Event Recognition in EEG Data

By

Kate Pabst

A thesis submitted in partial fulfillment of the

requirements for the

M.S. Biometry and Statistics

University at Albany

April 2015

**TABLE OF CONTENTS**

LIST OF FIGURES iii

LIST OF TABLES iv

INTRODUCTION 1

LITERATURE REVIEW 2

METHODOLOGY 3

Data collection and handling 3

Model definition (LDA) 5

Model definition (Stretch) 9

RESULTS AND DISCUSSION 11

Evaluation criteria 11

Evaluation 12

CONCLUSION 14

REFERENCES 15

**LIST OF FIGURES**

Figure 1: 10-20 system 1

Figure 2: Example of timing of data packets into the R script 3

Figure 3: Hardware/software schema 4

Figure 4: Example of events captured by keystrokes 5

Figure 5 : Typical right event 6

Figure 6: Calculating difference 7

Figure 7: Peak & valley identification with .95 and .05 quantiles 7

Figure 8: Separation of classes (LDA & PCA) 7

Figure 9: Prediction of new event 8

Figure 10: Autocorrelation chart for time periods 8

Figure 11: Difference before and after transformation 10

Figure 12: Multiplier for range of displacements 10

Figure 13: Example event time series 11

**LIST OF TABLES**

Table 1: Sensor positions 3

Table 2: Training and testing datasets 4

Table 3: Suggested sensor patterns across events 5

Table 4: Characteristic variables for LDA 6

Table 5: Possible errors 11

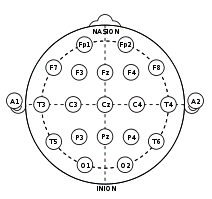
Table 6: Results tables (LDA and stretch methods) 12

Table 7: Center back error investigation 13

**Section 1 – Introduction**

Electroencephalography, or EEG, refers to the recording of electrical activity across the scalp of the head. The practice is not new. Scientists and physicians were exploring this topic as early as the late 1800’s and the first use in a human patient was in 1924 by a German psychiatrist Hans Berger. Since then, there has been much research into the use of EEG to diagnose medical conditions and investigate neurological processes.

Historically, EEG recordings have been obtained by placing electrodes on the scalp at predetermined locations. The locations of these sensors generally follow a placing scheme called the 10-20 system. 21 sensors (19 recording sensors or data channels, a ground sensor, and a reference sensor) are placed upon a subject’s head to measure the currents. (The 10-20 in the name refers not to the number of sensors but to the spacing between them.)

Figure 1: 10-20 system

Such setups have proved a useful tool for decades of researchers. But the use of EEG was primarily contained in hospitals, laboratories, and research institutions. In recent years, firms have begun to explore EEG devices for the consumer market. One of the first, Mindball, was produced in 2003 by Interactive Productline for a retail price of $20,000. Since then, many companies have developed EEG products for the consumer market. Of the 16 devices known to this author, 6 were released in the last 3 years alone.

Some devices are simply for entertainment. A toy manufacturer, Uncle Milton, unrolled a simple device in their Star Wars Force Trainer toy. Others provide much more insightful data and are meant to be used as personal health aides, hobbyist devices, and other purposes. Currently, several companies offer products of varying degrees of functionality for anywhere between $50-$800.

This expansion of EEG technology into the consumer market has resulted in a wave of new applications. Software developers, hobbyists, and even life coaches are now touting EEG solutions to a variety of problems. Some want to use the technology for guided mental exercises (i.e. brain entrainment) to improve mental state. Others want to play beer pong with an EEG headset. Whatever the motivation, there is a wealth of new ideas for this old technology. Crowd funding websites like Kickstarter.com are raising money for technology startups. iPhone application developers are cashing in on the trend by developing smartphone apps available on the iTunes store.

These new applications mean there is an increasing demand for the ability to receive and extract meaning from EEG data streams. There have always been researchers developing models for EEG data, but now these models are in demand by a wider audience. The models needed tend to have common requirements; they can be implemented by anyone with standard function libraries available in programming languages, they do not need expensive software, they can extract insights from fewer channels of messier data, and they can be run quickly to extract insights in real time.

This paper is an attempt to create two such models. The hardware used is the Muse device from the company InteraXon which began shipping product April 2014. It provides 4 channels of sensor data (considerably less than the 19 in clinical applications) and is designed to be worn as a headband.

**Section 2 – Literature Review**

The body of literature surrounding EEG is vast. Therefore, for this project literature searches concentrated on works that dealt with the following subjects; consumer EEG devices, linear discriminant analysis as applied to EEG, and blink or eye movement detection in EEG.

As discussed in the introduction, methods for consumer grade EEG devices should be fast and able to be implemented in any common programming language. The authors Tiganj, Mboup, Pouzat, and Lotfi (2010) noted this when they submitted a algebraic method. Other authors show (by their work) the ways in which consumer grade devices take EEG research into areas that are new, such as gaming. Chumerin, Manyakov, Van Vliet, Robben, Combaz, and Hulle (2013) create an algorithm that works with the device produced by a firm Emotiv to navigate a maze. Others, such as De Man (2014) describe an attempt to predict emotional reaction to visual content such as videos.

Garrett, Peterson, Anderson, and Thaut (2003) compared linear discriminant analysis to other, non-linear methods. In particular, they compared the performance of LDA to neural networks and support vector machines when classifying a type of task (such as composing a letter or performing multiplication problems mentally) performed by a subject. They concluded that non-linear methods work slightly better in this case. However, they are more difficult to implement. Linear discriminant analysis has also been compared to quadratic discriminant analysis, as in Vidaurre, Schogl, Cabeza, Scherer, and Pfurtscheller (2007). In this paper, three features (characteristic variables fed into decision model) were selected and each used in the LDA/QDA model to predict tasks. Here, the prediction models were updated continually to account for the fact that EEG patterns often change in time even within the same subject.

Sometimes eye movement in EEG is considered the primary signal. In other research, it is considered a noise that should be identified and removed. Much of the literature is focused on detecting blinks, but such papers where of interest here because methods only developed to detect blinks may be adapted to more general eye movements. Independent component analysis is a popular tool for identifying features such as blinks or eye movement. Here, a multivariate signal is separated into additive subcomponents. (The multivariate designation of the signal reflects the belief that the signal is comprised of at least two groups: reactions to large groups of neurons firing at the same time in response to stimuli, and voltage differences caused by muscle movements.) Joyce, Gorodnitsky, and Kutas (2004) propose a method using independent component analysis. Delome, Sejnowski, and Makeig (2006) note that ICA is a very common technique and test it against 5 other methods. Two of the same authors introduced ICA as a method of EEG data analysis in an earlier paper Makeig, Bell, Jung, Sejnowski (1996). Chambayil, Singla, and Jha (2010) use a neural network approach to detecting blinks.

Some inspiration for the methods in this paper (especially the stretch method) was drawn not from academic papers but from technology web forums or blogs. The problem of detecting blinks is a common one for consumer EEG enthusiasts. While many academic papers treat blinks and other eye movements as noise to be removed, many enthusiasts see them as an input to allow users to interact with content. Descriptions of projects such as the mind controlled EEG robots (Vjvarada, 2012) and consumer EEG blink detection (Grimm, 2012) proved to be useful in this project.

**Section 3 - Methodology**

**Section 3.1 – Data collection and handling**

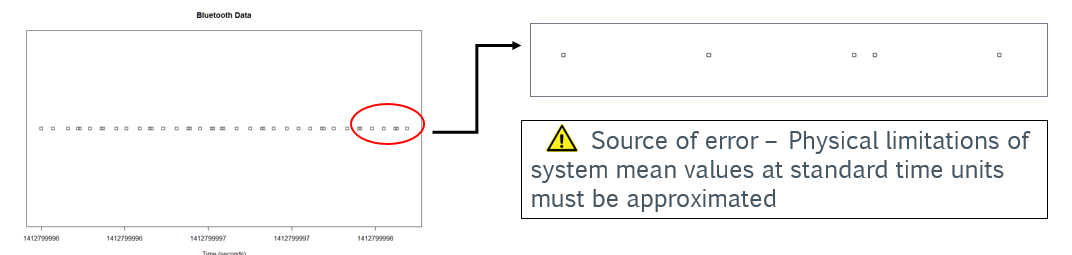
The Muse headband has 7 sensors; 2 are located on the forehead, 2 behind the ears, and an additional 3 serve as reference sensors. The forehead and ear sensors produce the data that is of interest in this model. A reference sensor provides a stable potential (voltage) against which a measuring sensor can be measured. The four channels of the measuring sensors are referred to hereafter as S1, S2, S3, and S4.

Table 1: Sensor positions

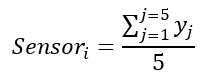
|  |  |
| --- | --- |
| Sensor | Position |
| S1 | Left ear |
| S2 | Left forehead |
| S3 | Right forehead |
| S4 | Right ear |

The device sends data via Bluetooth at 220 Hz (or 10 bits at 220 samples per second). A full explanation of the communication protocol is beyond the scope of this paper, but due to a variety of reasons (buffering, dropped data packets, etcetera) the data does not come into a client at perfectly spaced time intervals. Shown below are 300 data points on channel S1, with the space between the points representing time between arrivals. It is apparent both from the first chart and the enlarged sections below that the data comes in irregularly.

Figure 2: Example of timing of data packets into the R script



To translate the data into standard time intervals, a simple averaging method is used. A standard time unit is set at .1 seconds. Each standard time unit , the value of the 5 nearest neighbors are averaged to form the sensor value at time .

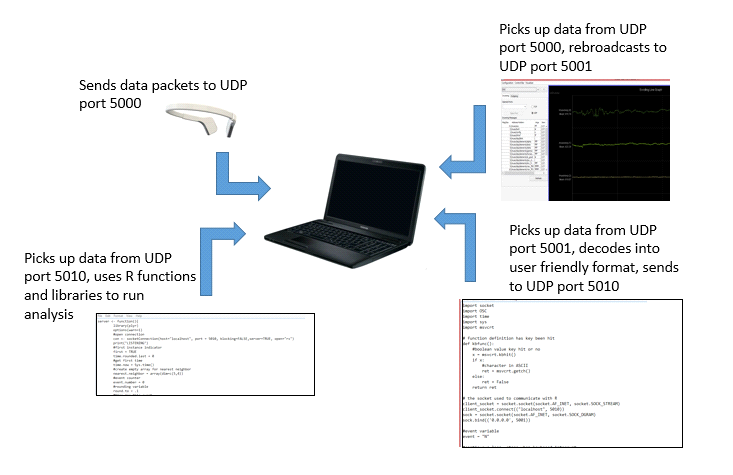


Eqn

It is desirable to use the R computing environment, with its well tested libraries of functions, to create the predictions. However, R does not have much support for streaming data into the environment in a real time manner. Therefore, a schema is set up in which the host computer (i.e. the computer interacting with the headband) acts as both a client and a server.

InteraXon provides free software online in their developer kit which will read the data on a TCP/UDP port and rebroadcast it to another TCP/UDP port of the user’s choice. Python, a high-level general purpose programming language, has libraries which make reading and decoding fairly straightforward. A python script acts as a client, reading the data in from a UDP socket and decoding it. The python script sends the decoded data to an R script which serves as a server. Here, the data is used to run analyses and create predictions. The schema is shown below.

Figure 3: Hardware/software schema



To collect data for analysis, a user sits at the computer wearing the EEG headband. R/Python scripts are run which record keystrokes from the user. The user makes repeated eye movements while indicating with a keystroke that they are doing so. Keystrokes accepted are U (for up), D (for down), R (for right), L (for left), and Space (for neutral or doing nothing). For example, when a user begins a right eye movement, they hit the R key. At the end of the movement, they hit Space to indicate return to the “other” states. A number of events were recorded for training and testing models for both the left/right and up/down classification problems. This data was divided into two somewhat equal parts for testing and training. The number of events for each dataset is shown below.

Table 2: Training and testing datasets

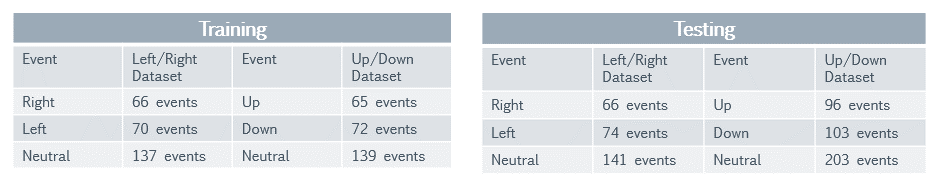


Figure 4: Example of events captured by keystrokes (movement between red lines)



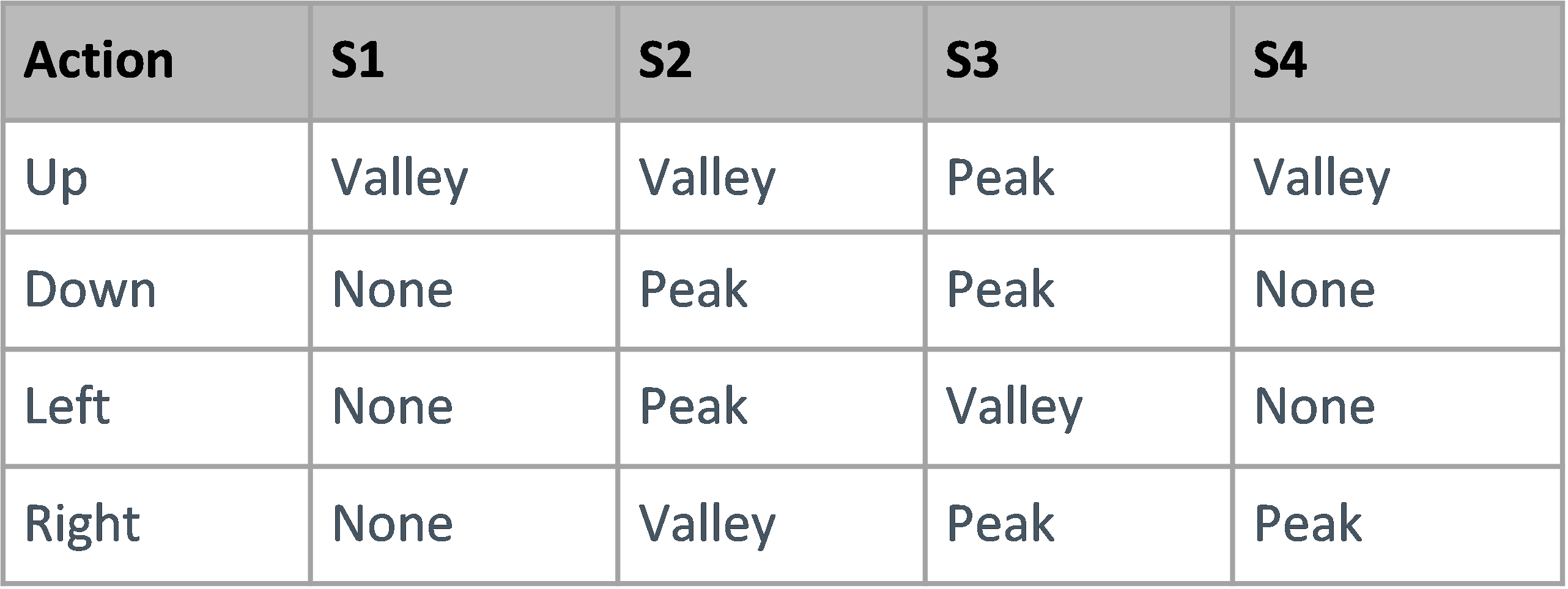
This method of data collection can introduce several sources of error. Some data accuracy will be lost when the standard time unit measurement must be estimated from the five nearest neighbors. There is also a slight time lag between the time the measurement occurs and the time that the data is read into the final UDP port. (Remember that the data packets travel between several UDP ports before being read into the R script.) Additionally, the data collection for building the models is reliant on the user entering the keystrokes at the correct times. A user may not hit the key indicating they are starting an eye movement at exactly the time the actual movement begins.

**Section 3.2.1 – Model definition (LDA)**

A key portion of the prediction method proposed uses a linear discriminant analysis approach. In order to use LDA, it is necessary to first create a matrix with columns representing characteristics and rows representing instances. In other words, one or more metrics must describe the nature of the sensor time series at any point in time. Many ways of describing the time series were explored. These include (but are not limited to) the raw value of the sensors, the smoothed value of the sensors, and different distance metrics (Euclidean, dynamic time warping, min/max) to average “profiles” (the sensor value of many averaged similar events). The final characteristic metric selected, however, describes the number of peaks and valleys in the time series.

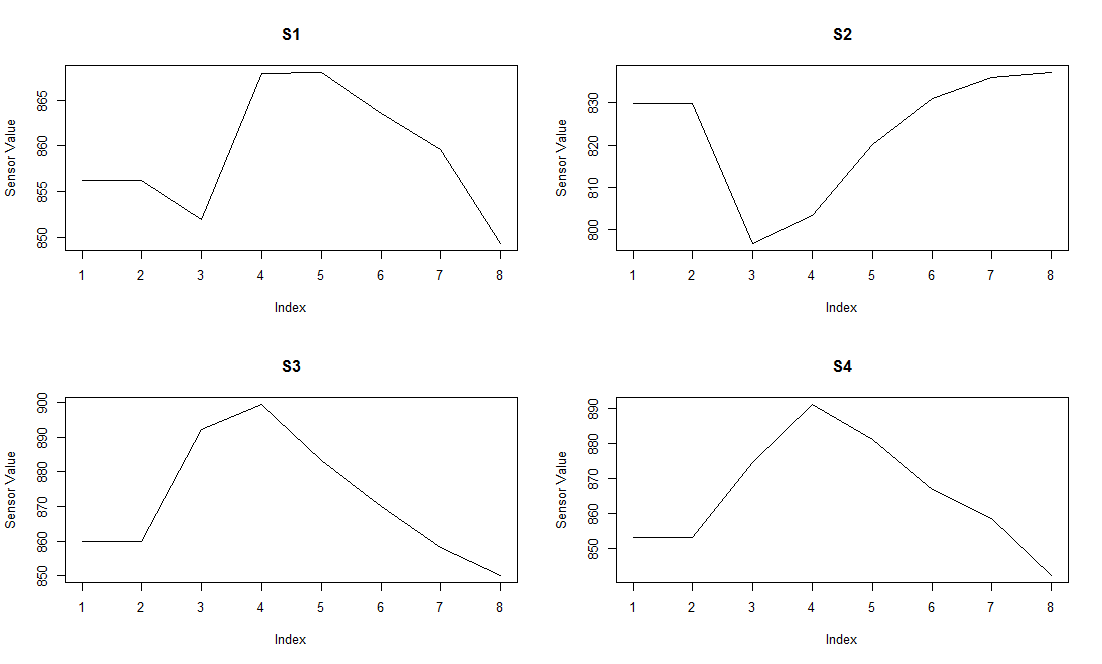
Each movement is characterized by variations in the sensor time series that are predictable in direction (if not in scale). It is readily apparent when watching the sensor time series on a display screen that certain movements will create spikes and troughs in different sensors. Not all sensors will react in a predictable manner at all events. In addition, what constitutes a pattern can be somewhat subjective. An example of patterns based on one person’s (this author’s) interpretation can be seen below.

Table 3: Suggested sensor patterns across events



A typical right movement is seen in the next figure. The figure shows an averaged profile, created by averaging many sensor time series together in which the event was identified as right. By examining the patterns over many events, it may seem that S2, S3, and S4 usually have similar peaks and valleys during the right event. However, the S1 time series may show either a peak or valley during the event. Therefore, one might conclude that S2, S3, and S4 are good predictors of the right event while S1 is not.

Figure 5: Typical right event

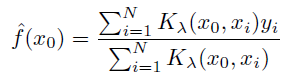


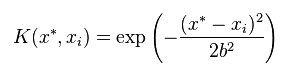
The table below shows a typical matrix fed into the algorithm. The user chooses a time window (here 5 standard time units or .5 seconds). Within the time window, the solution calculates how many peaks and valleys were experienced in the time series for this sensor.

Table 4: Characteristic variables for LDA

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Event | PeaksInS1 | ValleysInS1 | PeaksInS2 | ValleysInS2 | PeaksInS3 | ValleysInS3 | PeaksInS4 | ValleysInS4 |
| N | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| N | 0 | 1 | 2 | 0 | 0 | 1 | 0 | 0 |
| R | 2 | 2 | 1 | 0 | 2 | 1 | 0 | 0 |
| …. | …. | …. | …. | …. | …. | …. | …. | …. |

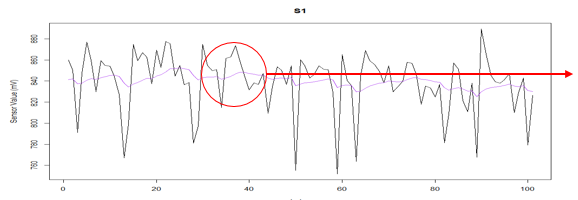
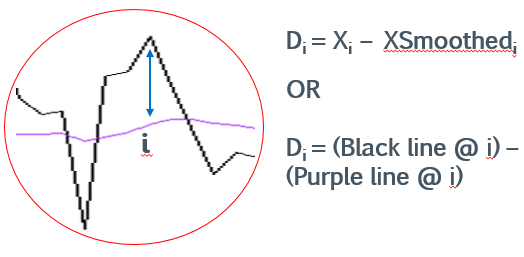
To create the table above, the peaks and valleys in the time series must be identified. The time series are first smoothed. For this model, kernel smoothing is implemented using R’s ksmooth function. The ksmooth function allows the choice of normal or box weighting. Normal is chosen for this model. The bandwidth can be selected by the user during the model training to increase or decrease smoothness, but for all the models in this paper the bandwidth = 25.

 Eqns.



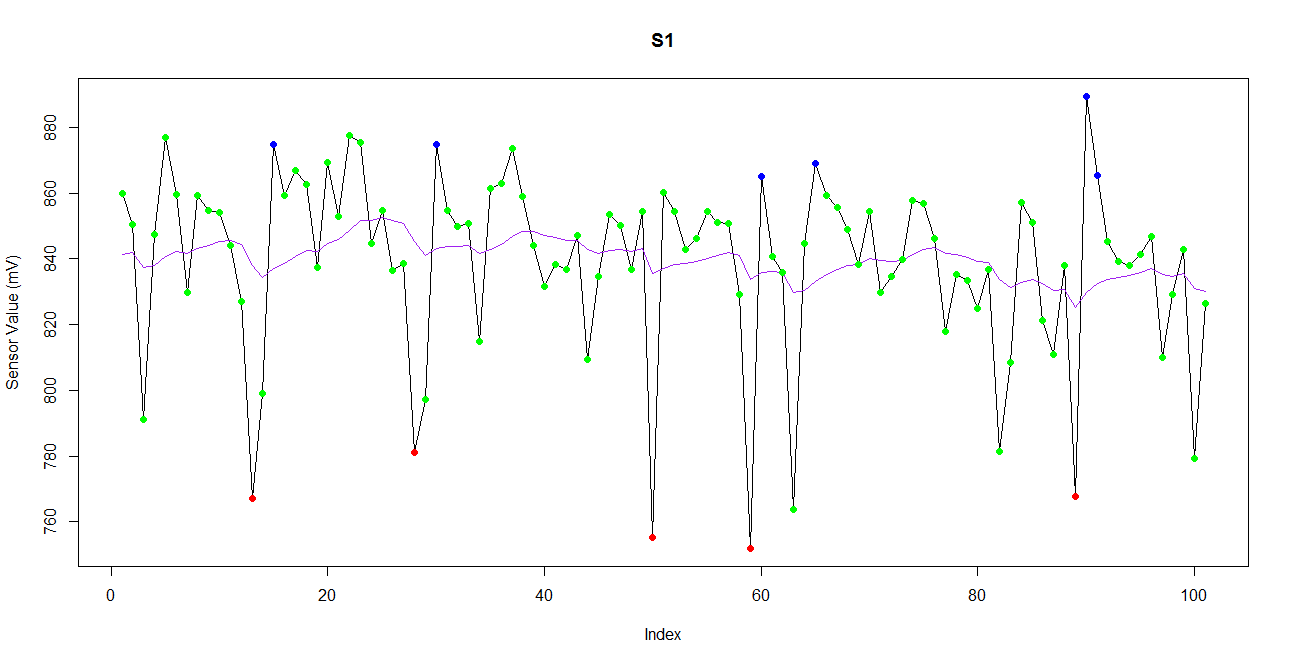
Next, the difference between the raw sensor value and the smoothed value is calculated. This is used to determine how far away a current sensor reading is from the changing mean of the time series. A relatively large bandwidth was chosen for the kernel smoothing because we want the line to reflect a non-stationary mean.

Figure 6: Calculating difference

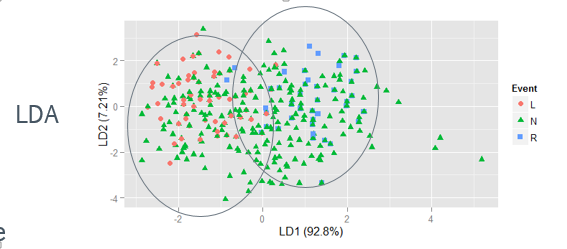
The last n difference data points (where n = bandwidth) are then used to create an upper and lower quantile. The upper and lower quantile can be set by the user during the running of the script files, but generally they are set at .9 and .1. Finally, any point whose difference is above the upper quantile is deemed a peak. Any point whose difference is below the lower quantile is deemed a valley. In the figure below, an example of using this method to identify peaks and valleys is shown. The green dots indicate a point within the control limits. The blue and red dots indicate peaks and valleys, respectively.

Figure 7: Peak & valley identification with .95 and .05 quantiles



With these peak and valley identifications, it is possible to create the table shown previously in Table 4. There remains one additional complication. If the separation between the classes in the LDA is shown on a plot (see figure 8), it is fairly easy to separate the right and left (or up and down) events. However, the N category (neutral or doing nothing) has significant overlap with both of the classes. The overlap in figure 8 is shown both for the LDA model and a similar PCA (principal components analysis). This is largely because not looking right or left leaves a great many things one might be doing. The user may be looking forward, fidgeting, blinking, or any number of other activities. Therefore it is hard to accurately predict the movement by simply picking the class with the highest probability returned by the LDA.

Figure 8: Separation of classes (LDA & PCA)



Because of the overlap problem, two additional constraints are placed on the prediction from the LDA. First, if all the sensors are within the control limits (the upper and lower quantiles), the prediction bypasses the LDA altogether and automatically assumes an N event. Second, the posterior probability of a predicted point is examined. The chart below shows the prediction process of new data for the LDA approach.

Figure 9: Prediction of new events

All 4 sensors within control limits?

Yes

Predict neutral

Receive measurements for new standard time unit

LDA model from training

No

No

No

Yes

Yes

Predict neutral

Predict right

Predict left

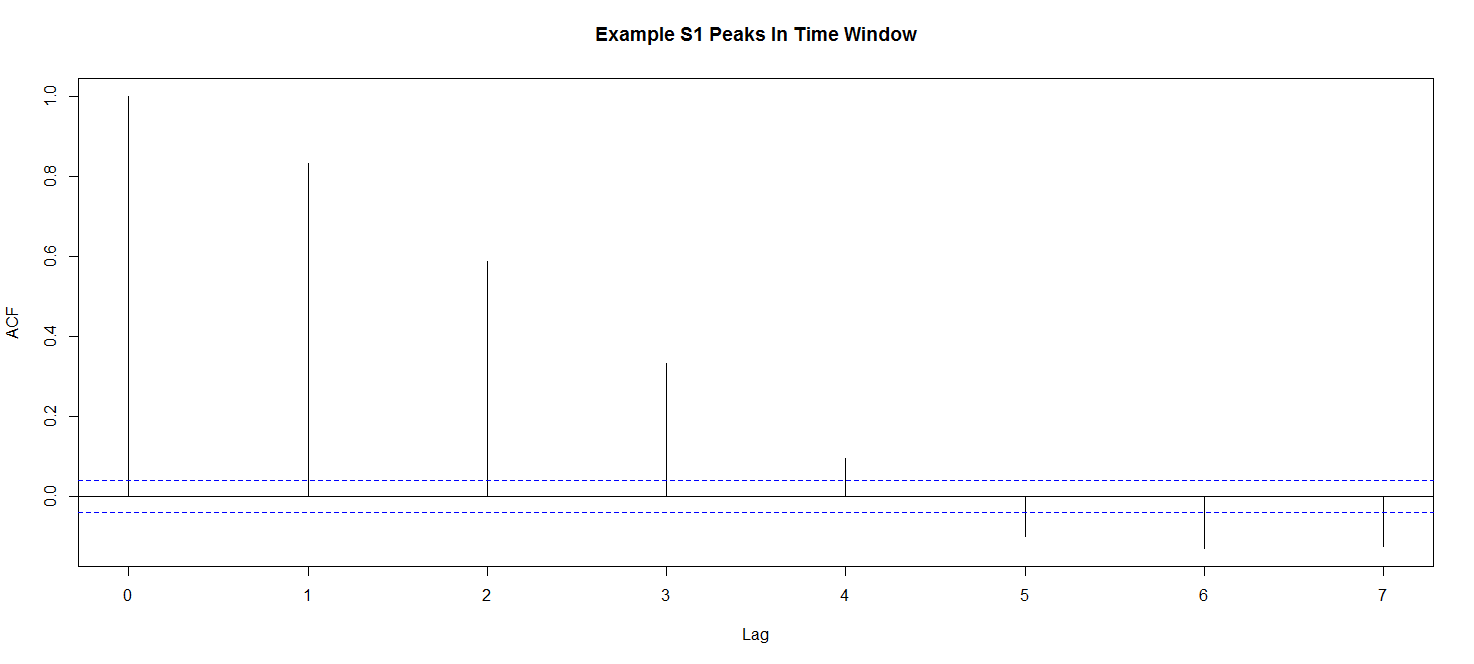
Posterior probability of right threshold?

Posterior probability of left > threshold?

Get posterior probability of all events from LDA

Linear discriminant analysis makes some assumptions about the nature of the problem. For example, it assumes that the subjects (rows of data) are independent subjects. For this application, it is clear that this is violated. Each row of data represents the number of extreme points in the last .5 seconds. Therefore, adjacent rows of data will use time periods that overlap. The figure below shows the correlation between different lag periods.

Figure 10: Autocorrelation chart for time periods

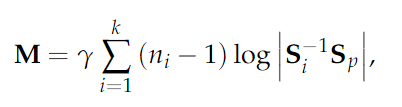


Additionally, LDA assumes that the covariance matrices in each class are similar. Box’s M test is used to test the equivalence of covariance matrices. As is shown, we reject equivalence.

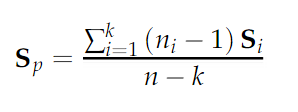
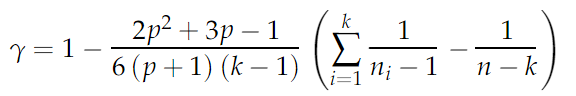
Box M test and results



M follows chi-square with (1/2)(p+1)(k-1) degrees of freedom



where



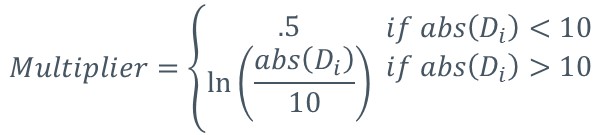


Although the assumptions are not perfectly satisfied, the LDA model still returns reasonable rates of misclassification. However, it may be desirable to consider a model which does not depend heavily on parametric assumptions. To this end, the second method of this paper was developed.

**Section 3.2.2 – Model definition (Stretch Method)**

The stretch method is a relatively simple algorithm which attempts to exaggerate or minimize features of a time series. For example, in LDA method discussed previously a difference between the value at time t and the smoothed line at time t is calculated. The smoothed line represents a type of average value for the time series. When the difference from the time series average is small, it is assumed that this variation is due to noise. When this difference is large, it is assumed that an event (such as an eye movement) is causing the difference. Therefore, a transformation is applied to exaggerate the difference in some ranges and minimize the difference in others. The intention in doing this is to make the difference obvious and easily identified when it is in the “significant” range.

To accomplish this, the difference (of the value at time t – kernel smoothed average at time t) was calculated for this sensor time series. The transformation consisted of multiplying the difference by a multiplier (shown below). Note that any number of transformations could be used so long as they exaggerated the features in one range and dampened them in another.



With this particular transformation, any displacements over ~ 13.6 will be amplified while any under ~13.6 will be dampened. This was chosen because, according to (admittedly subjective) user interpretation, displacements begin to be significant (caused by external events and not noise) in the 10-15 mV range.

Figure 11: Difference before and after transformation

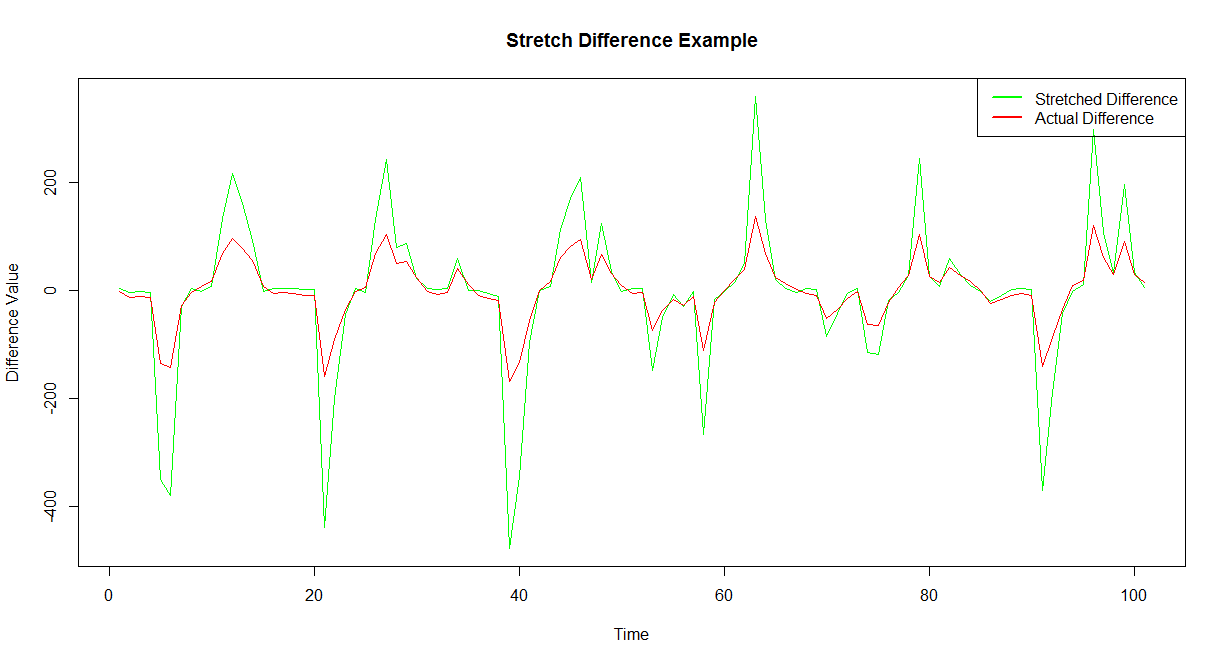
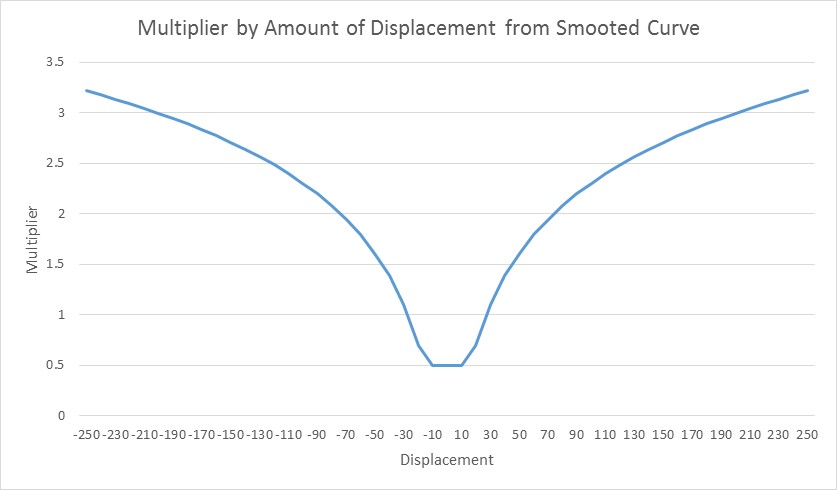


Figure 12: Multiplier for range of displacements



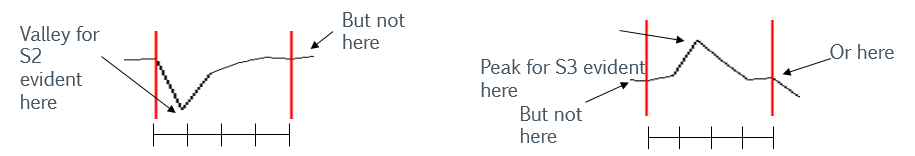
To use the transformed time series, the user must set rules as to what behavior to expect under different events. For example, one might observe that when a left event occurs sensor 2 shows a peak and sensor 3 shows a valley. This expected behavior is set mostly by visual inspection of the data and can be highly subjective. To determine what is a peak or valley, control limits are set. These control limits are simply cutoff points based on visual inspection and trial runs. Everything above the upper control limit is deemed a peak. Everything below the control limit is deemed a valley. At each time period, check if the presence of peaks and valleys matches the expected behavior. For example, at time period (seconds) = 3.4, sensor 2 is above the control limit and sensor 3 is below the control limit. Therefore, predict that this point is a left event.

**Section 4 – Results and Discussion**

**Section 4.1 – Evaluation criteria**

These two methods must be evaluated by their misclassification error. The error here is evaluated on an event level rather than on a time unit level. To explain this, please refer to the following figure.

Figure 13: Example event time series



In the example figure, an event takes place over .4 seconds. In the S2 sensor figure, it is easy to see that a peak may easily be detected at time = .1. At the end of the event, the time series has finished it’s fluctuation and the extreme point can no longer be detected. In the S3 figure, the peak may be detected at time = .3, but not at the beginning and end of the event. This points to the difficulty in correctly predicting for every time point in the event that an event is indeed occurring. Instead, we wish to measure whether at some point during the event, the algorithms were able to detect an event and correctly predict it. When measuring error at this level, there are several errors that could occur. Table 5 lists them.

Table 5: Possible errors

|  |  |
| --- | --- |
| Error Measure | Description |
| % Correct Only | Sometime during the event, the algorithm recognizes an event is occurring and correctly predicts it. Neutral classification (not recognizing an event) is allowed so long as > 1 time periods have the correct non-neutral event classification. |
| % Incorrect Only | Sometime during the event, the algorithm recognizes an event is occurring and incorrectly predicts it. Neutral classification (not recognizing an event) is allowed so long as > 1 time periods have the incorrect non-neutral event classification. |
| % Correct & Incorrect | Sometime during the event, the algorithm recognizes an event is occurring and both correctly and incorrectly predicts it. For example, a left eye movement is predicted as left for 2 time stamps and right for 1 time stamp. |
| % Not Recognized As Movement | The algorithm does not recognize that an event other than neutral is occurring. |

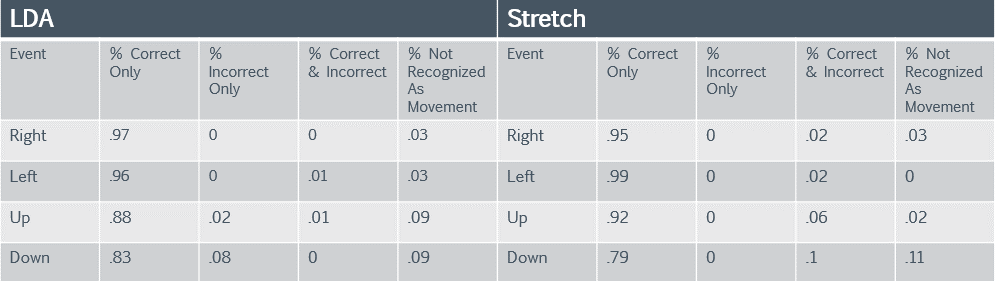
We are also interested in how well the algorithms predict neutral. Because neutral can encompass many things (looking forward, fidgeting, closing the eyes, etcetera), it is difficult to class all neutrals under one specific event. Therefore, the error rate for neutral is simply the percentage of time stamps in which neutral is correctly identified.

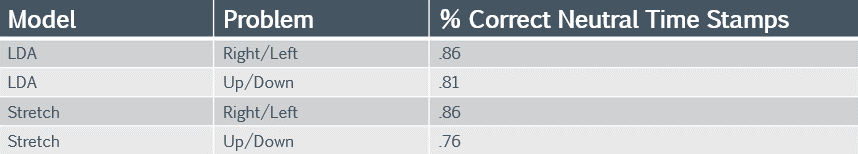
**Section 4.2 – Evaluation**

Each of the models has a number of parameters to set. For the LDA model these are the following: upper and lower control limits for identifying the extreme points, the width of the kernel for smoothing, the time window in which to look for number of extreme points, the left posterior threshold probability, the right posterior threshold probability, and the neutral posterior threshold probability. For simplicity’s sake, the kernel width and time window were set at 25 time points and 5 time points. The other parameters were set using a brute force optimization technique. That is, many runs at many different values of the parameters were conducted and the best combination chosen. The results from the final model chosen are shown in the results table.

The stretch method also has parameters that must be set. These include the width of the kernel for smoothing, the behavior expected for each event, and the upper and lower control limits for the extreme points. The kernel width was again set to 25 time units. The other constraints were set mainly by visually inspecting the data. The results from the final model chosen are shown in the results table.

Table 6: Results tables (LDA and stretch methods)





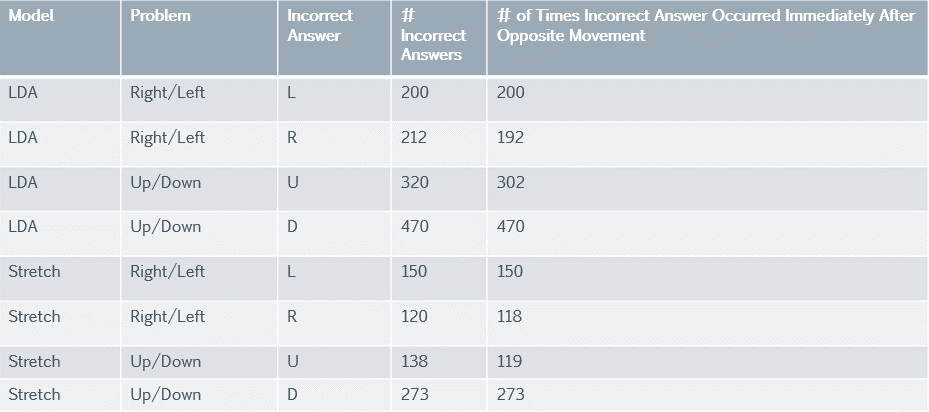
As can be seen from the results table, both the LDA and stretch models have a fairly high rate of correct classification for the right/left problem. The up/down problem has slightly worse performance for both models. The biggest source of error for both models can be seen in the % correct neutral time stamp metric. This is the percentage of the time that the model correctly assumes the user is not engaging in either event (up/down or left/right).

It is interesting to note, however, the pattern of misclassification in the % correct neutral time stamp measure. For example, in the LDA right/left problem there were overall 412 “wrong” answers. This means a time stamp in which the correct answer was “N” but the model predicted “L” or “R”. Of the 200 occurrences where “L” was incorrectly predicted, all 200 occurred .3 seconds or less after a right event. Of the 212 occurrences where “R” was incorrectly predicted, 192 occurred .3 seconds or less after a left event. This pattern of error continues in both the LDA and stretch models.

It is supposed that the reason for these errors is the “center back” behavior of one’s eyes. When a person is asked to look left, the most natural reaction afterwards is to look back to center. This requires making the opposite eye movement (right) to return to center. The model, it seems, is correctly picking up an eye movement on the way back to the center position. Further work on such a problem could include modifying the models to incorporate “return to center” movements.

The following table shows the result of further investigation into the “center back” error. For each incorrect neutral answer (i.e. a neutral misclassified as right, left, up, or down), the analysis counts how many were immediately after a movement of the opposite nature. For example, for each time period predicted as “L” that should have been “N”, how many of them were within .3 time seconds of an actual right event? The term “immediately after” used in the table refers to occurring within .3 time periods.

Table 7: Center back error investigation



It is important to note that these models can be sensitive to equipment malfunction. The headband consists of a plastic band with 7 sensors. The headband is manufactured to be worn by anyone, but shape and size of heads can vary quite a bit. The headband is adjustable, but if one or more sensors are not correctly flush against the skin the data becomes erratic and unusable. Additionally, it is important to keep the sensors clean. Residue such as oil buildup on the skin can interfere with the signal. When this happens, the data is extremely noisy. If the equipment is performing at less than optimal conditions, the data making up the sensor time series is of much worse quality and these error rates do not apply.

**Section 5 – Conclusion**

As the technology for EEG becomes cheaper and more accessible to consumers, hobbyists, and the like the interest in developing analysis techniques for such data will increase. Many researchers in many fields have explored EEG techniques and how to extract meaning from the resulting data. However, much of this research is predicated on the assumption of having conditions that may not be met in consumer applications. New applications will probably not have many channels of data from sensors applied in a rigorous laboratory setting. It is less likely that there will be the luxury of storing the data offline, scrubbing it, and running it through sophisticated software. The new applications for such technology will likely get less amount of data and perform real time analyses that can be implemented in commonly used programming language.

The models proposed here are based on linear discriminant analysis and a more general non-parametric algorithm. Both provide reasonable rates of misclassification for events and can be run quickly to produce real time predictions. Not all programmers may have access to libraries that perform LDA calculations. In that case, the stretch method is simple enough to program in any language. Both models need improvement in “false positives” (i.e. when the user is in the neutral state but the model predicts a movement such as right or left).

Future work could include efforts to reduce the false positives. Additionally, it could expand the methods to incorporate more movements. Predicting right/left/up/down in a single movement is more difficult as there is more overlap between the classes. Perhaps the most interesting direction for future work is to begin to apply these models to other stimuli. Future applications (such as those written for a mobile device) are not likely to simply attempt to predict eye movements. They will try to extract meaning from patterns produced by other, more interesting, stimuli. These could include being shown something that elicits emotion or tracking attention level to content. Models used here to predict something as simple as eye movement could serve as a starting point to predicting more complex user phenomena.

**REFERENCES**

Chambayil, Brijil, and Singla, Rajesh, and Jha R. “EEG Eye Blink Classification Using Neural Network.” Proceedings of the World Congress on Engineering Vol 1 (2010)

Chumerin, Nikolay, and Manyakov, Nikolay, and Van Vliet, Marijn, and Robben, Arne, and Combaz, Adrien, and Van Hulle, Marc. “Steady-State Visual Evoked Potential-Based Computer Gaming on a Consumer-Grade EEG Device.” IEEE Transactions on Computational Intelligence and AI in Games 5.2 (2012): 100-110.

Delome, Arnaud, and Sejnowski, Terrence, and Makeig, Scott. “Enhanced Detection of Artifacts in EEG Data Using Higher-Order Statistics and Independent Component Analysis”. Neuroimage 34.4 (2007): 1443-1449.

Garret, Deon, and Peterson, David, and Anderson, Charles, and Thaut, Michael. “Comparison of Linear, Nonlinear, and Feature Selection Methods for EEG Signal Classification.” IEEE Transactions on Neural Systems and Rehabilitation Engineering 11.2 (2003): 141-144.

Grimm, Frank. “Consumer EEG Blink Detecion”. blog.frankgrimm.net. May 23, 2012.

Joyce, CA., and Gorodnitsky, IF., and Kutas, M. “Automatic Removal of Eye Movement and Blink Artifacts from EEG Data Using Blind Component Seperation”. Psychophysiology 41 (2004): 313-325.

Makeig, Scott, and Bell, Anthony, and Jung, Tzyy-Ping, and Sejnowski, Terrence. “Independent Component Analysis of Electroencephalographic Data.” Advances in Neural Information Processing Systems 8 (1996): 145-151.

Man, de Jeroen. “Analysing Emotional Video Using Consumer EEG Hardware.” Human-Computer Interaction 8511 (2014): 729-738.

Tiganj, Zoran, and Mboup, Mamadou, and Pouzat, Christophe, and Lofti, Bouchra. “An Algebraic Method for Eye Blink Artifacts Detection in Single Channel EEG Recordings.” 17th International Conference on Biomagnetism Advances – BioMag (2010): 175-178

Vidaurre, C., and Shlogl, A., and Cabeza, R., and Scherer, R., and Pfurtscheller, G. “Study of On-Line Adaptive Discriminant Analysis for EEG-Based Brain Computer Interfaces.” IEE Transactions on Biomedical Engineering 54.3 (2007): 550-555.

Vjvarada, “The Mastermind Project – Mind Controlled Robots Using EEG”. decibel.ni.com. Oct 15, 2012