td>
<td>82 %</td>
<td>14</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>1.00 E+03</td>
<td>mlp</td>
<td>1</td>
<td>88.67 %</td>
<td>80 %</td>
<td>14</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>1.00 E-03</td>
<td>polynomial</td>
<td>1</td>
<td>100 %</td>
<td>94 %</td>
<td>14</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>1.00 E-02</td>
<td>polynomial</td>
<td>1</td>
<td>100 %</td>
<td>94 %</td>
<td>14</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>1.00 E-01</td>
<td>polynomial</td>
<td>1</td>
<td>100 %</td>
<td>94 %</td>
<td>14</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>1.00 E+00</td>
<td>polynomial</td>
<td>1</td>
<td>100 %</td>
<td>94 %</td>
<td>14</td>
<td>100 %</td>
<td>100 %</td>
</tr>
</tbody>
</table>
Another parametric sweep was performed by sweeping the c value, which is a measure of how much to punish for errors during training. Higher c values generally lead toward overfitting, while lower values may lead to more underfitting of the data, but that does not prove to be the case here. With this data, the trend for higher c...
generalizes the model even further, which is evident by the increase in testing accuracy with higher c values.
CONCLUSION

The application of SVM to classify the color that I saw by analyzing the EEG signals associated with the colors was successful. The feature extraction using PCA provided features that were both relevant to describing the colors and accurate enough that even when reduced to only 14 features, the SVM model was able to classify them correctly.
REFERENCES


clear; close all; clc

% The ranges below were obtained from "EEG Signal Processing" pg 10-11
% Delta  0.5 Hz - 4 Hz
% Theta  4 Hz - 7.5 Hz
% Alpha  8 Hz - 13 Hz
% Beta   14 Hz - 26 Hz
% Gamma  30 Hz - 45 Hz

% Possible kernels: linear, quadratic, mlp, polynomial, rbf
kernel = 'rbf';
SampleSize = 1000;
NumPosTrainSamples = 75;
NumNegTrainSamples = 75;
NumPosTestSamples = 25;
NumNegTestSamples = 25;
TotalPos = NumPosTrainSamples + NumPosTestSamples;
TotalNeg = NumNegTrainSamples + NumNegTestSamples;
ClassNamePos = 'Red';
ClassNameNeg = 'Blue';
FeatureMatrix_Train = [];
FeatureMatrix_Test = [];
TrainClassifications = [];
TestClassifications = [];
NumEigVects = 1;  % Number of eigenvectors to use from PCA

%% Import the data
DirName = 'C:\Users\Jorge\Desktop\EEG data\';
filename_Pos = strcat(DirName, ClassNamePos, '.csv');
filename_Neg = strcat(DirName, ClassNameNeg, '.csv');
Jorge_P = csv2mat(filename_Pos, true, TotalPos, SampleSize);
Jorge_N = csv2mat(filename_Neg, true, TotalNeg, SampleSize);

%% Define the file to import
for FileIndex = 0:NumPosTrainSamples + NumNegTrainSamples - 1;
    % Generate the classifications for the data.
    NameIndex = mod(FileIndex, NumPosTrainSamples) + 1;
    if FileIndex < NumPosTrainSamples
        Classification = ClassNamePos;
        Jorge = Jorge_P;
    else
        Classification = ClassNameNeg;
    end
end
Jorge=Jorge_N;
end
TrainClassifications=[TrainClassifications;Classification(1)];

%% Process the data

% Surface Laplacian: Filter the data
Jorge_SL=SL_Filter(Jorge,NameIndex);

% Power Line Noise: Filter the data
wo = 60/(128/2);  bw = 0.1;[b,a] = iirnotch(wo,bw);
Jorge_SL_PL=filter_obj(b,a,Jorge_SL);

% Frequency domain conversion of the time domain data
Jorge_FFT=FFT_MultipleNodes(Jorge_SL_PL);

Data_freq=[Jorge_FFT.AF3;Jorge_FFT.AF4;Jorge_FFT.F3;Jorge_FFT.F4; ... 
Jorge_FFT.F7;Jorge_FFT.F8;Jorge_FFT.FC5;Jorge_FFT.FC6; ... 
Jorge_FFT.O1;Jorge_FFT.O2;Jorge_FFT.P7;Jorge_FFT.P8; ... 
Jorge_FFT.T7;Jorge_FFT.T8]';

% Feature reduction using PCA
PCa_data_freq=pca(Data_freq);
PCa_data_freq=PCa_data_freq(1:NumEigVects,:);  %Discard less important
Seqi
PCa_data_freq=PCa_data_freq(:)';
FeatureMatrix_Train=[FeatureMatrix_Train;PCa_data_freq];
end

%% Perform the training
% Use SVM to learn from the data
SVM_Model=svmtrain(FeatureMatrix_Train,TrainClassifications,'kernel_function',ke
rnel);
% Use the SVM model to check the Training Accuracy
SVM_Results=svmclassify(SVM_Model,FeatureMatrix_Train);

NumOfMissClass = sum(SVM_Results~=TrainClassifications);
MisClassRate = NumOfMissClass/(length(SVM_Results));
CorrectClassRate = 1-MisClassRate;
display(['Correct Classification(Training Samples):' num2str(CorrectClassRate 
*100) '%']);

%% Use the learning model to predict from new data
for FileIndex=0:NumPosTestSamples+NumNegTestSamples-1;
    % Generate the filenames for the data to be imported.
NameIndex = mod(FileIndex, NumPosTestSamples) + 1;
if FileIndex < NumPosTestSamples
    Classification = ClassNamePos;
    NameIndex = NameIndex + NumPosTrainSamples;
    Jorge = Jorge_P;
else
    Classification = ClassNameNeg;
    NameIndex = NameIndex + NumNegTrainSamples;
    Jorge = Jorge_N;
end
TestClassifications = [TestClassifications; Classification(1)];

%% Process the data
% Surface Laplacian: Filter the data
Jorge_SL = SL_Filter(Jorge, NameIndex);

% Power Line Filter: Filter the data
Jorge_SL_PL = filter_obj(b, a, Jorge_SL);

% Frequency domain conversion of the time domain data
Jorge_FFT = FFT_MultipleNodes(Jorge_SL_PL);

Data_freq = [Jorge_FFT.AF3; Jorge_FFT.AF4; Jorge_FFT.F3; Jorge_FFT.F4;...
             Jorge_FFT.F7; Jorge_FFT.F8; Jorge_FFT.FC5; Jorge_FFT.FC6;...
             Jorge_FFT.O1; Jorge_FFT.O2; Jorge_FFT.P7; Jorge_FFT.P8;...
             Jorge_FFT.T7; Jorge_FFT.T8];

% Feature reduction using PCA
PCA_data_freq = pca(Data_freq);
PCA_data_freq = PCA_data_freq(1:NumEigVects,:);  % Discard less important
% eigen vectors
FeatureMatrix_Test = [FeatureMatrix_Test; PCA_data_freq];
end

SVM_Results = svmclassify(SVM_Model, FeatureMatrix_Test);

NumOfMissClass = sum(SVM_Results ~= TestClassifications);
MisClassRate = NumOfMissClass / (length(SVM_Results));
CorrectClassRate = 1 - MisClassRate;
display(['Correct Classification (Testing Samples): ' num2str(CorrectClassRate * 100) ' %']);

% Plots the average power spectral density for each channel

% The ranges below were obtained from "EEG Signal Processing" pg 10-11
% Delta      0.5 Hz - 4 Hz
% Theta      4 Hz - 7.5 Hz
% Alpha      8 Hz - 13 Hz
% Beta       14 Hz - 26 Hz
% Gamma      30 Hz - 45 Hz

Fs=128;
N=1000;

AF3_Red_mean=mean(Jorge_P.AF3);     AF3_Red_mean_FFT=(1/(Fs*N)) * abs(fft(AF3_Red_mean)).^2;
AF4_Red_mean=mean(Jorge_P.AF4);     AF4_Red_mean_FFT=(1/(Fs*N)) * abs(fft(AF4_Red_mean)).^2;
F3_Red_mean=mean(Jorge_P.F3);       F3_Red_mean_FFT=(1/(Fs*N)) * abs(fft(F3_Red_mean)).^2;
F4_Red_mean=mean(Jorge_P.F4);       F4_Red_mean_FFT=(1/(Fs*N)) * abs(fft(F4_Red_mean)).^2;
F7_Red_mean=mean(Jorge_P.F7);       F7_Red_mean_FFT=(1/(Fs*N)) * abs(fft(F7_Red_mean)).^2;
F8_Red_mean=mean(Jorge_P.F8);       F8_Red_mean_FFT=(1/(Fs*N)) * abs(fft(F8_Red_mean)).^2;
FC5_Red_mean=mean(Jorge_P.FC5);     FC5_Red_mean_FFT=(1/(Fs*N)) * abs(fft(FC5_Red_mean)).^2;
FC6_Red_mean=mean(Jorge_P.FC6);     FC6_Red_mean_FFT=(1/(Fs*N)) * abs(fft(FC6_Red_mean)).^2;
T7_Red_mean=mean(Jorge_P.T7);       T7_Red_mean_FFT=(1/(Fs*N)) * abs(fft(T7_Red_mean)).^2;
T8_Red_mean=mean(Jorge_P.T8);       T8_Red_mean_FFT=(1/(Fs*N)) * abs(fft(T8_Red_mean)).^2;
P7_Red_mean=mean(Jorge_P.P7);       P7_Red_mean_FFT=(1/(Fs*N)) * abs(fft(P7_Red_mean)).^2;
P8_Red_mean=mean(Jorge_P.P8);       P8_Red_mean_FFT=(1/(Fs*N)) * abs(fft(P8_Red_mean)).^2;
O1_Red_mean=mean(Jorge_P.O1);       O1_Red_mean_FFT=(1/(Fs*N)) * abs(fft(O1_Red_mean)).^2;
O2_Red_mean=mean(Jorge_P.O2);       O2_Red_mean_FFT=(1/(Fs*N)) * abs(fft(O2_Red_mean)).^2;

AF3_Blue_mean=mean(Jorge_N.AF3);    AF3_Blue_mean_FFT=(1/(Fs*N)) * abs(fft(AF3_Blue_mean)).^2;
AF4_Blue_mean=mean(Jorge_N.AF4);    AF4_Blue_mean_FFT=(1/(Fs*N)) * abs(fft(AF4_Blue_mean)).^2;
F3_Blue_mean=mean(Jorge_N.F3);      F3_Blue_mean_FFT=(1/(Fs*N)) * abs(fft(F3_Blue_mean)).^2;
F4_Blue_mean=mean(Jorge_N.F4);      F4_Blue_mean_FFT=(1/(Fs*N)) * abs(fft(F4_Blue_mean)).^2;
F7_Blue_mean = mean(Jorge_N.F7); F7_Blue_mean_FFT = (1/(Fs*N)) * abs(fft(F7_Blue_mean)).^2;
F8_Blue_mean = mean(Jorge_N.F8); F8_Blue_mean_FFT = (1/(Fs*N)) * abs(fft(F8_Blue_mean)).^2;
FC5_Blue_mean = mean(Jorge_N.FC5); FC5_Blue_mean_FFT = (1/(Fs*N)) * abs(fft(FC5_Blue_mean)).^2;
FC6_Blue_mean = mean(Jorge_N.FC6); FC6_Blue_mean_FFT = (1/(Fs*N)) * abs(fft(FC6_Blue_mean)).^2;
T7_Blue_mean = mean(Jorge_N.T7); T7_Blue_mean_FFT = (1/(Fs*N)) * abs(fft(T7_Blue_mean)).^2;
T8_Blue_mean = mean(Jorge_N.T8); T8_Blue_mean_FFT = (1/(Fs*N)) * abs(fft(T8_Blue_mean)).^2;
P7_Blue_mean = mean(Jorge_N.P7); P7_Blue_mean_FFT = (1/(Fs*N)) * abs(fft(P7_Blue_mean)).^2;
P8_Blue_mean = mean(Jorge_N.P8); P8_BLUE_mean_FFT = (1/(Fs*N)) * abs(fft(P8_Blue_mean)).^2;
O1_Blue_mean = mean(Jorge_N.O1); O1_Blue_mean_FFT = (1/(Fs*N)) * abs(fft(O1_Blue_mean)).^2;
O2_Blue_mean = mean(Jorge_N.O2); O2_Blue_mean_FFT = (1/(Fs*N)) * abs(fft(O2_Blue_mean)).^2;

FreqAxis = linspace(0,64,SampleSize/2);
figure;plot(FreqAxis,AF3_Red_mean_FFT(1:SampleSize/2),'r',FreqAxis,AF3_Blue_mean_FFT(1:SampleSize/2),'b'),title('AF3');xlabel('Frequency(Hz)');ylabel('Power/Frequency (uV^2/Hz)')
figure;plot(FreqAxis,AF4_Red_mean_FFT(1:SampleSize/2),'r',FreqAxis,AF4_Blue_mean_FFT(1:SampleSize/2),'b'),title('AF4');xlabel('Frequency(Hz)');ylabel('Power/Frequency (uV^2/Hz)')
figure;plot(FreqAxis,F3_Red_mean_FFT(1:SampleSize/2),'r',FreqAxis,F3_Blue_mean_FFT(1:SampleSize/2),'b'),title('F3');xlabel('Frequency(Hz)');ylabel('Power/Frequency (uV^2/Hz)')
figure;plot(FreqAxis,F4_Red_mean_FFT(1:SampleSize/2),'r',FreqAxis,F4_Blue_mean_FFT(1:SampleSize/2),'b'),title('F4');xlabel('Frequency(Hz)');ylabel('Power/Frequency (uV^2/Hz)')
figure;plot(FreqAxis,F7_Red_mean_FFT(1:SampleSize/2),'r',FreqAxis,F7_Blue_mean_FFT(1:SampleSize/2),'b'),title('F7');xlabel('Frequency(Hz)');ylabel('Power/Frequency (uV^2/Hz)')
figure;plot(FreqAxis,F8_Red_mean_FFT(1:SampleSize/2),'r',FreqAxis,F8_Blue_mean_FFT(1:SampleSize/2),'b'),title('F8');xlabel('Frequency(Hz)');ylabel('Power/Frequency (uV^2/Hz)')
figure;plot(FreqAxis,FC5_Red_mean_FFT(1:SampleSize/2),'r',FreqAxis,FC5_Blue_mean_FFT(1:SampleSize/2),'b'),title('FC5');xlabel('Frequency(Hz)');ylabel('Power/Frequency (uV^2/Hz)')
figure;plot(FreqAxis,FC6_Red_mean_FFT(1:SampleSize/2),'r',FreqAxis,FC6_Blue_mean_FFT(1:SampleSize/2),'b'),title('FC6');xlabel('Frequency(Hz)');ylabel('Power/Frequency (uV^2/Hz)')
function [ EEG_FFT_data ] = FFT_MultipleNodes( EEG_object )
% Take FFT of every channel
EEG_FFT_data.Fs=EEG_object.SamplingRate;
EEG_FFT_data.AF3=abs(fft(EEG_object.AF3));
EEG_FFT_data.F7=abs(fft(EEG_object.F7));
EEG_FFT_data.F3=abs(fft(EEG_object.F3));
EEG_FFT_data.FC5=abs(fft(EEG_object.FC5));
EEG_FFT_data.T7=abs(fft(EEG_object.T7));
EEG_FFT_data.P7=abs(fft(EEG_object.P7));
EEG_FFT_data.O1=abs(fft(EEG_object.O1));
EEG_FFT_data.O2=abs(fft(EEG_object.O2));
EEG_FFT_data.P8=abs(fft(EEG_object.P8));
EEG_FFT_data.T8=abs(fft(EEG_object.T8));
EEG_FFT_data.FC6=abs(fft(EEG_object.FC6));
EEG_FFT_data.F4=abs(fft(EEG_object.F4));
EEG_FFT_data.F8=abs(fft(EEG_object.F8));
EEG_FFT_data.AF4=abs(fft(EEG_object.AF4));
end

function [ Object_SL ] = SL_Filter( Object, Index )
% Surface Laplacian: Filter the data
Object_SL.SamplingRate=Object.SamplingRate;
Object_SL.AF3=Object.AF3(Index,:)-(1/3)*(Object.F3(Index,:)+Object.F7(Index,:)+Object.FC5(Index,:));
Object_SL.F7=Object.F7(Index,:)-(1/3)*(Object.F3(Index,:)+Object.AF3(Index,:)+Object.FC5(Index,:));
Object_SL.F3=Object.F3(Index,:)-(1/3)*(Object.AF3(Index,:)+Object.F4(Index,:)+Object.FC5(Index,:));
Object_SL.FC5=Object.FC5(Index,:)-(1/3)*(Object.T7(Index,:)+Object.F7(Index,:)+Object.FC5(Index,:));
Object_SL.T7=Object.T7(Index,:)-(1/2)*(Object.FC5(Index,:)+Object.P7(Index,:));
Object_SL.P7=Object.P7(Index,:)-(1/2)*(Object.T7(Index,:)+Object.P7(Index,:));
Object_SL.O1=Object.O1(Index,:)-(1/2)*(Object.P7(Index,:)+Object.O2(Index,:));
Object_SL.O2=Object.O2(Index,:)-(1/2)*(Object.O1(Index,:)+Object.P8(Index,:));
Object_SL.P8=Object.P8(Index,:)-(1/2)*(Object.O2(Index,:)+Object.T8(Index,:));
Object_SL.T8=Object.T8(Index,:)-(1/2)*(Object.FC6(Index,:)+Object.P8(Index,:));
Object_SL.FC6=Object.FC6(Index,:)-(1/3)*(Object.T8(Index,:)+Object.F8(Index,:)+Object.FC6(Index,:));
Object_SL.F4=Object.F4(Index,:)-(1/3)*(Object.AF4(Index,:)+Object.F3(Index,:)+Object.FC6(Index,:));
Object_SL.F8=Object.F8(Index,:)-(1/3)*(Object.F4(Index,:)+Object.AF4(Index,:)+Object.FC6(Index,:));
Object_SL.AF4=Object.AF4(Index,:)-(1/3)*(Object.F4(Index,:)+Object.F8[Index,:)+Object.FC6[Index,:]);
end

function [ Object_filt ] = filter_obj(b,a, Object )
% Filter the data
Object_filt.SamplingRate=Object.SamplingRate;
Object_filt.AF3=filter(b,a,Object.AF3);
Object_filt.F7=filter(b,a,Object.F7);
Object_filt.F3=filter(b,a,Object.F3);
Object_filt.FC5=filter(b,a,Object.FC5);
Object_filt.T7=filter(b,a,Object.T7);
Object_filt.P7=filter(b,a,Object.P7);
Object_filt.O1=filter(b,a,Object.O1);
Object_filt.O2=filter(b,a,Object.O2);
Object_filt.P8=filter(b,a,Object.P8);
Object_filt.T8=filter(b,a,Object.T8);
Object_filt.FC6=filter(b,a,Object.FC6);
Object_filt.F4=filter(b,a,Object.F4);
Object_filt.F8=filter(b,a,Object.F8);
Object_filt.AF4=filter(b,a,Object.AF4);
end

%Written by Jorge Heredia 01-30-2014

function [ data ] = csv2mat( filename,IsDCRemoved,NumSamples,SampleSize)
%CSV2MAT This function takes a file with .csv extension as an input and
%creates a structure that contains the channel information.
% This function assumes that the data that is obtained from the Emotiv
% Workbench, such that it has been tailored to that specific labeling and
% presenting of the data.
M=NumSamples;
N=SampleSize;

% Creates a data structure which contains both the numeric data and the text data
DataStruct=importdata(filename);
% Extract the data into separate variables
NumData=DataStruct.data;
TextData=DataStruct.textdata;
% TextData returns the labels together, so it is split into a string array
TextArray=strsplit(char(TextData),','); %Comma delimit
temp=strsplit(char(TextArray(1)),':');
data.TestName=temp(2);
temp=strsplit(char(TextArray(3)),':');
data.SamplingRate=temp(2);
temp=strsplit(char(TextArray(4)),':');
data.SubjectName=temp(2);
temp=strsplit(char(TextArray(6)),':');
data.Chan=temp(2);
temp=strsplit(char(TextArray(7)),':');
data.Units=temp(2);
TextArray=strsplit(char(TextArray(5))); %Space delimit
temp=strsplit(char(TextArray(1)),':');
TextArray(1)=temp(2);
% Retrieve the number of elements in TextArray
[Rows,Cols]=size(TextArray);

% Creates a structure which contains the labeled data
for i = 1:Cols
    CurrentData=['data.' char(TextArray(i))];
    % Append the labeled data
    eval([CurrentData ' = NumData(:,i);']);
    if i>=3 && i<=31
        if IsDCRemoved
            eval([CurrentData ' = ' CurrentData ' - mean(' CurrentData ');']);
            eval([CurrentData ' = ' CurrentData '(1:N*M);']);
            eval([CurrentData ' = reshape(' CurrentData ',[M,N]);']);
        end
    end
end

function [ data ] = csv2mat( filename,IsDCRemoved,NumSamples,SampleSize)
%CSV2MAT This function takes a file with .csv extension as an input and
% creates a structure that contains
% This function assumes that the data that is obtained from the Emotiv
% Workbench, such that it has been tailored to that specific labeling and
% presenting of the data.
M=NumSamples;
N=SampleSize;
% Creates a data structure which contains both the numeric data and the text data
DataStruct=importdata(filename);
% Extract the data into separate variables
NumData=DataStruct.data;
TextData=DataStruct.textdata;
% TextData returns the labels together, so it is split into a string array
TextArray=strsplit(char(TextData),','); % Comma delimit
temp=strsplit(char(TextArray(1)),':');
data.TestName=temp(2);
temp=strsplit(char(TextArray(3)),':');
data.SamplingRate=temp(2);
temp=strsplit(char(TextArray(4)),':');
data.SubjectName=temp(2);
temp=strsplit(char(TextArray(6)),':');
data.Chan=temp(2);
temp=strsplit(char(TextArray(7)),':');
data.Units=temp(2);
TextArray=strsplit(char(TextArray(5))); % Space delimit
temp=strsplit(char(TextArray(1)),':');
TextArray(1)=temp(2);
% Retrieve the number of elements in TextArray
[Rows,Cols]=size(TextArray);

% Creates a structure which contains the labeled data
for i = 1:Cols
    CurrentData=['data.' char(TextArray(i))];
    % Append the labeled data
    eval([CurrentData ' = NumData(:,i);']);
    if i>=3 && i<=31
        if IsDCRemoved
            eval([CurrentData ' = ' CurrentData ' - mean(' CurrentData ');']);
            eval([CurrentData ' = ' CurrentData '(1:N*M);']);
            eval([CurrentData ' = reshape(' CurrentData ',[M,N]);']);
        end
    end
end
end