Classification of EMG Hand Motion Data Using Neural Networks and Random Forest Algorithms

Tayla Goodmanson, Thomas Wuertz, Victor Chuol, Alex Jonas

University of Minnesota Duluth

Abstract

This study tested the hypothesis that, given data for EMG hand motion sensor readings taken every five milliseconds for forty milliseconds, our team would be able to predict the end hand gesture class of other given data better than a random guesser. Data was collected using a MYO armband which consisted of eight sensors and was administered around the forearm. Total data collected consisted of 11,674 observations divided almost evenly between four gesture types: 'rock', 'paper', 'scissors', and 'ok'. Three separate prediction techniques were used consisting of K-Means Clustering, Feed-Forward Neural Networks, and Random Forest Algorithms. Final results demonstrated prediction rates above guessing (25% accuracy), with K-Means having at best 32.98% accuracy, Random Forest having 59%, and Feed-Forward Neural Network having 47.02%.

Introduction

Our project analyzed data from EMG (electromyography) muscle sensors to predict hand movements. This work is important in the industry of prosthetics. Our data set consisted of four files, each corresponding to a different hand movement: "rock", "paper", "scissors", and "ok". There were sixty-four columns of data from eight EMG sensors taken over a span of forty milliseconds, with data collected from the sensors every five milliseconds. Each file consisted of approximately three thousand data samples resulting in a grand total of about twelve thousand data samples.

The objective of our project was predicting a hand gesture depending upon the initial sensory input. The team's goal was to take the data samples and create three algorithms that attempted to correctly predict what sensory input resulted in a certain hand gesture. Simply stated, if the team were given sensory input with no hand gesture, then we can correctly predict the hand gesture at an accuracy rate greater than 25%, which is the expected probability if we were to make a random guess about a given input's classification.

The three algorithms employed to predict hand gestures given a random data sample were K-Means Clustering, Neural Network, and Random Forest Algorithm, the latter implemented by hand.

Similar research has been conducted in the last decade, yielding positive results in classifying EMG gesture data, as discussed below. The importance around the data is understanding which sensors contribute to which muscular movement. With more concise knowledge on muscular movement, artificial muscular movement within prosthetics will continue to become more robust and will ultimately better serve an individual in need of a prosthetic.

This report will start by discussing work related to classification of EMG sensory data. We will then delve into how we accomplished our proposed objective of hand gesture classification given only sensory input by first discussing the hardware used when collecting EMG data. Following that we will discuss how we cleaned and grouped the data in order to make the data samples understandable. Next, we will discuss in detail our implementation of K-Means Clustering, Random Forest, and Neural Networks and analyze how effective they were in classifying hand gestures. Finally, we will present our team's conclusions, along with proposing future work within the topic of EMG classification.

Related Work

Researchers have implemented several methods to develop systems of prediction for hand gestures. Some methods rely heavily on neural networks, such as convolution and feedforward artificial networks (Atzori et al. 2016, Zhang et al. 2019). Other researchers have used methods such as random forest algorithms, transfer learning algorithms, fuzzy logic algorithms, and k-nearest neighbors (Atzori et al. 2016, Côté-Allard et al. 2019, Karuna et al. 2018, Zhang et al. 2019). While our project also includes random forest algorithms, we differ by using clustering analysis.

Studies on gesture prediction using EMG muscle data use a variety of hand movements as well as participants within their analysis. A few notable studies have used between five and seven hand gestures employing at least five test subjects during data collection

(Côté-Allard et al. 2019, Karuna et al. 2018, Zhang et al. 2019). A different study used fifty different hand movements and almost eighty participants (Atzori et al. 2016). Our data set differs from the other studies as it focuses only on four gestures that are being held in place by a single subject. Data collection among studies also differ as retrieval of data in two studies employed sensory collection from the wrist, finger, and hand of a subject (Atzori et al. 2016, Karuna et al. 2018) while two other studies (as well as the data we will be using) used a MYO armband to collect EMG data (Côté-Allard et al. 2019, Zhang et al. 2019). Finally, only one study used a variety of participants that included non-injured individuals and amputees (Atzori et al. 2016). All remaining studies including the data we will be analyzing used non-injured individuals. Despite the subjects being non-injured, meaningful correlations can still be derived and used in further research on prosthetics.

Proposed Work

Our project is split into four parts:

- Pre-processing: Every sample in our dataset contained 65 different attributes. In order to deal with this high dimensionality, our dataset required different pre-processing techniques.
- Clustering: The first algorithm we proposed to classify the EMG data was K-means clustering. In this algorithm we were looking for four distinct clusters that we could classify as either “rock”, “paper”, “scissors”, or “ok”.
- Neural Networks: The second algorithm we proposed to classify our data into one of four gestures was Neural Networks using the neuralnet package available in R.
- Random Forest from scratch: The final algorithm we proposed was creating and testing a Random Forest algorithm written from scratch. This means that no Random Forest packages were used when running this algorithm.

Pre-Processing

The first step we took in pre-processing the data was checking for NA’s. We found that all four of the files were devoid of NA’s, so we moved on to assigning names to each column in the data frames since they did not have any originally. Because each row contains eight different readings over 40 milliseconds from eight sensors, we chose to name each column with both their sensor and the time at which the reading was taken. Additionally, we labeled one final column in our data with the type of hand gesture the sample corresponded to. We assigned the following type values: “1” for “rock”, “2” for “scissors”, “3” for “paper”, and “4” for “ok”. This allowed us to append each of the four datasets to create one large data set, which we later refer to as the original dataset.

The next step in our data processing was critical for the rest of the project. With a total of sixty-four attributes as well as an additional attribute for the gesture type, our dataset had high dimensionality. To deal with this, we created new data sets out of the original set. First, we created an averaged data set which contains the average EMG readings for each sensor. Next, we created a totaled data set which contains the sum of the EMG readings for each sensor. Finally, we created a spread-out data set that focused on only the sensors without respect to time. These are detailed more clearly in a later section.

Proposed Clustering

To achieve our goal in predicting gesture type using clustering, we chose to implement the k-means algorithm with k=4 on three of the data sets: the original dataset, the averaged dataset, and the spread dataset. The totaled dataset was omitted due to its similarity to the averaged dataset in how it was calculated, as averages are simply a scaled form of a summation. It is also common practice to scale your data before implementing k-means; however, our data was already on the same scale so there was no need to scale any of the three data sets prior to creating the clusters.

Prior to fitting clusters, we made new copies of each of the three data sets where we removed the column containing the type. This allowed us to form clusters based only on the sensor readings and not on the type which was already known. We then created the clusters on the copied data in an attempt to predict the four gesture types.

To determine the accuracy of the clusters for each of the data sets, the clusters were compared to types listed in the original copies of the three data sets. Because the numbers assigned to the clusters through the k-means implementation may not match the numbers we assigned to each gesture type, we compared the results of the clusters to the types we assigned by determining which cluster matched up the most with a given gesture type, then assigning that particular cluster to that type. After determining which clusters matched certain gestures the most, we determined the overall accuracy of the fit for each of the three sets we implemented clustering on. If the accuracy for a given fit was over 25% (the probability of any one gesture group given a random guess), then we can conclude the resulting fit is a good fit.

Proposed Neural Network

The team created a standard feed-forward neural network in order to predict gesture type given sensory inputs. The data set used for the neural network consisted of all data sets combined with each 5ms interval of recorded data split into its own dataset. Meaning instead of 64 different sensory observations for each data set the new dataset consisted of only 8 sensory observations. This dataset was named the 'spread' dataset which is explained in a later section.

Once the spread dataset was created and shuffled it was divided with 10% of the data being used for testing and 90% used for training. The reason for splitting and shuffling the data is to ensure the neural network does not simply memorize the data set. The training technique used for the neural network was supervised learning, meaning the algorithm was given the eight sensory inputs along with gesture class. The sensory input can be thought of as showing the neural network the questions along with the gesture class being the answer.

Once the neural network has been trained, the neural network will then be tested on our test data with gesture classes removed. The neural network will output a prediction list with probabilities for each of the 4 gesture class types given each test data set. Next, we must sift through the prediction list, with the highest prediction value of the four classes being the neural networks' prediction of the type. With the prediction list sorted, we compare the prediction list to the actual test data gesture classes and create a confusion matrix of correct and incorrect predictions. Once completed and the confusion matrix is made, we can simply take the correct predictions against all predictions for our accuracy rate.

The structure we used for our neural network was eight inputs for each 5 ms interval of sensory data, a hidden layer, and four outputs corresponding to the probability of the dataset consisting of certain gesture classes.

Proposed Random Forest by Hand

As we began crafting our random forest implementation, we decided we would test our implementation data on multiple forms of data, but not the original set due to its high dimensionality. The sets we used included the totaled dataset using the original sensor data averaged among each sensor at 5ms and truncated into 8 sensor readings, as well as an averaged dataset with every 5ms totaled among the individual sensors. This implements the hypothesis that since each sensor is in the same position, the total and average readings of each sensor should result in a correlation between the sensor readings.

We also decided to take on a third dataset, the spread dataset, which when implemented minimized elongated data by treating every 5 ms as its own row entry in the data set, expanding the dataset with each interval of 5ms. This would be key to forming our training hypotheses and giving us a plethora of data to run the algorithm on. All these approaches would also give us a data set that would include eight features for us to process. This is optimal because decision trees can be hard to implement with large quantities of features.

Our implementation will work with both Rpart and C5.0 decision trees. This approach will work by analyzing decisions based on numerous decision trees. We will use thousands of decision tree predictions to form a prediction that will be an average of all aggregated predictions. The implementation will consider all features M of the data but will randomly select m features such that $m < M$.

The R implementation we develop will return a set of trees with random features selected from our shuffled training data set (type feature included). In this specific implementation we will be running analysis on a static number of features and a pseudorandom amount of features We can in turn take this set of trees and create predictions using the shuffled training data (without type feature). A mean table of the prediction tables will be used as a prediction itself using the shuffled training data (with type feature) as a comparison. With this comparison we can predict and process an error rate for our method.

Experimental Evaluation

During the implementation stage of our project, the focus was to predict four separate classifications of hand gestures using data collected from an arm band, so each algorithm was built to output one predicted grouping. To ensure the best possible fits were developed from clustering, neural networks, and our random forest algorithm implementation constructed by hand, we allowed for a variation on which data sets were used within each evaluation. We used a total of four data sets: the original data set, an averaged data set, a totaled data set, and a spread data set. The latter of the three were constructed from the original set to improve the results of our algorithms when needed. These algorithms are documented below.

Hardware

Data was collected using a MYO armband which is worn around the forearm. The armband contains 8 sensors that collect EMG data. To our knowledge, the data was collected from a single individual wearing the armband.

Data Set

The original dataset we used for experimentation contained 11,674 observations with sixty-five variables. Sixty-four of the variables represent the sensor the reading was taken from as well as the time it was taken at. There are eight readings for each of the eight sensors corresponding to eight different points in time within a forty second range. The final variable indicated the gesture type which was either "rock", "paper", "scissors", or "ok". The total observations for each of the types were as follows: 2,909 "rock", 2,942 "paper", 2,902 "scissors", and 2,921 "ok".

The averaged dataset contained the 11,674 observations taken from the original dataset but with nine variables. Eight of the variables correspond to the averages for the eight sensors. Each one was created by taking the average EMG reading for a specific sensor across the span of forty milliseconds from a single row of the original dataset. The final variable indicated the gesture type.

The totaled dataset contained the 11,674 observations taken from the original dataset with nine variables. Eight of the variables correspond to the sums for the eight sensors. Each one was created by taking the total of eight readings for a specific sensor across the span of forty milliseconds from a single row of the dataset. The final variable indicated the gesture type.

The spread dataset contained 93,392 observations with nine variables. To make this set, we took a single row of data from the original dataset and made eight new rows out of it. This was done to eliminate the aspect of time so we could focus solely on the sensors. Each new row contained eight variables for the sensors, but were formed from different times of evaluation. For example, if we consider the first row of the original data set which contains sixty-four readings as a combination of eight sensors and eight different times, we would make eight new rows out of this. The first new row would contain the sensor readings from all eight sensors at the first point in time (5 ms), then the second row would contain the readings from all eight sensors for the second point in time (10 ms), etc. A new row would be made for each interval of time that readings were taken (for a total time range of 40 ms with 5 ms intervals), and all of the new rows would have a final column indicating the gesture type that matched the type from the row they were created from.

Clustering Implementation

For clustering analysis we chose to use R's built in kmeans function with k=4 to create four clusters from our data with the hopes that each cluster developed would correspond to one of the four gesture types. After creating a fit using kmeans, we then used the fviz_cluster function from the factoextra package to create a visual for analysis. For further analysis, we calculated the accuracy of a fit by reassigning the numbers of the generated clusters to match the gesture type that a given cluster was most likely to predict by assessing which type had the highest concentration within that cluster. This process was done on three of the data sets: the original set, the averaged set, and the spread set.

Clustering on the original data yielded the clusters displayed in Figure 1. We were initially surprised to see such a low accuracy result from the clustering, which we see was 29.16% and is summarized in Table 1, but after visualizing the clusters we can see the reason for such a large error. All four of the clusters as viewed in the figure overlap significantly. The reason for this behavior is that this data set has a very high dimensionality, meaning there are too many factors to make clearer clusters on. We can also see that the "ok" cluster, which is depicted as the green region within Figure 1, is completely encased within the other three regions. As a result of this, any predictions made that should fall into the category for "ok" could easily be assigned one of the other types instead.

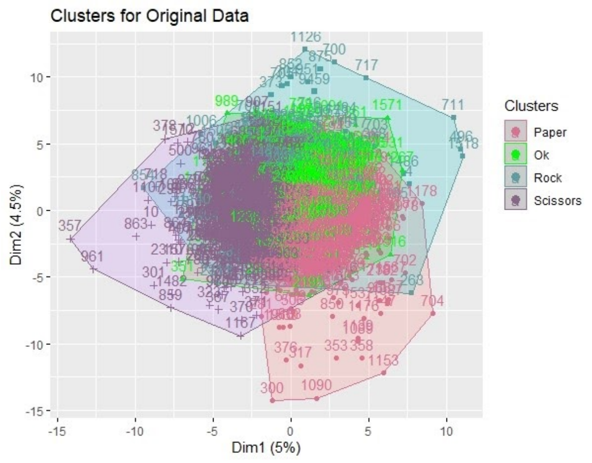

Figure 1: Plot of clusters obtained using k-means on the original dataset with k=4. The cluster numbers shown represent the gesture type that is most likely for a particular grouping.

Clustering on the averaged data provided improved results over clustering on the original data. There was a small increase in accuracy from 29.16% to 31.94% as displayed in Table 1. We can also see in Figure 2 that the individual points are much more clearly separated into distinct groups than they were in Figure 1, and there are even fewer outliers which has led to an increase in accuracy. The most likely reason for this improvement was the reduction in dimensionality, as clustering was done on eight attributes instead of sixty-four.

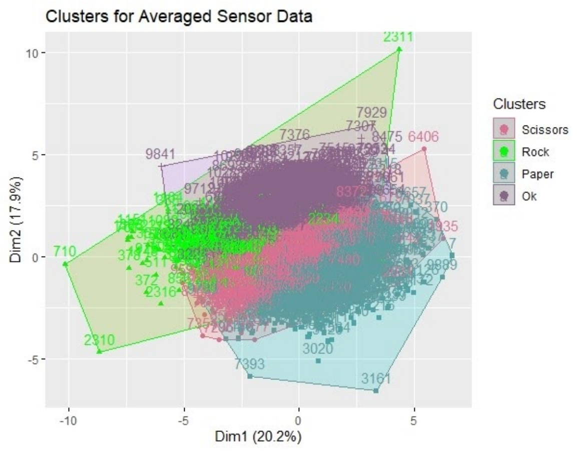

Figure 2: Plot of clusters obtained using k-means on the averaged dataset with k=4. The cluster numbers shown represent the gesture type that is most likely for a particular grouping.

Our final attempt at clustering was implemented on the spread data. This data is similar to the averaged data in that the dimensionality has been reduced to eight attributes, so we expected similar results. In Figure 3 we can see the clusters are distinct like we had seen with the averaged data; however there appears to be an overlap between the “rock” and “scissors” clusters as well as overlap between the “paper” and “ok” clusters. This result may have occurred for a few reasons, the first being the increase in observations. While the original and averaged sets contained the same number of samples, the spread set contained 8X that amount. Additionally, we know that the “rock” and “scissors” motions both require part of your fist to be clenched, while the “paper” and “ok” motions require that same part of your fist to remain open and flat, which affects the readings on specific sensors. The combination of these factors, with a larger emphasis on the increase in observations, led to a final accuracy of 32.98% as shown in Table 1.

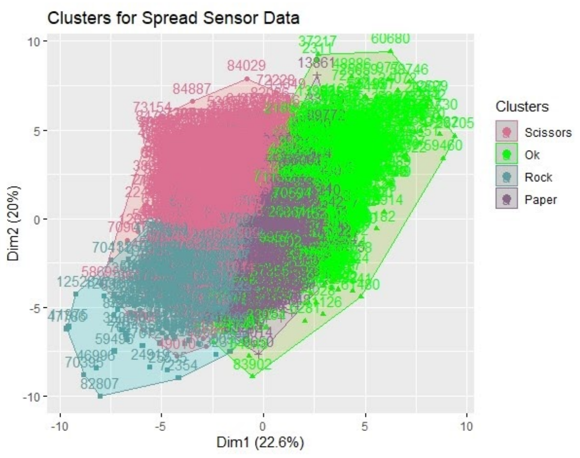

Figure 3: Plot of clusters obtained using k-means on the spread dataset with k=4. The cluster numbers shown represent the gesture type that is most likely for a particular grouping.

In comparing the results of clustering on the original, averaged, and spread data sets as summarized in Table 1, we see that the accuracies for all three were quite close. The original set had the lowest accuracy as a result of the high dimensionality, while the spread set had the highest accuracy, which we can assume is from having a reduction in the number of attributes and an increase in the number of samples. All three accuracies were greater than 25%, thus our hypothesis that clustering could predict each type better than that percentage was correct, although we would have liked to have seen even better results.

Table of Accuracies	
Data Set	**Accuracy**
Original	29.16%
Averaged	31.94%
Spread	32.98%

Table 1: The calculated accuracy for k-means fits on various data sets using k=4. The accuracy shown is the highest accuracy obtained when considering which types each cluster obtained is most likely to match.

Neural Network Implementation

To create our feed-forward neural network, our team used the neuralnet package from R. Sensory inputs used to train and test the neural network were also reduced, as upon further inspection our group realized only a few key sensors were instrumental. Our training data was also reduced from 84,052 to the first 5,000 data set entries, as we saw otherwise that the neural network was unable to properly converge.

During our team's random tree forest implementation, we discovered that sensor 7 appeared to be the most important predictor for gesture class type. Expanding upon this finding, the neural network was trained only on sensors 6, 7, and 8 rather than all eight sensor inputs. The structure of the feed-forward neural network is shown in Figure 1.

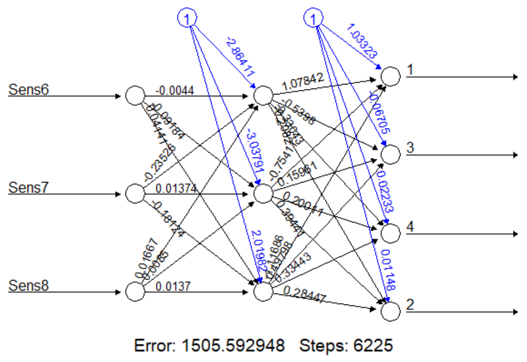

Figure 1: Neural Network obtained through three inputs, from sensors 6, 7, 8. One hidden layer was used consisting of three nodes, excluding the bias variable, which then outputted to 4 possible outputs. The four possible outputs corresponding to the four possible gesture types.

Our team decided on testing the feed-forward neural network against individual datasets consisting of one gesture class as well as the spread data set. The results on all sets are shown in Figure 2 and Figure 3, as well as Table 1.

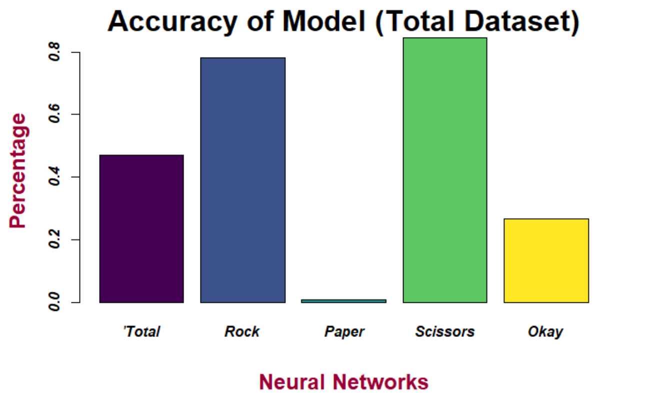

Figure 2: Accuracy of feed-forward neural network on individual datasets consisting of one gesture class as well as the Total data set.

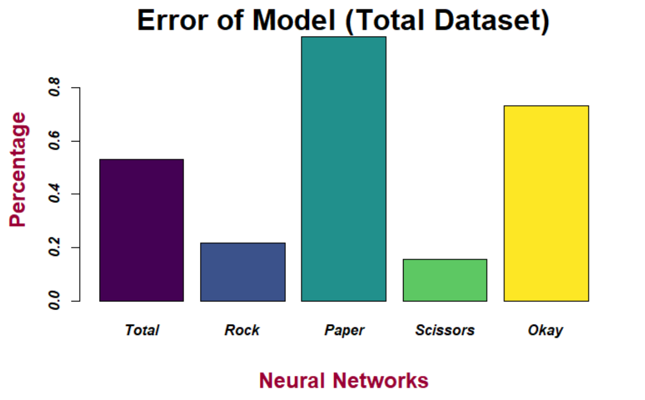

Figure 3: Error rate of feed-forward neural network on individual datasets consisting of one gesture class as well as the Total data set.

Data Set	Total	Average	Spread
Total	52.34%	50.03%	47.02%
Rock	84.29%	69.37%	84.29%
Scissors	60.51%	19.29%	60.51%
Paper	0.03%	0%	0.88%
Okay	29.27%	33.24%	25.68%

Table 1: Accuracy and Error rate of feed-forward neural network on individual datasets consisting of one gesture class as well as the Total data set (labeled 'Total').

The neural network achieved 52.34% accuracy on the totaled data set. To further analyze why we achieved this accuracy, the neural network was tested against individual datasets (the individual datasets being 'rock', 'paper', 'scissors', 'ok'). The individual datasets followed the same principle as the spread dataset where they were split into 10% testing and 90% training. When a prediction list was made, the highest prediction value in the list for a given data entry was considered the predicted gesture class. This predicted gesture class was predicted against the actual gesture class where the test set gesture class value was not removed. Each individual datasets' overall accuracy contributed to the 'Total' or 'totaled' dataset accuracy.

The paper dataset had a detrimentally negative impact on the totaled data set accuracy, as the paper dataset only predicted the correct gesture class about 27 times out of 2,942 total predictions. Further analysis of the prediction list for the paper dataset demonstrated that prediction of class value as being paper consistently had the second highest prediction value of 31.9%-33.79% while the highest prediction ranged from 35.64%-44.18%.

Investigation into how overall accuracy would be affected if the top two prediction values were randomly picked for the prediction value could result in a promising increase in overall accuracy for the neural network model.

Random Forest Implementation by Hand

Prior to our random forest implementation, we formed shuffled training and testing sets on the totaled, average, and spread datasets. We then introduced the use of the Rpart and C5.0 decision trees into our random forest implementation using a function named 'getfit'. This function returns a Rpart or C5.0 decision tree based on the formula it takes in. This formula can be randomized to mimic the randomization selection criteria of a random forest.

Since the random forest algorithm creates thousands of trees that select a random criteria of features, we decided to randomize the selection of features using a random decimal generator from 0 to 1. We used that decimal value multiplied by the total number of features to select the number of features selected for each tree. We then treated this number as a T value. To get decision trees with at least three features, we skewed this number by increasing the Tmin value from 0 to .375.

The procedure described above was truncated into a 'getformula' function that would source back information to the getfit function when needed. We also implemented an option to select a static amount of features as a traditional Random Forest would. We decided this optimal value by testing the performance of four, five, and six features on our datasets. We decided to test at the most six features so dimensionality would create a significant enough difference.

After the formulas were processed, we iteratively used the 'getfit' and 'getformula' functions for n trees created. Ideally, we would create thousands of decision trees to base our decision on.

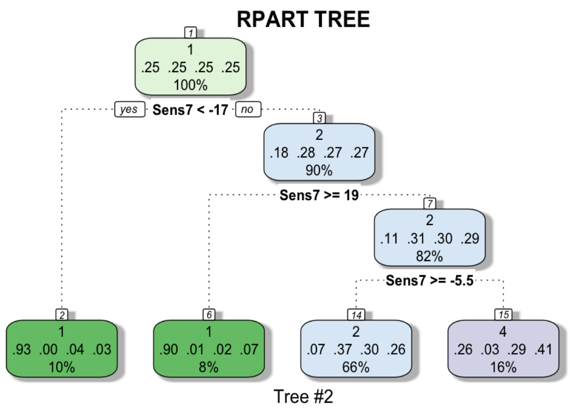

Figure 1: Decision Tree #2 in the rpart decision tree list

Once the decision trees are stored in a list using the 'get_n_trees' function, 'bind predictions' then iterates over the table and creates predictions of testing data using all three data sets. These predictions are then stored in individual lists where we can iteratively form Mean tables.

Methods:

- getformula(n.feat,features,targetft,random)
- getfit(new_formula,data,type)
- get_n_trees(ntrees,train,features,targetft,type , random)
- bind_predictions(num_predict,test,trees)
- ge_mean(prediction_list)

Note that the 'getfit' and 'get_n_trees' functions take in type values. This integer value (0:1) indicates whether to treat the input as a Rpart prediction (0) or as a C5.0 prediction (1). 'getformula' as well as the 'get_n_trees' function, both take in a variable named random. "random" takes in an integer value (0:1) that indicates whether to treat this as a classic random forest implementation or a random feature size selecting model.

Method hierarchy:

+ **get_n_trees ← getfit() ← getformula()**
+ **bind_predictions ← get_n_trees()**
+ **get_mean ← bind_predictions()**

This dynamic prediction table below, uses an optimal reduce value of .45. We found this optimal value by testing multiple reduce values that skewed the number of features selected. The optimal reduce value returned the highest accuracies in the search of relations between features analyzed and accuracies.

DYNAMIC TABLE OF ACCURACIES PREDICTIONS: 400		
Data Set	**Rpart Accuracy**	**C5.0 Accuracy**
Totaled	26%	26%
Averaged	51%	44%
Spread	45%	49%

Table 1: Dynamic prediction accuracies for the datasets implemented

The predictions found in tables 3, 4, and 5 are formed using a mean table of all predictions in a set of trees, formed using the static method of the algorithm. The set of trees which was analyzed is composed of N predictions. N predictions of 40, 200, and 400 are tested primarily to reduce CPU usage.

These predictions were also formed using a data set length of 2,000 samples. This ensured that the sample amount used for each analysis would be consistent but also assists with CPU usage of experiment computers.

STATIC TABLE OF ACCURACIES PREDICTIONS: 40		
Data Set	**Rpart Accuracy**	**C5.0 Accuracy**
Totaled	26%	25%
Averaged	43%	43%
Spread	51%	59%

Table 2: Static prediction accuracies with N=40 for three datasets

STATIC TABLE OF ACCURACIES PREDICTIONS: 200		
Data Set	**Rpart Accuracy**	**C5.0 Accuracy**
Totaled	26%	25%
Averaged	51%	49%
Spread	51%	52%

Table 3: Static prediction accuracies with N=200 for three datasets

STATIC TABLE OF ACCURACIES PREDICTIONS: 400		
Data Set	**Rpart Accuracy**	**C5.0 Accuracy**
Totaled	26%	26%
Averaged	52%	48%
Spread	53%	52%

Table 4: Static prediction accuracies with N=400 for three datasets

Our predictions show that we have similar overall accuracy from both decision trees. We get an overall accuracy of 42% for both decision tree types. From this individual experiment, we could hypothesize that the totaled data does not create much of a correlation between sensors. Yet we have seen correlation in the previous experiments.

Our best individual accuracy performance came from the spread dataset, but the averaged dataset also achieved an accuracy similar to the spread set. This gives the impression that the averaged and spread datasets create correlations between sensor data. We were able to achieve a 59% accuracy using C5.0 decision trees and the spread dataset. We also achieved a 53% accuracy with a Rpart model.

Given that run time for creating Rpart trees is at a minimum four times higher than C5.0 decision trees, C5.0 decision trees would be the ideal candidate for use because they can greatly improve test times and return improved results.

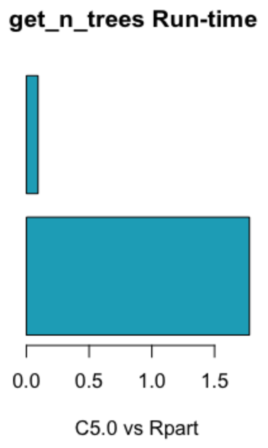

Figure 2: get_n_trees run-time comparison of C5.0 and RPart trees

Conclusions and Future Work

Conclusions

During the experimental evaluation of the EMG data set, our group was successfully able to conclude the hypothesis proposed in the introduction. Specifically, all three of the models we implemented had an accuracy of over 25%. This means our models were able to predict the gesture group better than randomly guessing. The best accuracy obtained from each model is as follows:

- Clustering - 32.98%
- Neural Net - 50.34%
- Random Forest - 59%

The random forest classifier accuracy was higher in the specific case of the spread dataset implementation of the C5.0 decision tree; overall accuracy of the random forest classifier implementation sits at 43.79%. In directly comparing the classifiers, the Random Forest Algorithm outperformed the Feed-Forward Neural Network by 8.66% and the clustering classifier by 26.02% in overall accuracy.

In comparing the results between clustering and neural networks, we know K-means clustering is one method for reducing data to groups of similar objects through distance association, meaning there are rules that must be followed to find patterns. Neural networks differ in that they are unlike conventional statistical or rule-based systems. They are not programmed to perform particular tasks according to strict rules but instead isolate patterns in the data by training on historical data using a learning algorithm. Thus, they allow for larger encompassing pattern recognition to form during the learning phase as compared to the K-means clustering. In other words, Neural Networks "create" rules for pattern recognition while K-means clustering follows predefined rules for pattern recognition.

In theory, rule creation would allow for more diverse selection of prediction strategies which ultimately would result in a more accurate strategy than strict rule-based strategies could provide. Strict rule-based strategies follow a “one size fits all” approach whereas rule creation allows rules to be created for the specific data set the neural network is learning. This is why the neural network performed much better than clustering at predicting the gesture classification.

In comparing the neural networks to random forests, we know random forests, like the neural networks, are not limited by a strict set of rules but rather attempts to find patterns within the data. Therefore it would make sense that Random Forest would have a similar accuracy to a Feed-Forward Neural Network (1.66% difference in overall accuracy), as both random forest and neural networks are designed with the fluidity to create their own set of rules.

Future Work

Out of our results from all three models, the best accuracies came when using the spread data set. This means that the models did not take into account the aspect of time. This is due to how the readings were taken, as the data was taken while holding any given gesture for the whole time rather than going through the motions of making the gesture. Thus, at least from the observations we got from the datasets we tested on, time was not crucial for analysis but could still be looked at more closely in the future.

Our team had started to create a rolling average for each sample in the original dataset to still include the aspect of time for further analysis, but because of the time constraint on the project we were unable to get it fully implemented. If a rolling average dataset were used to train each of the three classifiers, there could still be the possibility of improved accuracy as the data points would be more balanced through averaging with this method.

Finally, for the random forest implementation, we would like to attempt experiments with higher prediction values and explore with further tuning of the model. Seeing as though it is common to run thousands of trees, we would also like to attempt creating up to 8,000 trees and assessing the accuracy differences.

References

Atzori, Manfredo, et al. “Deep Learning with Convolutional Neural Networks Applied to Electromyography Data: A Resource for the Classification of Movements for Prosthetic Hands.” *Frontiers in Neurorobotics*, University of Applied Sciences Western Switzerland, 7 Sept. 2016.

Brieman, Leo, “Random Forests”, statistics Department University of California Berkeley, Berkeley Statistics, Jan 2001.

Côté-Allard, Ulysse, et al. “Deep Learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning.” *Arxiv.org*, Natural Sciences and Engineering Research Council of Canada, 26 Jan. 2019.

Karuna, M., et al. “Classification of Wrist Movements through EMG Signals with Fuzzy Logic Algorithm.” I*EEE Xplore*, IEEE, 21 June 2018.

Zhang, Zhen, et al. “Real-Time Surface EMG Pattern Recognition for Hand Gestures Based on an Artificial Neural Network.” *National Center for Biotechnology Information*, MDPI, 18 July 2019.