Data Analysis of EMG Responses of Hand Gestures Using Matlab and R

Hector Estrella, Hamid Shahnasser
School of Engineering
San Francisco State University
San Francisco, California
hestrell@mail.sfsu.edu

***Abstract*— Electromyography (EMG) sensors have been used to detect hand gestures and muscle movements. One of the most important aspects of hand gesture recognition is reducing the time to recognize specific hand movements. In this paper three EMG sensors will be used in the right hand to recognize four hand gestures: rest, supination, wrist extension, and bicep flex. This project uses Matlab to recognize the 4 hand gestures. To improve the separation of the four hand gestures, feature classification will be used, specifically: mean absolute value, waveform length, zero crossings, and slope sign changes. Classification Learner will be used to train and test data and use data analysis techniques, specifically: Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K nearest neighbors (KNN), and Tree. MATLAB and R will both be used to find which of the 4 data analysis techniques are fastest and most accurate. Then, R will be used to find the minimum data required to train our data and get accurate separation from our four hand gestures. By utilizing R and realizing the minimum training data required, the recognition time achieving 100% testing results using KNN with *N* = 10 of the 4 hand gestures decreased the runtime by 10%.**

***Index Terms*—EMG Sensors, supination, wrist extension, bicep flex, MATLAB, feature classification, mean average value, signal length, turns ratio, slope sign changes, Data Learner, Linear Discriminant Analysis, Quadratic Discriminant Analysis, K nearest neighbors, Tree, R.**

I. INTRODUCTION

Electromyography (EMG) sensors have been used to observe muscle movements in the hand. The ultimate goal has been to find the lowest possible recognition time of hand gestures, so that they can be used for real time robotic hand movements.

II. PROJECT METHODS

Matlab can be programmed to compile hand gesture data and then analyze the data using Classification Learner. Classification Learner is limited in that once the data has been programmed into training and testing, Classification Learner cannot change which observations will be used for training and testing. R has no such restriction: it can use as little or as much of the data for training and testing for data analysis. If the goal is to find the lowest possible time to achieve 100% testing results for data analysis, then Matlab in conjunction with R is one possible way to go. Chen [1] has used EMG sensors for hand gesture recognition. He defined many hand gestures. This report will use 4 hand movements: rest, wrist extension (wext), supination (sup), and bicep flex (bflex). The hardware will consist of 3 EMG sensors connected to 3 inputs of an Arduino. The Arduino will be connected to a laptop. Matlab will be used to collect the muscle sensor readings. Once the readings have been collected, Matlab will be programmed for feature extraction of the data. Feature extraction has been used by Phinyomark [2] and it will be utilized in this experiment. I will then use the built in Matlab program called Classification Learner to use 4 data analysis techniques: LDA, QDA, KNN, and Tree to separate the 4 hand gestures. I will then use R to find the fastest compilation time by using the least amount of training data. Once the EMG sensor data is collected, Classification Learner is used to train and test the data using various Data Analytical Techniques. The best outcome is to run Matlab with the Classification Learner and get 100% recognition of the hand movements in the fastest time. R will be used to find the lowest amount of training data needed to get 100% testing results. In the next sections this paper describes the proposed process of the data analysis techniques, including hardware and software descriptions. Then, this paper ends with a conclusion and final remarks for future works.

III. DATA ANALYSIS TECHNIQUES

In this project, an inexpensive system is developed that will utilize the Arduino microcontroller, 3 EMG sensors, batteries for EMG sensors, a laptop and the student version of Matlab 2019a. The student version of Matlab may have slower compilation times than the complete version of Matlab, and that needs to be considered.

A. Feature Extraction

Feature Extraction has been proposed by Phinyomark [2] for data analysis using EMG sensors. Phinyomark [2] listed 20 feature extractions. This project will utilize the following:

$$\text{Mean absolute value (MAV)} \quad \text{MAV} = \frac{1}{N}\sum_{n=1}^{N}\left|x_n\right|$$

Waveform length (WL)

$$WL = \sum_{n=1}^{N-1} \left| x_{n+1} - x_n \right|$$

Zero crossing (ZC)

$$ZC = \sum_{n=1}^{N-1} \left[\operatorname{sgn}\left(x_n \times x_{n+1}\right) \cap \left| x_n - x_{n+1} \right| \geq threshold \right];$$

$$\operatorname{sgn}(x) = \begin{cases} 1, & if\ x \geq threshold \\ 0, & otherwise \end{cases}$$

Slope Sign Change (SSC)

$$SSC = \sum_{n=2}^{N-1} \left[f\left[\left(x_n - x_{n-1}\right) \times \left(x_n - x_{n+1}\right) \right] \right];$$

$$f(x) = \begin{cases} 1, & if\ x \geq threshold \\ 0, & otherwise \end{cases}$$

3 EMG sensors (Biceps, Front, Back) and 4 features (MAV, Waveform Length, Zero crossings and number of slope sign changes) for each sensor gives 12 total features. A window length (WL) and Window Increment (Winc) is initially set in the Matlab program to do feature extraction. Initially, WL = 100 and Winc = 2 is used. This will give a total of 1608 training observations and 32 kB to compile. WL will be increased to lower the training observations. Lower observations mean a faster run time. The 12 features that will be used in this project are: Biceps_mav, Biceps_wl, Biceps_zc, Biceps_ssc, Front_mav, Front_wl, Front_zc, Front_ssc, Back_mav, Back_wl, Back_zc, Back_ssc. Mean absolute value has been abbreviated *mav*, waveform length is *wl*, zero crossings is *zc*, and slope sign changes is *ssc*.

B. *Data Analysis: Linear Discriminant Analysis (LDA)*

We will be using LDA for data analysis to separate our classes (rest, wrist extension, supination, and bicep flex). LDA can separate the classes by hypothesizing that the observations are from a Gaussian distribution and making estimates of the following mean and variance distributions [3]:

$$\hat{u}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

Mean

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{u}_k)^2$$

Variance

In the equations, n is the amount of training observations and n_k is the amount of observations in the kth class.

The probability class estimation is:

$$\hat{\pi}_k = \frac{n_k}{n}$$

An observation $X = x$ is then made to the class for which the Discriminant Function is largest.

$$\hat{\delta}_k(x) = x \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{u}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

Discriminant Function of LDA

LDA is trying to cluster the data points of the classes together by using their mean and variance. This will also separate the classes from each other. A graph of LDA analysis is shown in Figure 13.

C. *Data Analysis: Quadratic Discriminant Analysis (QDA)*

Another way to separate classes is by using a Gaussian distribution as in LDA, but not making estimates of the mean and variance [3].

The Bayes classifier will assign observation $X = x$ to the class for which the discriminant function is largest.

$$\delta_k(x) = -\frac{1}{2}(x - u_k)^T \sum_{k}^{-1} (x - u_k) - \frac{1}{2} \log \left| \sum k \right| + \log \pi_k$$

$$= -\frac{1}{2} x^T \sum_{k}^{-1} x + x^T \sum_{k}^{-1} u_k - \frac{1}{2} u_k^T \sum_{k}^{-1} u_k - \frac{1}{2} \log \left| \sum k \right| + \log \pi_k$$

Discriminant Function of QDA

QDA is trying to cluster the data points of the classes together by using their mean and variance. This will also separate the classes from each other.

D. *Data Analysis: K nearest neighbors (KNN) with N = 10*

KNN is trying to separate the 4 classes. For positive integer K and a test observation x_0, it finds the K points in the training data nearest to $x_0 = N_0$. KNN will use the Bayes classifier [3]:

$$\Pr(Y = j | X = x_0)$$

KNN will then apply Bayes rule and classify the test observation x_0 to the class with largest probability:

$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j)$$

If $N = 10$ is used, KNN will classify observation x_0 by considering its 10 nearest neighbors and if observation x_0 is next to a plurality of its nearest neighbors, observation x_0 will be correctly assigned.

E. *Data Analysis: Tree*

We have a total of 12 predictors or features in the data. The data will be divided by as little of the predictors as possible. As an example, a decision tree will be made in which a certain value of one of the features will separate one of the classes from the other classes. Figure 16 shows a decision tree example: when biceps.mav is greater than 3.1391, *Biceps Flex* can be separated from the rest of the data. When front.mav is greater than 2.5748, *Wrist Extension* can be separated from the rest of the data. When back.mav is greater than 2.5076, *Supination* can be separated from the rest of the data. The only data left is *Rest.* So, when back.mav is less than 2.5076, *Rest* is separated from the rest of the data.

F. Theory on LDA, QDA, KNN and Tree

I. If the data is linear, then LDA will be more accurate than QDA and QDA will overfit the training data and give worse results in testing than training.

II. If the data is quadratic, QDA will be more accurate than LDA.

III. As the number of observations decrease, then LDA should give better results. KNN gives better results when N is larger. We will use $N = 10$.

IV. Tree gives very accurate results, but the calculations are longer than LDA, QDA, and KNN. Tree will take a lot longer to compute when the predictors are very large. In this project, the predictors = 12, which is very small. A large number of predictors (N > 1000) will take a lot of programming runtime, lasting several hours.

IV. HARDWARE COMPONENTS

This project utilized 3 EMG sensors, 3 power shields, 3 batteries for the EMG sensors, the Arduino microcontroller, and a laptop.

Figure 1. Rest Motion. The "Front" EMG sensor is in the Extensor Digitorim muscle region.

Listed below is the description of the Hardware components used in this project.

A. 3 EMG Sensors

The MyoWare Muscle Sensor is inexpensive costing $38. It has connections for Raw EMG Output. 3 Raw EMG Outputs will be collected. It also has LED indicators that brighten when the muscles flex. It needs biomedical sensor pads.

B. Sensor pads

Biomedical sensor pads are needed to connect the muscle and the ground of a hand or arm (see Figure 1). A pack of 50 costs $25. One of the issues with using EMG sensors is loose contact [3]. It is imperative to change the sensor pads frequently.

C. Myoware Power Shield with coin cell battery

To power the muscle sensors, 3 Myoware power shields will be used. They cost $4.50 each. They will need coin cell batteries, which cost $2.00 each.

D. Arduino Duemilanove

The Arduino is a well-known inexpensive microcontroller. 3 outputs of the EMG sensor will be connected to 3 inputs and the ground of the Arduino. The Arduino can be used in real time to collect the raw EMG sensor data. It will be connected to our laptop.

E. Laptop

Collection of the EMG sensor data as well as the Matlab programs were run using a laptop PC. The PC was running 64 bit Windows 10 Home with a 1.6 GHz processor (Intel *i*5) and 8 GB of RAM.

V. DATA COLLECTION AND SOFTWARE COMPONENTS

For this project, we will be using Matlab to collect the EMG data from the 3 sensors, Matlab to run feature extraction on the data, Classification Learner to run Data Analysis, R to find what the lowest the number of observations can be used to get 100% testing results, and Classification Learner again to get the fastest results for our Data Analysis.

A. Data Collection

3 trials were collected for each hand motion with 3 degrees of freedom and a rest class. The motions were rest, wrist extension, supination and biceps flex. The rest motion was shown in Figure 1. The other hand motions are shown below.

Figure 2. Wrist Extension: Moving the wrist from down to upward.

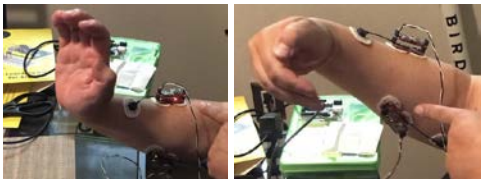

Figure 3. Supination: Moving the right wrist clockwise. The "Back" EMG sensor is in the Brachioradialis muscle region.

Figure 4. Bicep Flex: Moving the right forearam upward. The "Biceps" EMG sensor is in the Biceps Brachii muscle region.

Matlab was used to run a simple code for feature extraction of the sensor data from the 3 EMG sensors, at a collection frequency of 25 Hz. 3 trials were collected for each activity. 500 data points were collected for each trial. The trials took 20 seconds.

B. Graphs of 3 EMG sensor values

Matlab was used to show the EMG sensor data output for the first trial. The data will be listed as rest, wext for wrist extension, sup for supination, and bflex for biceps flex.

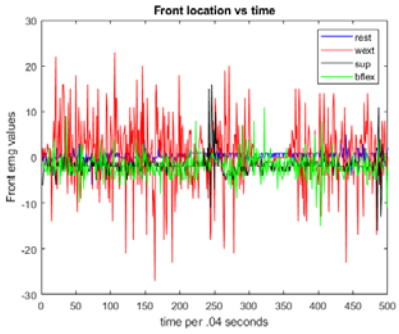

Figure 5. Front EMG vs time. Wext has largest EMG values.

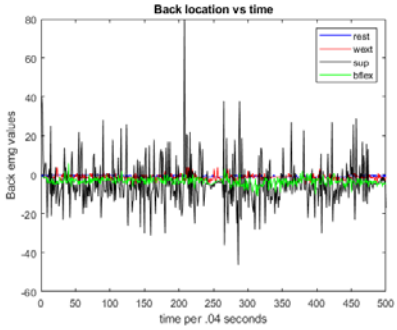

Figure 6. Back EMG vs time. Sup has largest EMG values.

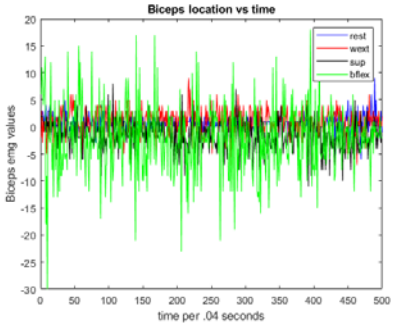

Figure 7. Biceps EMG vs time. Bflex has largest EMG values.

C. *MATLAB feature extraction program*

The Matlab program will run feature extraction of the EMG data of the 3 trials. Initially, we set trials 1 and 2 for training and trial 3 for testing. Thus, 67% of the data will be used for training and 33% of the data will be used for testing. Initially, we set WL = 100 and Winc = 2. After the program is run, this gives us 12 predictors or features for feature extraction with 1608 observations and 32 kB of data to compile. If the training data gives good results in Classification Learner, then we can increase Winc in our program to lower the observations, since lower observations decrease the Matlab feature extraction program runtime.

D. *Classification Learner*

Classification Learner is a built in program in Matlab. It can be used for data analytical techniques. It has more than 20. We will be using LDA, QDA, KNN = 10, and Tree for the 12 features in this project. Initially, 2 trials were used for training and 1 trial for testing.

Initially, WL = 100 and Winc = 2. Since good test results were achieved, Winc was increased to 5, 10 and 20.

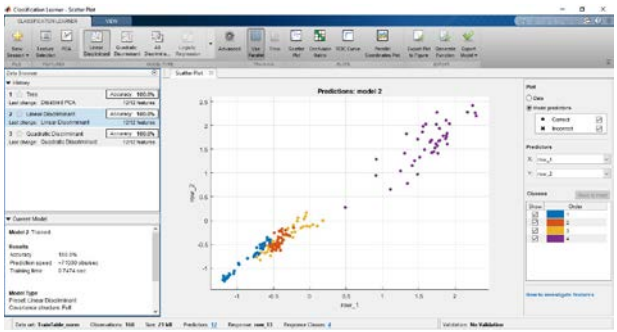

Figure 8. LDA for 2 training (67%), 1 test (33%), 100 = WL, Winc = 20, 168 observations, 21 kB, time is 0.747 secs. For classes: 1 is rest, 2 is Wext, 3 is Sup, 4 is Bflex.

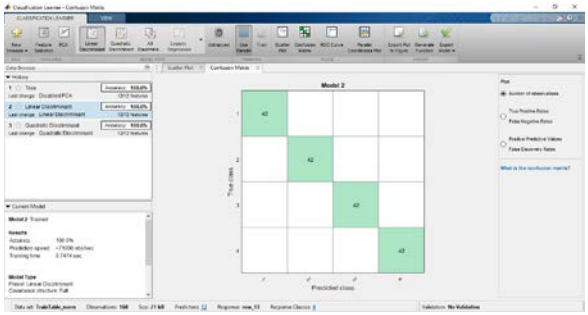

Figure 9. Confusion Matrix for LDA for 2 train (67%), 1 test (33%), WL = 100, Winc = 20, 100% training results.

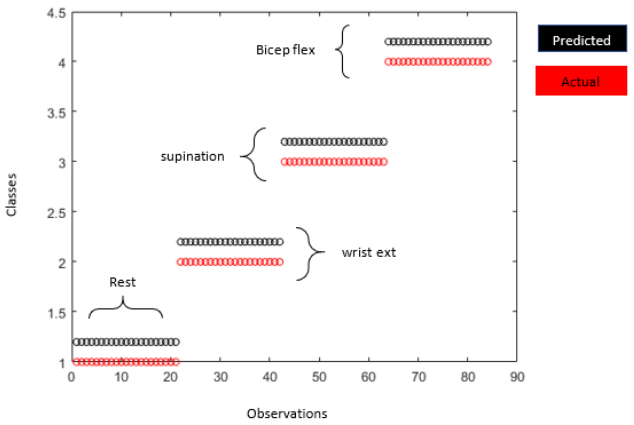

Figure 10. Test result for LDA, 2 training (67%), 1 testing (33%), WL = 100, WInc = 20, normalized. The test results achieved 100% accuracy.

E. *Using R to find the lowest possible training set*

R is a program used by Data Scientists and Mathematicians for Data Analysis. It has built in functions for data analysis. It will be utilized to find the lowest possible training data to get 100% accuracy on either LDA, QDA, KNN, or Tree. When using WL = 100 and Winc = 20, there are a total of 252 total observations on the 3 sets (84 for each set). We have split them up in Matlab using 2 training (67%) and 1 test set (33%). R can randomly split the 252 observations into a percentage for

training and testing. Classification Learner cannot randomly split the observations into training and testing. To run faster Matlab compilation time, we can use the lowest possible training set found in R until we get less than 100% accuracy.

VI. Experimental Results

A. R Experimental Results

R was run with different percentage training data using WL = 100 and Winc = 20. The following are the experimental results for 10 random tests for LDA, QDA, KNN and Tree.

% of Total Data	LDA		QDA		KNN	N = 10	Tree	
	Testing Accuracy	Error Rate	Testing Accuracy	Error Rate	Testing Accuracy	Error Rate	Testing Accuracy	Error Rate
67	100	0	99.76	0.24	99.76	0.24	99.52	0.48
33	100	0	96.69	3.31	99.65	0.35	99.29	0.71
30	100	0	97.86	2.14	99.55	0.45	99.44	0.56
28	100	0	97.36	2.64	99.67	0.33	98.35	1.65
27	99.89	0.11	97.67	2.33	99.13	0.87	99.35	0.65

Figure 11. Accuracy of 4 LDA, QDA, KNN and Tree

The results show that LDA was giving 100% testing accuracy until it reached 27% of the data. KNN with N = 10 averaged 99.13% accuracy and Tree averaged 99.35% accuracy with 27% training data. This is an average of the 10 random tests and I was able to get 100% accuracy for 6 of the 10 tests for both Tree and KNN with $N = 10$.

Here are the results for 2 of the random data sets using LDA.

68 training (top) and
184 testing (bottom) observations

```
lda.train_class bflex rest sup wext
          bflex    20    0   0    0
          rest      0    9   0    0
          sup       0    0  17    0
          wext      0    0   0   22

lda.class bflex rest sup wext
    bflex    43    0   0    0
    rest      0   54   0    0
    sup       0    0  46    0
    wext      0    0   0   41
```

68 training (top) and
184 testing (bottom) observations

```
lda.train_class bflex rest sup wext
          bflex    19    0   0    0
          rest      0   14   0    0
          sup       0    0  18    0
          wext      0    0   0   17

lda.class bflex rest sup wext
    bflex    44    0   0    0
    rest      0   49   0    0
    sup       0    0  45    0
    wext      0    0   0   46
```

Figure 12. Confusion Matrix for the test results for 2 random data sets for LDA for WL = 100, Winc = 20 at 27% of total data using R. LDA gives 100% testing results.

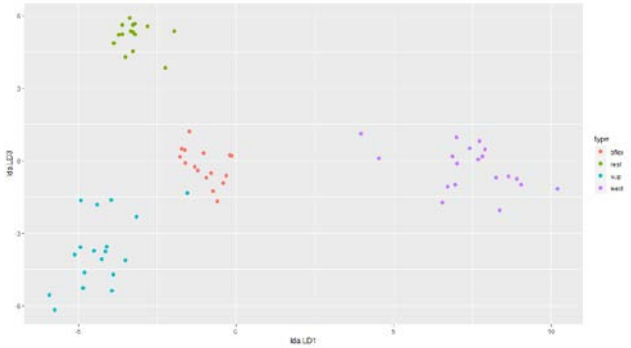

Figure 13. Separation of the 4 classes for 1 random data set for LDA for WL = 100, Winc = 20 at 27% of total data using R. LDA gives 100% testing results.

Below are the results for 2 more random data sets using KNN.

68 training observations
184 testing observations
Mean = .99456
Error rate = .005435

```
knn.pred bflex rest sup wext
   bflex    40    0   0    0
   rest      0   51   0    0
   sup       0    0  46    0
   wext      0    0   1   46
```

68 training observations
184 testing observations
Mean = .99456
Error rate = .005435

```
knn.pred bflex rest sup wext
   bflex    43    0   0    0
   rest      0   54   0    0
   sup       0    0  45    0
   wext      0    0   1   41
```

Figure 14. Confusion Matrix for the test results for 2 random data sets for KNN for N = 10, WL = 100, Winc = 20 at 27% of total data using R. KNN for N = 10 gives 99.5% testing results for both.

Below are the results for 2 random data sets using Tree.

68 training observations
184 testing observations
Mean = .97826
Error rate = .0217

```
tree.pred bflex rest sup wext
   bflex    42    0   0    0
   rest      1   54   0    3
   sup       0    0  46    0
   wext      0    0   0   38
```

68 training observations
184 testing observations
Mean = .99456
Error rate = .0054

```
tree.pred bflex rest sup wext
   bflex    39    0   0    0
   rest      1   51   0    0
   sup       0    0  47    0
   wext      0    0   0   46
```

Figure 15. Confusion Matrix for the test results for 2 random data sets for Tree WL = 100, Winc = 20 at 27% of total data using R. Tree gives 97.8% and 99.5% testing accuracy.

The tree structure in R for 2 random data sets are shown below.

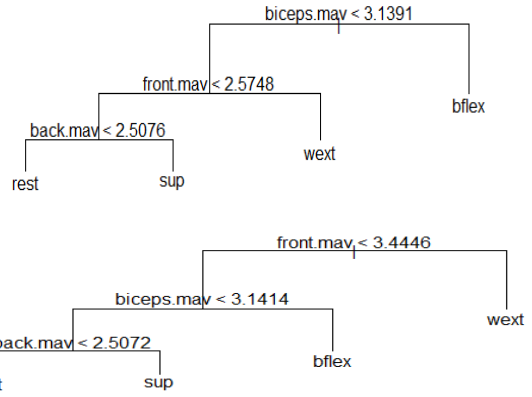

Figure 16. Tree structure for 2 random data sets for Tree utilizing WL = 100, Winc = 20 at 27% of the total data using R. Tree gives 97.8% and 99.5% testing results, top and bottom, respectively.

B. Classification Learner Results

Previously, we had assigned our trials with 2 training sets and 1 test set. The 2 training sets give us 67% of the total data set, but R has also shown that if the training set is changed to 27% for training and 73% for testing (see Figure 11), one may still get 100% testing results for Classification Learner. The training and testing set were changed in the Matlab program accordingly.

Increasing Winc decreases the training observations. If Winc can be increased to Winc = 20 when only one trial set is used as a training set, the number of observations decreases to 84. According to R, this should still give 100% training results for LDA, and this will be proved using Classification Learner. The lower the number of observations in the training set should also decrease the total programming runtime, since only 12 kB

are now used. The Classification Learner runtime has been approximately .77 seconds throughout the decrease in observations. Classification Learner is fast and can process as high as 700,000 observations a second. Thus, since our observations are less than 1700, this barely changes the Classification Learner's compilation time for LDA, QDA, KNN, and Tree. This can be shown on Figures 20 and 21.

Finally, our 3 trial tests are modified to get 27% training and 73% testing. The trial set will be decreased from 500 data points to 420 data points and 40 data points each were added from trial 1 to trials 2 and 3. The new trial 1 was the new training set and new trials 2 and 3 became the new testing sets. Implementing this in the MATLAB feature extraction program will give 27% training data. The MATLAB feature extraction program and Classification Learner will then be rerun. Our results are shown below.

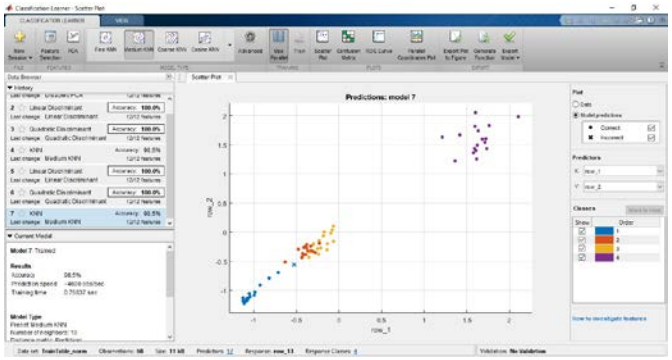

Figure 17. KNN with *N* = 10 for 27% training, 73% testing, WL = 100, Winc = 20, 68 observations, 11 kB, time is 0.766 secs. For classes: 1 is rest, 2 is Wext, 3 is Sup, 4 is Bflex.

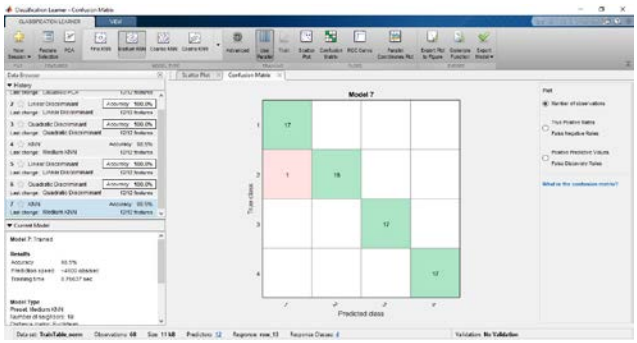

Figure 18. Confusion Matrix for KNN with *N* = 10 for 27% training, 73% testing, WL = 100, Winc = 20. 98.5% training results.

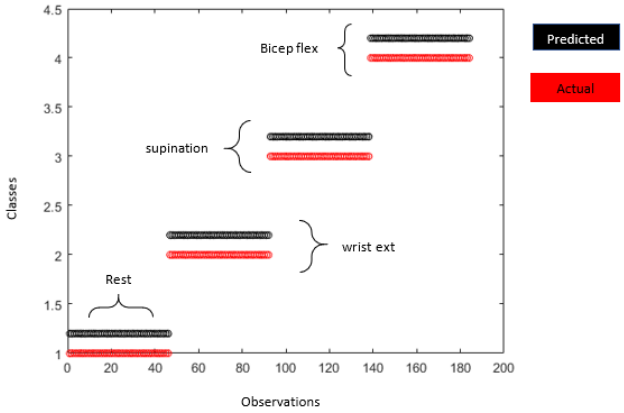

Figure 19. Test result for KNN with N = 10, 27% training, 73% testing, WL = 100, Winc = 20, normalized. The test results give 100% accuracy.

VII. Final Results

A. Summary of Classification Learner Results

The following Classification Learner results are now summarized below.

		Training Accuracy %	Testing Accuracy %	Program Time seconds	Compile Time seconds	Total Time seconds
1	LDA; WL = 100, Winc = 2; 1608 obs; 167 kB; 67% training	100	100	11.164	0.845	12.009
2	QDA; WL = 100, Winc = 2; 1608 obs; 167 kB; 67% training	100	100	11.164	0.777	11.941
3	KNN; WL = 100, Winc = 2; N = 10 1608 obs; 167 kB; 67% training	99.9	100	11.164	0.759	11.923
4	Tree; WL = 100, Winc = 2; 1608 obs; 167 kB; 67% training	100	100	11.164	0.767	11.931
5	LDA; WL = 100, Winc = 5; 648 obs; 70 kB; 67% training	100	100	5.414	0.831	6.245
6	QDA; WL = 100, Winc = 5; 648 obs; 70 kB; 67% training	100	100	5.414	0.743	6.157
7	KNN; WL = 100, Winc = 5; N = 10 648 obs; 70 kB; 67% training	100	100	5.414	0.776	6.19
8	Tree; WL = 100, Winc = 5; 648 obs; 70 kB; 67% training	100	100	5.414	0.772	6.186
9	LDA; WL = 100, Winc = 10; 328 obs; 37 kB; 67% training	100	100	3.2	0.825	4.025
10	QDA; WL = 100, Winc = 10; 328 obs; 37 kB; 67% training	100	99.39	3.2	0.771	3.971
11	KNN; WL = 100, Winc = 10; N =10 328 obs; 37 kB; 67% training	100	100	3.2	0.763	3.963
12	Tree; WL = 100, Winc = 10; 328 obs; 37 kB; 67% training	100	100	3.2	0.76	3.96

Figure 20. Summary of Classification Learner Results.

		Training Accuracy %	Testing Accuracy %	Program Time seconds	Compile Time seconds	Total Time seconds
13	LDA; WL = 100, Winc = 20; 168 obs; 21 kB; 67% training	100	100	2.16	0.712	2.872
14	QDA; WL = 100, Winc = 20; 168 obs; 21 kB; 67% training	100	100	2.16	0.771	2.931
15	KNN; WL = 100, Winc = 20; N = 10 168 obs; 21 kB; ; 67% training	100	100	2.16	0.763	2.923
16	Tree; WL = 100, Winc = 20; 168 obs; 21 kB; 67% training	100	100	2.16	0.76	2.92
17	LDA; WL = 100, Winc = 20; 84 obs; 12 kB; 33% training	100	100	2.038	0.76	2.798
18	QDA; WL = 100, Winc = 20; 84 obs; 12 kB; 33% training	99.3	85.12	2.038	0.745	2.783
19	KNN; WL = 100, Winc = 20; N = 10 84 obs; 12 kB; 33% training	100	100	2.038	0.765	2.803
20	Tree; WL = 100, Winc = 20; 84 obs; 12 kB; 33% training	100	80.95	2.038	0.771	2.809
21	LDA; WL = 100, Winc = 20; 68 obs; 11 kB; 27% training	100	98.37	1.902	0.749	2.651
22	QDA; WL = 100, Winc = 20; 68 obs; 11 kB; 27% training	100	76.09	1.902	0.777	2.679
23	KNN; WL = 100, Winc = 20; N =10 68 obs; 11kB; 27% training	98.5	100	1.902	0.765	2.667
24	Tree; WL = 100, Winc = 20; 68 obs; 11 kB; 27% training	100	82.07	1.902	0.758	2.66

Figure 21. Summary of Classification Learner Results.

Matlab and Classification Learner were rerun giving results of 100% testing for KNN with N = 10 while the total time decreased to 2.67 seconds with only 68 observations.

B. Timing when lowering observations

The following are the total runtimes in the Matlab program and Classification Learner.

	1608 obs WL = 100 67% train	167 KB Winc = 2	648 obs WL = 100 67% train	70 KB Winc = 5	328 obs WL = 100 67% train	37 KB Winc = 10	168 obs WL = 100 67% train	21 KB Winc = 20	84 obs WL = 100 33% train	12 KB Winc = 20	68 obs WL = 100 27% train	11 KB Winc = 20		
	Total Time seconds	Faster	Total Time seconds	Faster than 1608 obs	Total Time seconds	Faster than 648 obs	Total Time seconds	Faster than 328 obs	Total Time seconds	Faster than 168 obs	Total Time seconds	Faster than 84 obs	Faster than 168 obs	Faster than 328 obs
LDA	12.009	N/A	6.245	1.92	4.025	1.55	2.872	1.40	2.798	1.03	2.651	1.06	1.08	1.52
QDA	11.941	N/A	6.157	1.94	3.971	1.55	2.931	1.35	2.783	1.05	2.679	1.04	1.09	1.48
KNN with N = 10	11.923	N/A	6.19	1.93	3.963	1.56	2.923	1.36	2.803	1.04	2.667	1.05	1.10	1.49
Tree	11.931	N/A	6.186	1.93	3.96	1.562	2.92	1.36	2.81	1.04	2.66	1.06	1.10	1.49

Figure 22. KNN runtime decreased 10% when going from 168 observations to 68 observations. KNN runtime was 49% faster when going from 328 observations to 68 observations.

C. Analysis of Results

I. MATLAB with Classification Learner was used for Data Analysis of 4 hand movements (Rest, Wrist Extension, Supination, and Bicep Flex).

II. R was used to find the lowest possible observations to get 100% testing results for random observations. The lower limit was 27% training (68 observations) and 78% testing (184 observations) for LDA in R. An average of 99.1% testing results for KNN with N = 10 in R during 10 random tests was achieved. 6 of the tests gave 100% testing results for KNN with N = 10.

III. Classification Learner with 27% training data achieved 100% testing results for KNN with N = 10.

IV. At our lowest limit predicted in R (68 observations), lowering from 84 to 68 training observations gave results of 5% faster for KNN with N = 10.

VIII. CONCLUSION

This project utilized Matlab, Classification Learner and R in EMG sensor analysis, including Feature Extraction. Here are 5 conclusions from our results:

I. MATLAB was used to program Feature Extraction of our 4 hand gestures.

II. Classification Learner was used to compile our 12 predictors.

III. R can be used to find the lowest limit of training data needed to get 100% results.

IV. By utilizing the testing results of R, the lower limit was reduced from 168 observations to 68 observations.

V. At our lowest limit, we found that the total time was 2.67 seconds for KNN with N = 10.

VI. Lowering from 168 to 68 training observations decreased the runtime 10% for KNN with N = 10.

For future works, the MATLAB programming time needs to be decreased to make it more viable for real time implementation of EMG sensor analysis. A faster computer will probably lower the runtime as well as using the full version of MATLAB 2019a instead of the student version. Still, it has been shown that using MATLAB, Classification Learner, and R can be utilized to find the minimum runtimes to get 100% training results for the 3 EMG sensors and these techniques can be used for future EMG sensor data analysis.

REFERENCES

[1] Chen, Xiang, et al. "Hand gesture recognition research based on surface EMG sensors and 2D-accelerometers." *2007 11th IEEE International Symposium on Wearable Computers*. IEEE, 2007.

[2] Phinyomark, A., et al. "Evaluation of EMG feature extraction for movement control of upper limb prostheses based on class separation index." *5th Kuala Lumpur International Conference on Biomedical Engineering 2011*.

[3] James, Gareth, et al. *An introduction to statistical learning*. Vol. 112. New York: Springer, 2013.

[4] Lotte, Fabien. "A new feature and associated optimal spatial filter for EEG signal classification: Waveform Length." *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012.

[5] Zhang, Xiaorong, He Huang, and Qing Yang. "Real-time implementation of a self-recovery EMG pattern recognition interface for artificial arms." *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2013.

[6] Teetor, Paul. "R Cookbook: Proven Recipes for Data Analysis." *Statistics, and Graphics. Cambridge: O'Reilly Media* (2011).

[7] Zhang, Xiaorong, and He Huang. "A real-time, practical sensor fault-tolerant module for robust EMG pattern recognition." *Journal of neuroengineering and rehabilitation* 12.1 (2015): 18.

[8] Wickham, Hadley, and Garrett Grolemund. *R for data science: import, tidy, transform, visualize, and model data*. " O'Reilly Media, Inc.", 2016.

Your Name	Position	Research Field	Personal Webpage
Hector Estrella	Master Student	EMG data analysis: Matlab, R	http://github.com/lieben4uandme
Hamid Shahnasser	Professor	Communication Networks	http://engineering.sfsu.edu/erdc/faculty_profile/shahnasser.html