Working Title

Dale Markowitz

April 12, 2015

Abstract

Sustaining ones attention is a perpetually difficult tasks for humans, not only because our minds tend to wander but also because we lose attention unpredictably, often not noticing our own attentional lapses until long after they have begun. In this experiment, we explore how EEG-derived neurofeedback provided to participants in real-time can help humans to notice not only to notice when they have lost attention, but also create better mental representations of the task at hand. Participants responded to a task in which they were instructed to attend to pictures of faces and ignore superimposed pictures of places, or attend to pictures of places and ignore faces, while receiving real-time neurofeedback from a wireless EEG headset. We provided participants neurofeedback derived from how face-like or place-like their EEG data appeared when categorized by a classifier. We found that the classifiability of each participants EEG data predicted how will he/she performed on a face/place task without feedback, and that later providing participants with neurofeedback was correlated with both higher task accuracy post-feedback as well as higher classifiability post-feedback.

0.1 Introduction

Its a classic experiment: balance on one foot with your eyes opened, and then balance on one foot with your eyes closed. The first is easy, the second is impossible. Why? Because our brains are experts in control theory. In the first case, we stand on one foot starting with our best guess of the correct posture and position we should take to maintain balance. Meanwhile, our eyes observe how closely our actual position matches our intended position, and dictates to our bodies how to adjust our stances to bring ourselves closer to balance. When we close our eyes, we remove a channel of immediate feedback on how our actions effect our balance, and hence balancing becomes all the more challenging. Feedback is clearly important in helping us successfully achieve our goalsboth the literal, immediate feedback we use in tasks like balancing and the more subtle type of feedback we might look for from a final exam or from a midyear review. But what if we could use feedback to help ourselves improves the most fundamental determiner of our actionsour thoughts? In this thesis, we explore how humans can learn to improve their attention by receiving real-time feedback on their internal attentional states. We use an inexpensive and nonintrusive EEG device to measure the fluctuating voltages along participants scalps, and interpret this data using machine learning techniques to construct a metric of participants attention. We then provide participants real-time feedback on how well they are attending to their given tasks by making the tasks harder or easier if the participants are attending weakly or strongly, respectively. Brain-Computer Interfaces (BCIs) for attention training are not newin fact, they have lived

a rich life as a therapy for ADHD of undetermined efficacybut we believe that our use of machine learning techniques to give participants feedback determined directly by the contents of their thoughts can make attention training much more effective than before.

0.2 Background

Neurofeedback was first shown to be effective when, in 1985, Joe Kamiya at the University of Chicago taught an adult to alter his brain-wave frequencies [9].

0.2.1 Machine Learning for EEG/fMRI Data

Traditional fMRI and EEG analyses have focused on understanding the relationships between spatial brain regions or voltage frequencies and function. On the other hand, machine learning techniques that have been highly fruitful in domains including object, handwriting, and speech recognition system, discover relationships between input data and its categories by fitting a model from a given model class to data in a way that minimizes the error of that model on the data. As a result, machine learning techniques can find representations and mappings of large, high-dimensional data sets that are more subtle than what a human being could predict by examining the data through traditional techniques alone. The application of machine learning techniques to fMRI dataoften called multi-voxel pattern analysis or MVPAhas thus been highly fruitful [11]. For example, Kamitani et al. [8] used MVPA to understand how line orientation is represented in the vi-

sual cortex, a phenomenon which occurs at cortical columns at the sub-voxel scale. Using MVPA (particularly support vector machines), they were able to recognize brain activity associated with orientation of gratings, even though this phenomenon occurs beyond the resolution of fMRI, by taking advantage of the patterns of voxel activation spread throughout the visual cortex. Machine learning techniques have also proved useful for EEG data analysis. Several studies use support vector machines to map EEG signals to prosthetic control (See Kens paper citation 6,7,8,9,10,11).

0.2.2 Real-Time EEG BCI's

Several studies use support vector machines to map EEG signals to prosthetic control (See Kens paper citation 6,7,8,9,10,11).

0.2.3 EEG Signals and Attention

Many studies have found evidence that oscillation frequencies of EEG voltages relate to attention. Ray et al [15] studied EEG signals that reflect attention to internal and external tasks (recognizing faces versus mental arithmetic, for example), and found that alpha activity was greater in the internal versus external attentional tasks.

0.2.4 Attention Training

EEG neurofeedback training for sustained attention has been of particular interest in serving as a treatment for ADHD. A 2012 review [9] of 14 such treatments found that the feedback was "probably efficacious" but found that most experiments suffered from design flaws. Most studies examined

theta/beta frequencies with a single electrode placed at Cz according to the 10-20 electrode system, and gave participants neurofeedback through a video game. The theoretical motivation for such analyses comes from the fact that many patients with ADHD have shown stronger power in the theta and slow-wave frequencies than controls, and less beta power in the central and frontal regions. Indeed, several studies [3] have found that EEG analyses of these frequency ranges can predict the presence of ADHD in patients with accuracies above 85%. A limitation of such vigilance-based attention analyses is that it does not distinguish between a participant being focused in general, and being focused on a target task at hand.

0.3 Contributions

Our experiment is inspired largely by deBettencourt et al. [4], who conducted a similar study using fMRI rather than EEG data. In both experiments, participants receive feedback related to how confidently a given classifier classifies the participants fMRI or EEG data as being focused on the target category. Past attentional training experiments have used alpha power as a metric for attention, but we find this approach problematic because the feedback relates only to general vigilance, and has no knowledge of to what subject the participants vigilance is attending. Our feedback, on the other hind, is directly tied to the content of the participants thoughts.

0.4 Experiment Setup

0.4.1 Definitions

Trial	a 50 second long interval in which a single face/place pair is displayed to the participant
Non-Lure Trial	a go trial, in which the correct response for the participant is to respond (i.e. click). These trials make up 90 percent of all trials.
Lure Trial	a no-go trial in which the correct response is not to respond (no click). These trials make up 10 percent of all trials.
Block	a set of 50 trials during which participants are instructed to attend to only faces or to only places
Pre-Training Block	a block in which the participant is not wearing the Emotiv. Data is collected to evaluate prior participant performance on the face/place task.
Stable Block	a block in which the participant is wearing the Emotiv, but not receiving neurofeedback. Data collected during this time period is used to create a model for later EEG classification.
Feedback Block	a block in which the participant receives real-time feedback classified by the model created with data from the stable blocks.

0.4.2 Setup

In this experiment, participants were shown superimposed pictures of faces and places. They were then instructed to either focus on images of faces and ignore images of places, or ignore faces and focus on places. When focusing on faces, participants were asked to click a button if a shown face

was female, and to not click if a shown face is male (which particular gender a participant was asked to respond to is randomized). Similarly, participants were asked to click a button during a place task if the shown image was outdoors (indoors), and restrain from clicking with then place was indoors (outdoors). Each superimposed picture was shown to the participant for 50 seconds during a trial. During both tasks, 90 percent of images shown to participants were from the click or go category, and the remaining 10 percent were from the no click or no-go category. As a result of the infrequency of the no-go trials, these trials were much more difficult to respond to correctly than the go trials. Thus we call these no-go trials **lure** trials, since they lure a participant to continue clicking after having clicked through a string of go trials.

During the first 30 minutes of the experiment, participants were shown 50/50 mixtures of faces and places. Data collected from this time period was then used to create a model that, given a reading of an EEG measurement, predicted the probability of that data point belonging to the face or place category (i.e. was this data point more likely collected when the participant was participating in a face task, or a place task?). During the remaining 20 minutes of the experiment, the image mixture ratio–the percent of face to place in the superimposed image stimulus–was determined by how "face=like" or "place-like" the classifier rated participants' brains. When a participant was supposed to be focusing on the face task but their brain looked "face-y", the face was made more difficult to see, so that the task become more challenging. Conversely, when the classifier predicted with high confidence that the participant's brain resembled the target class

category, the image ratio was altered so that the task became easier.

0.5 Neuromancer Software/Hardware Stack

In order to run the aforementioned experiment, the author created a software stack to coordinate data collection from the EEG headset along with showing participants stimuli, collection participant responses, and analyzing the collected data in real time. The EEG headset chosen for this experiment was the Epoch Emotiv, as described below. The software framework for coordinating the experiment, Nueromancer, was built in many languages and consists of many interchangeable modules, also described below.

0.5.1 Epoch Emotiv

The Epoch Emotiv [1] is a wireless, hundred-dollar, consumer-grade EEG headset. It contains fourteen data channels plus two reference channels, sampling at 128 samples per second. The headset communicates with a computer via a proprietary Bluetooth dongle.

The Emotiv is extremely susceptible to motion/eyeblink artifacts, and thus a research-grade EEG device with (for example) 32 or 64 electrodes may be preferable in many experiment settings. However, whereas a typical EEG setup might use an electrode cap which connects electrodes to participants' scalps via messy conductive gel, the Emotiv can be cleanly placed on participants' heads using only sponges soaked in conductive solution (contact solution, for example). Thus the Emotiv often can be setup more quickly than a typical EEG. However, the headset design comes at the cost off be-

Figure 1: Epoch Emotiv Headset. https://emotiv.com/media/

ing difficult to align consistently across participants heads, as well as being unable to penetrate thick hair. At least four participants were unable to complete this experiment because the of these reasons.

At least a handful of psychology experiments have used the Emotiv, including several from Princeton Senior Theses ([14], [2], and of course this thesis)! Additionally, Duvinage et al. [5] explored the performance of the Emotiv for classifying P300 signals, an event-related potential that occurs roughly 300 milliseconds after a participant sees an "oddball" stimulus in a sequence of stimuli [13]. Duvinage found that the Emotiv performs significantly worse than a medical grade EEG device, and recommend its use only in situations in which reliability and a high signal-to-noise ratio are non-critical.

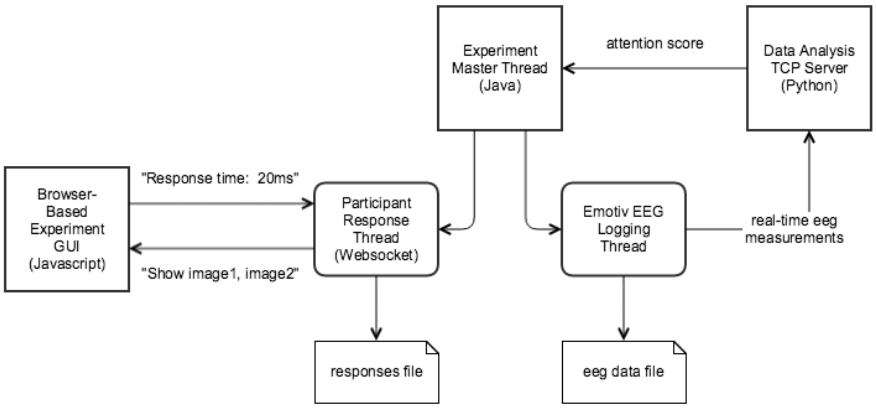


Figure 2: Neuromancer Component Diagram

0.5.2 Neuromancer Overview

The Neuromancer software stack consists of three parts: the experiment GUI, data analysis server, and experiment controller. The experiment GUI displays images to the participant and records participant responses. The data analysis server analyzes collected data, creates a model, and classifies data used for real-time feedback. The experiment controller coordinates collecting EEG data, controlling the experiment GUI, and writing responses and metadata to file.

0.5.3 Experiment Controller

The Neuromancer experiment controller was written as Java (so chosen because the Emotiv API was available in Java). At its start, the controller creates a thread dedicated to polling the Emotiv for data. It continually writes this data to file using a buffered writing structure. This logging thread exposes methods to the main controller thread for starting, pausing,

and stopping data collection. The logging thread starts logging when the participant is shown the first block of the experiment, and stopped when the experiment ends.

In addition to running data collection in the background, the experiment controller coordinates the evolution of the experiment through its phases. That is, the controller guides the experiment's transition from pre-training to stable to feedback to post-training phase, as well as the transition from block to block and trial to trial.

The design of the experiment controller, which is the core of the Neuromancer software stack, was built with as a compromise between flexibility and practicality. At its core, it conducts the experiment via a DFA inner class as explained below. Future users wishing to modify Neuromancer for their own experiments can do so by modifying this DFA inner class.

A DFA or deterministic finite automaton is a deterministic finite state machine. It consists of a series of states and a set of rules for transitioning between states. An diagram of the DFA designed for this experiment is given below:

Listing 1: Dfa Class

```
private class Dfa{
        public void start(){
                /* Method to start DFA, may
                   include setting initial
                   state, displaying start
                   message, etc. */
                return;
        }
```

```
        public void doNext(boolean fromTimer){
                /* Describes how the DFA
                   transitions from its
                   current state to the next
                   state.  The boolean
                   parameter fromTimer is
                   given so that this can be
                   triggered either by a timer
                   , or (for example) via a
                   user-initiated event (like
                   a mouse click).
                */
                return;
        }


}
```

0.5.4 Graphical User Interface

The user interface in this experiment was built to run in a web browser, which is in this case Google Chrome. Writing a browser-based experiment interface yields many potential advantages over more specialized, language-specific GUI frameworks. For one, many templates exist both for the layout components of a browser interface (HTML templates, for example) as well as for the code that might control the browser interface. Second, because Google Chrome (and any other browser) executes Javascript code the same way on every hardware device, essentially any computer that can run a browser can run the Neuromancer GUI. The experiment participant may log responses to the experiment stimuli from any computer that is capable of running a browser, or even a tablet or mobile device. Finally, although not implemented in Neuromancer, using the browser as a display lends it-

self easily to mirrored applications, where, for example, the experimenter might desire access to stimuli that the participant is viewing, even if the experimenter and participant are not using the same computer.

In Neuromancer, the experiment GUI consists of a simple HTML layout and Javascript program. The brain of the GUI is the Java Experiment controller, as described above. The Javascript program opens a Websocket [6] that communicates with Java via a light-weight TCP protocol. It displays images in the location and at the opacity described by the controller.

0.5.5 Data Analysis Server

Python was chosen as the data analysis language for this experiment. While Matlab is a standard language for EEG/BCI analysis, Python provides several advantages over Matlab. First, Python is free and open source. Second, it is unique amongst many programming languages in that it is both a general purpose language, but also well-suited for data analysis and machine learning. On the one hand, libraries like Numpy, Scipy [7], and Scikit-Learn [12] all used extensively in this project, make array manipulations and statistical/machine learning analysis easy and fast. In fact, many of these toolkits are implementations of Matlab functions for Python! Conversely, it is much more difficult to build web tools like a TCP server in Matlab than it is in Python (in fact, this is a pay-for Matlab add-on library!).

The Data Analysis Server for this experiment was built as a Python TCP server. This server listened on a predetermined port for connections made from the Java Experiment Controller. Throughout the course of the experiment, Java creates two types of connection with the Data Analysis

Server: a "data line" is created when the Java data logging thread-the thread that continually polls the Emotiv for readings-sends a "data" message to the Python server. After this connection is accepted, Java writes every value it reads from the Emotiv to the Python Data Analysis server.

Additionally, the Java Experiment Controller opens a "control line" with Python by sending a "control" message. Python acknowledges two types of commands on the control line. Java sends Python a **model** command to instruct the Data Analysis Server to create a model based on a journal file name and EEG data file name that Java passes to Python along with the command. When Python finishes building a model, it sends Java an acknowledgement, and starts to store in memory Emotiv measurements written from the data line. Subsequently, the Java control line may send Python a **pred** command along with a timestamp indicating the start time of the current trial. Upon receiving such a command, Python uses the provided timestamp to identify the correct data to feed into its previously created model. If such data exists, Python makes a prediction based on the data and returns the prediction to Java. Importantly, Java also informs Python when it has started a new experiment block, at which point Python empties its in-memory EEG data store. Limiting the size of the data Python stores in memory at any time is important for keeping response times low. With its current design, the Python and Java programs are designed to run on the same machine within the same working directory. However, by simply storing the journal and EEG data files in a network-mounted directory, we can allow the Python and Java programs to run on different computers. Such a setup might be useful if, for example, we would like to run our data

collection program on a local computer while performing computationally expensive Python data processing on a remote computer.

0.6 Data Analysis Techniques

0.6.1 Introduction to EEG Data

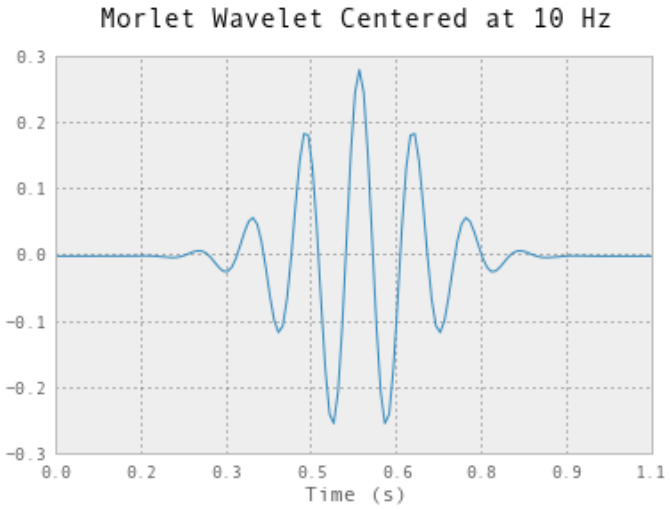
0.6.2 EEG Preprocessing

Eyeblink Removal

Frequency Band Analysis

One of the most common features to consider when analyzing EEG data are the frequency components of the data. For example, alpha band, beta band, etc. Fourier decompositions, which represent time series data in a basis of sines and cosines of varying frequencies, are perhaps the most common technique in signal processing for analyzing signal frequency makeup. Unfortunately, Fourier decompositions only yield entirely meaningful results when their input signal is a periodic function. If this is not the case, as with EEG data, then the Fourier decomposition describing time points near the temporal boundaries of the input signal will be distortedthat is, the transform will only be meaningful in the temporal middle of the input signal. As a result, Fourier decomposition does not property capture the quick frequency evolution we expect to see in EEG signals.

Wavelet decompositions are similar to Fourier Decompositions, but represent an EEG signal in a basis of functions known as wavelets. One commonly used wavelet in EEG analysis is the Morlet wavelet. It consists of a



temporally-centered sine wave whose amplitude decays as a Gaussian with distance from its center. Wavelets, although computationally expensive to calculate, provide the temporal resolution that Fourier analysis fails to capture.

In this experiment, we performed face/place classification using a Morlet wavelet basis. We used wavelets spaced from 1 to 60 Hertz on a logarithmic scale. The results of a wavelet decomposition of a single participant over a single trial are given below.

Time Binning

Removing Baseline Signal

Dimensionality Reduction

Principal Component Analysis or PCA is a dimensionality reduction technique that takes as input a data array and a number k, and returns a set of

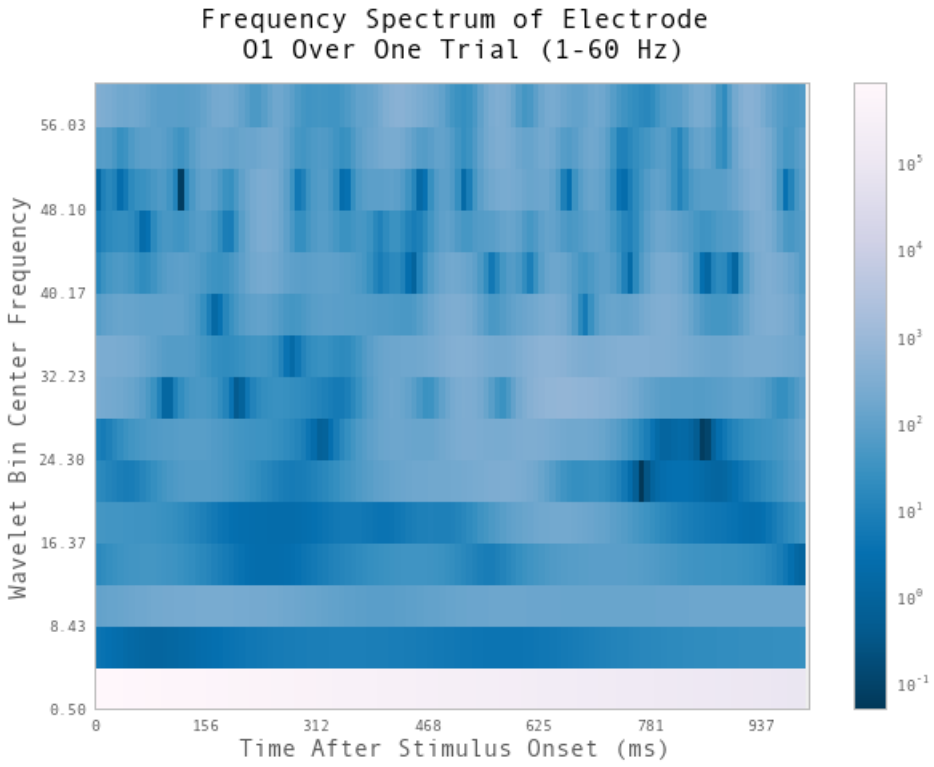


Figure 3: Wavelet decomposition of electrode O1

k vectors such that the first vector has the highest possible variance within the given data, the second vector has the second highest variance given the constraint that it is orthogonal to first vector, and so on. Independent Component Analysis or ICA is similar to PCA, but given a parameter k, returns the k orthogonal vectors that span the data but together have the largest possible variance within the data. Is this true?

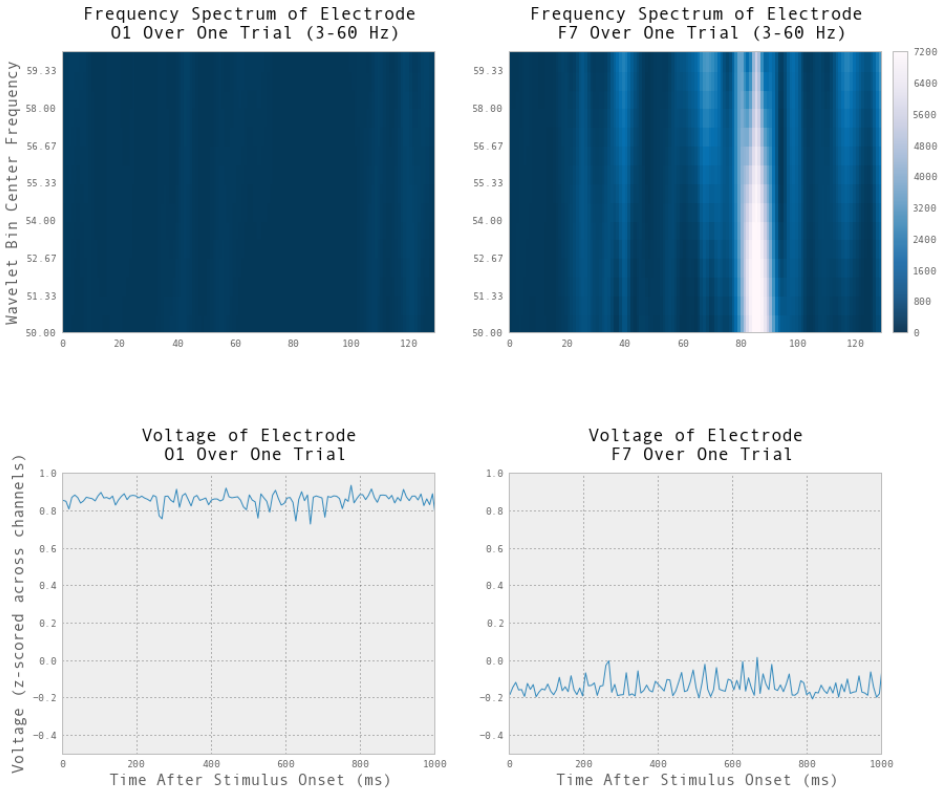


Figure 4: Eyeblinks can be seen easily in frequency space for electrode F1

z-score measures

0.6.3 Machine Learning Classifiers

Logistic Regression with Penalization

A logistic regression is a simple linear model which takes as input n-dimensional training points and labels, and models the relationship between input vectors and labels as the outcome of Bernoulli trials ...

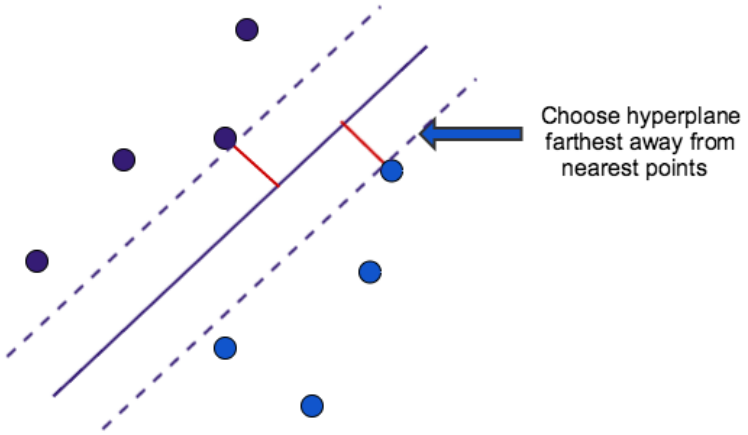


Figure 5: Support Vector Machines find the a separating hyperplane between points in X in Y that maximizes the distance between the hyperplane and those points nearest to the hyperplane.

Support Vector Machines

Support Vector Machines (SVMs) are a commonly used "off-the-shelf" classifier for BCI applications. In their most basic form, SVMs work as follows: data are provided as a n-dimensional vectors, with each point labeled -1 or 1 (faces or places, in this case). Let all training points that are labeled as faces be contained in the set X, and all points labeled as places fall in the set Y. Then the SVM algorithm then finds an n-1 dimensional hyperplane $\mathbb{H} = \{x \in \mathbb{R} | A^T x = b\}$ for some matrix A and $b \in \mathbb{R}$, chosen such that the Euclidean distance between the points in X and Y closest to $\mathbb{H}$ is maximized. In other words, the SVM algorithm finds:

$$\underset{A,b}{\operatorname{argmax}} \min_{z \in X \cup Y} d(H, z)$$

,

where $d(H, z)$ is the Euclidean distance between H and z.

As described above, an SVM divides data that is linearly separable. However, a small modification can allow SVM's to quickly classify data that is not linearly separable. SVM algorithms maximize the Euclidean distance between points and hyperplanes in a way that depends only on a calculation of dot products. It is possible to map input feature vectors into arbitrarily high dimensional space using a variety of methods, changing the dot products of the vectors in the new space. If the dot product of vectors in this new space is a simple function of the vectors in the lower dimensional space, the an SVM algorithm can use this dot product function, also known as a "kernel trick", to calculate a margin-maximizing hyperplane between points in this higher dimensional space. Past BCI experiments have successfully used Gaussian and Radial Basis Function kernels to classify EEG data [10].

0.7 Results

0.7.1 Simple Time Bin Analysis

The simplest method of data analysis used in this experiment simply bins data into time buckets, subtracts baseline values from time buckets, averages the data across time bins, z-scores the resulting data across channels, and classifies the data using a Logistic Regression model with L2 penalization. The results of such an analyses over combinations of time bin widths and offsets from the stimulus onset are given below. The intensity of each grid

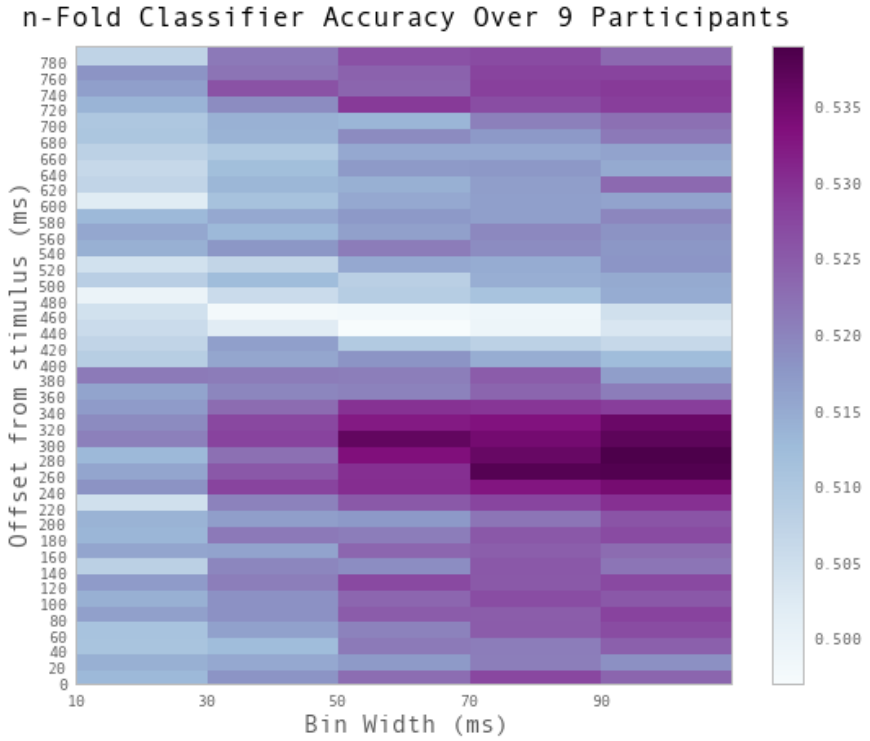


Figure 6: n-Fold classifier accuracy over various bin widths and offsets after stimuli.

cell represents the n-fold classifier on the data set using the (time bin width, time after stimulus onset) pair. Classification is averaged over 9 participants, after conducting 15-fold validation over 30 epochs/30 minutes worth of training data.

The results of this analysis as shown in Figure 6 shows that, for the nine participants sampled, classification accuracy was highest at 280 milliseconds after stimulus onset with a bin width of 90 milliseconds. With these parameters, classification accuracy is 0.54 with a standard deviation

of 0.06 across participants. It is possible that this bin of time is particularly classifiable because it captures the P300 EPR, which begins at roughly 250 to 300 milliseconds after the participant is shown an outlier or "oddball" stimuli.

0.7.2 Wavelet Time Bin Analysis

0.7.3 Component Analysis

0.7.4 Classification Accuracy versus Participant Accuracy

A major hypothesis of this experiment was the notion that classifier accuracy is correlated with a subject's ability to focus intently on faces while ignoring places, or the reverse. Consider, for example, an SVM that divides face/place data points as shown below. We predict that a participant who can concentrate well on the face/place task produces EEG readings that are separable by some learning algorithm, in this case an SVM hyperplane. We further predict that when a participant is concentrating poorly, his/her EEG readings lie closer to the hyperplane i.e. face/place concentration boundary than when he/she is focusing intently on one category or the other.

To test our first hypothesis, we consider how well a participant's classifier accuracy predicts that participant's performance on responding to lure trials. In other words, we train a model on a given participant (in this case, we use WHATEVER WE DECIDE TO USE), and calculate the 15-fold classifier accuracy of this classifier on the collected data. We then compare this measurement to the participant's percent correct responses on lure trials. The results are given in figure 7.

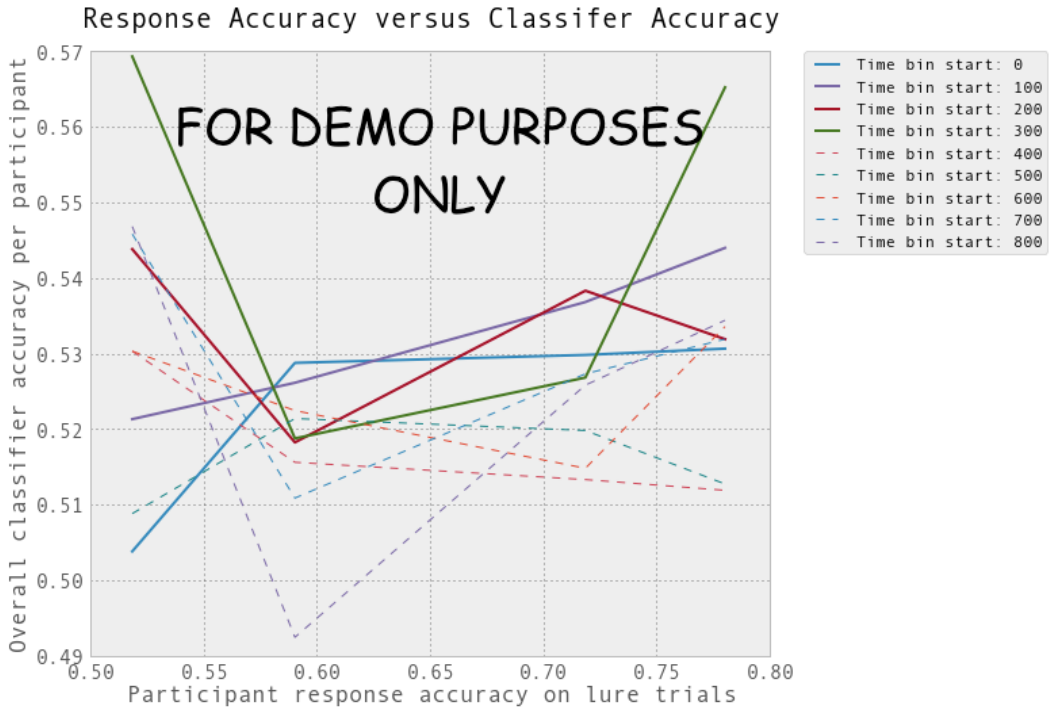


Figure 7: 15-fold classifier accuracy over various bin widths and offsets after stimuli.

Additionally, within a single participant, we would like to know whether the participant's brain was "more classifiable" when he/she performed well on lure trials than when he/she made an incorrect response before lure trials. A logistic regression model provides a measure of "confidence" for each of its predictions, which is the model's estimate of the probability of the given sample point of belonging to each of two categories. We can think of the difference between these two probabilities, then, as representing how far the given data is from the model's decision boundary. We would like to know if a particular participant's EEG readings were closer to this decision boundary before the participant made a mistake (false alarm) than before

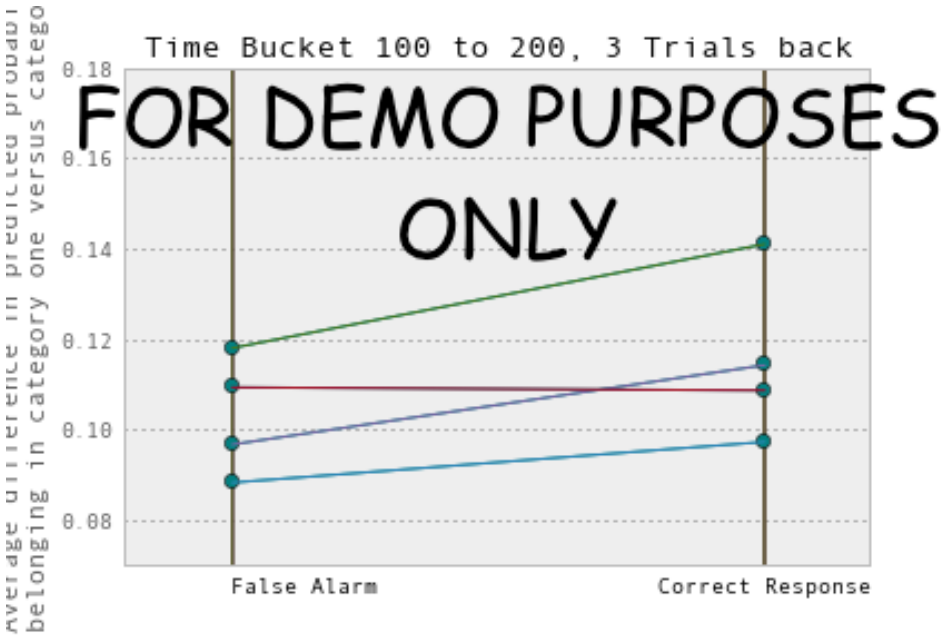


Figure 8: Classifier accuracy before false alarms versus correct rejectinos.

the participant responded to the lures correctly (correct rejection). We plot the average difference in category probabilities before correct rejections and false alarms over 9 participants in figure 8.

0.7.5 Realtime Feedback

A core goal of this experiment was to study how well humans can learn to control their sustained attention using feedback generated on the basis of the content of their thoughts. In this experiment, training was achieved by altering the opacity of the target category image on the basis of the participant's level of concentration on that category. Because of its simplicity, we chose to provide participants with feedback calculated on the basis of a logistic regression classifier with L2 regularization, calculated on a z-scored,

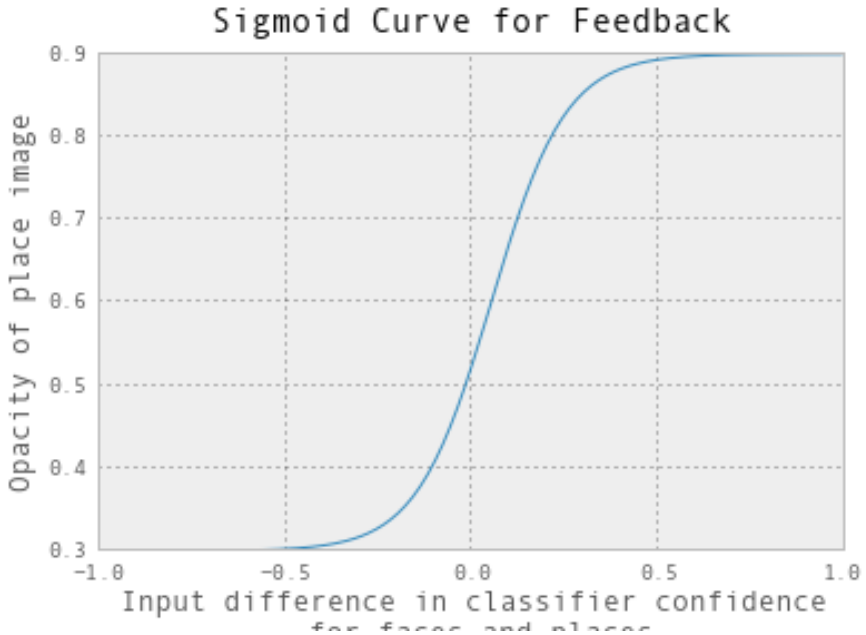


Figure 9: Sigmoid curve maps classifier confidence differences to image opacity values, ensuring that opacity is scaled appropriately to the range of variation in human concentration

100 ms time bin that began 100 ms after stimulus onset. The code for the calculation of this feedback can be found in the Appendix. The exact face/-place image ratio for the feedback was calculated by looking at the classifier confidence calculated for the trial preceding the current trial. This classifier confidence fell between a range of 1 for perfect concentration on the target category, and -1 for perfect concentration on the distractor category. In general, participants' classification confidence was confined to a small range between

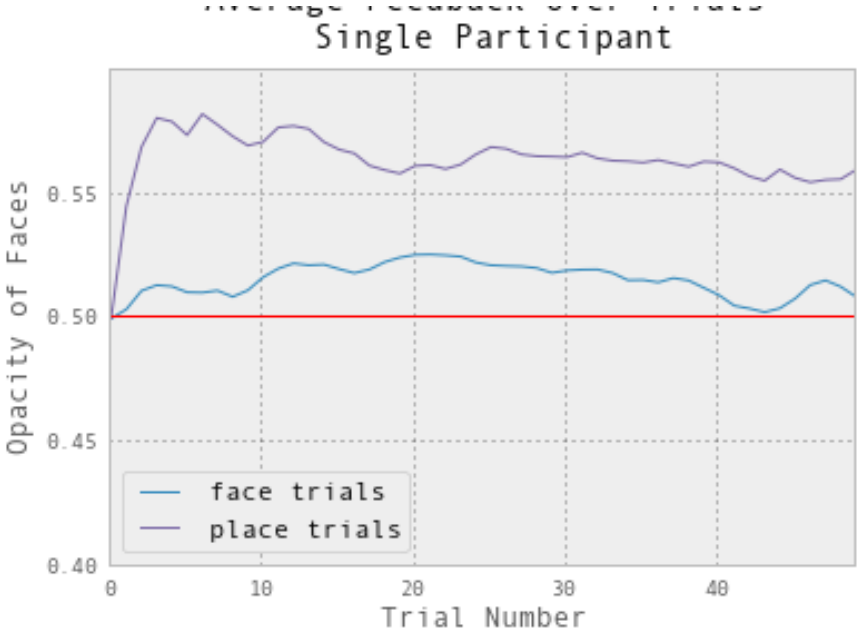


Figure 10: Average feedback received by one participant over all trials.

Bibliography

[1] Emotiv epoch, 2014.

[2] Michael Adelson and R Schapire. Emotiv Experimenter. *compmem.princeton.edu*, 2011.

[3] Robert J. Barry, Adam R. Clarke, and Stuart J. Johnstone. A review of electrophysiology in attention-deficit/hyperactivity disorder: I. Qualitative and quantitative electroencephalography. *Clinical Neurophysiology*, 114:171–183, 2003.

[4] Megan T deBettencourt, Jonathan D Cohen, Ray F Lee, Kenneth A Norman, and Nicholas B Turk-Browne. Closed-loop training of attention with real-time brain imaging. *Nat Neurosci*, 18(3):470–475, 03 2015.

[5] Matthieu Duvinage, Thierry Castermans, Mathieu Petieau, Thomas Hoellinger, Guy Cheron, and Thierry Dutoit. Performance of the Emotiv Epoc headset for P300-based applications. *Biomedical engineering online*, 12:56, 2013.

[6] I. Fette and A. Melnikov. The websocket protocol rfc.

[7] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2015-04-10].

[8] Yukiyasu Kamitani and Frank Tong. Decoding seen and attended motion directions from activity in the human visual cortex. *Current biology : CB*, 16(11):1096–1102, June 2006.

[9] N. Lofthouse, L. E. Arnold, S. Hersch, E. Hurt, and R. DeBeus. A Review of Neurofeedback Treatment for Pediatric ADHD. *Journal of Attention Disorders*, 16(5):351–372, 2012.

[10] F Lotte, M Congedo, a Lécuyer, F Lamarche, and B Arnaldi. A review of classification algorithms for EEG-based brain-computer interfaces. *Journal of neural engineering*, 4(2):R1–R13, 2007.

[11] Kenneth A Norman, Sean M Polyn, Greg J Detre, and James V Haxby. Beyond mind-reading : multi-voxel pattern analysis of fMRI data. (x), 2006.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] Terence Picton W. The P300 Wave of the Human Event-Related Potential, 1992.

[14] Nicole S Rafidi, Kenneth Norman, and Robert Schapire. A Brain-Computer Interface for Enhanced Learning using Classification of EEG Data. *compmem.princeton.edu*, 2012.

[15] W J Ray and H W Cole. EEG alpha activity reflects attentional demands, and beta activity reflects emotional and cognitive processes. *Science (New York, N.Y.)*, 228(4700):750–752, 1985.