Multi-Class Classification for Basic Hand Movements

Kening Zhang, Zijing Gu, Wenlong Zhao, Yuesheng Luo

December 2017

1 Introduction

The technique of electromyography (EMG) is used to record and analyze skeletal muscle activities. The activated muscle cells produce electric field potential that can be detected by an electromyograph, and these signals are evaluated to identify biomedical abnormalities, investigate biomechanics of movements, and many else. As EMG system being widely applied to clinical and biomedical areas for the study of moving muscles, the data of EMG based hand movements are largely collected, and the recognition of these hand movements becomes a relevant problem. Once a signal is detected, can we obtain its corresponding activity? Ideally, a good hand movement classifier can be used in prosthesis to recognize user intent and to move the prosthetic arm accordingly. This will not only benefit hand amputees in their life but also contribute to robotic exoskeleton technologies.

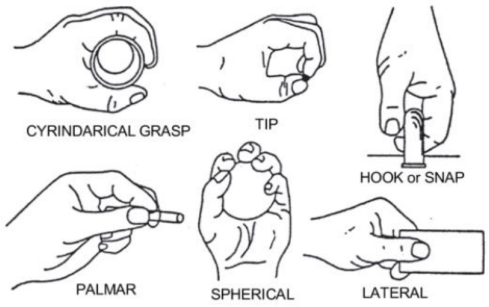

Figure 1: Six hand gestures

The goal of this project is to classify six daily life hand grasps (illustrated in Figure 1) whose surface electromyographic signals are recorded using Delsys' EMG System. After applying several supervised learning algorithms, we compared the testing results to see which algorithm or combination gave the highest

test rate of accuracy on our data set. The classifiers we used include feedforward neural network, Adaptive Boosting, Random Forest, Linear Discriminant Analysis (LDA), Principal Component Analysis (PCA), etc. We also implemented Empirical Mode Decomposition (EMD) and eight feature extraction functions to re-project the data features and improve the classification performance. As a result, we managed to raise our test accuracy rate from only 23% to over 94%.

2 Related Work

Our topic is inspired by another project called "Human Activity Recognition with Smartphones [3]" done by previous students. Basically, they used the MATLAB Neural Network Toolbox to classify biological action data such as walking and sitting performed by human beings while having a smartphone attached to their waist. The data were collected through the internal accelerometer and gyroscope of the smart phone. Their test accuracy rates reached over 97% after first few trials. Following this ideology, we felt we could implement a more fleshed out version of their project by fitting the data into other machine learning models and analyzing the differences. We also chose a more complicated biological dataset recorded from muscle EMG signals which was much harder to classify.

In [8], the researchers aimed to increase the ability for pattern recognition on the conventional features of a set of EMG based hand movements data. What makes this study unique from other similar researches is its use of only two electrodes and the implementation of EMD to acquire extra features. EMD is especially useful for analyzing nonlinear and non-stationary signals. It decomposes the signal to a number of IMFs, each representing a simple oscillatory function with symmetric envelopes respected to zero. With a simple linear classifier, the result shows that the IMF transformed data have a positive effect on the accuracy rate. We used this paper as a reference to extract feature to improve our classification performance.

[2] uses feed-forward neural network and back propagation algorithm to classify hand gestures, and an 89% correctness was achieved on a typical test set. The researchers used computer vision based techniques to acquire hand gesture data. They recorded ten hand gestures videos and converted the videos into binary images. The images were used to create a 3D matrix of $30 \times 30 \times 5N$, where 30×30 was the hand image size and the N was the number of examples. The generated matrix of data was inputted into Neural Networks with 750 neurons in the input layer, 7 neurons in the single hidden layer and 5 neurons for the output layers. The method was then tested with five users and ten gestures. The system could recognize either the number of fingers or the pointed direction.

The focus of [4] is to use a constructive feedforward one-hidden-layer neural network to classify different facial expression inputs. Each subject was recorded with 5 expressions: neutral, smile, anger, sadness and surprise. The test accuracy rate ranged from 93.75% to 100%. With the pruning method decreasing

the input-side weights by 30%, the performance of the network was improved. The reason for choosing a one-hidden-layer was that it performs considerably well for image recognition. "Deeper" neural networks need more input-side connections for new hidden units which may cause irrelevant connections to the prediction. Compared with a back-propagation-based recognition network, the constructive technique generates significantly fewer hidden units with a better performance.

[1] uses the template-based random forest to extract features from skeleton based human gestures. The dataset of Italian Gestures by ChaLearn 2014 which contains 13858 gesture from 20 Italian cultural signs recorded by Microsoft Kinect sensor was used. Both spatial and temporal information gained from the skeleton features of gesture was used to train the model. The researchers used score fusion to compare different method of classification to yield the best overall performance. Rather than classifying the features into clusters and construct interactions among each cluster, gesture templates focus on learning the static sequential features. The method goes through 5 steps: normalize the joint coordinates, represent gestures with skeleton features, construct gesture templates for spatial learning methods, used random Decision Forests to train classification, and use score fusion to combine the prediction of different models. It resulted in a Jaccard Score of 78.75% and received a 7th place from the 17 submitters.

[5] states a new method for face image recognition by using linear regression to find the pattern from single object class on a linear space. The paper mentioned that when the face picture had too many dimensions vectors, the attempt to recognize patterns became hard. Therefore feature extraction methods were extremely needed. The paper proposed dimension reduction methods like PCA and LDA but decided to choose the simple but efficient LRC. The researchers fused the concept of wavelet decomposition and discriminant analysis to form a complicated feature extraction stage. Modular LRC approach was used to deal with severe contiguous occlusion. Finally, it showed that the downsampled images combining with LRC could achieve better results than the traditional approaches.

3 Dataset and Preparation

We chose the second database of sEMG for Basic Hand Movements Data Set [7] selected from UCI machine learning repository [6]. Through a 2-channel EMG system by Delsys Bagnoli Handheld EMG Systems, the data were collected from a 22-year-old healthy male performing the six hand movements illustrated in Figure 1 for 100 times each and for three consecutive days. The gestures consisted of freely and repeatedly grasping of different items with the speed and force up to the subject. The muscle activation signals were collected by two channels at a rate of 500 Hz for 5 seconds, giving each vector 5000 features. The data features were stored in three mat files (one file per day) with 100×6 data vectors per file.

For the data organization process, we used two MATLAB concatenate functions, horzcat and vertcat, to combine the data in three files and two channels together into one matrix called "feature" which had 1800 rows (size of the dataset) and 5000 columns (number of data features). Vectors were kept in an order such that those represented the same movement stayed together. For example, all 300 vectors of gesture 1 occupied the first 300 rows of the "feature" matrix, gesture 2 occupied the next 300 rows, and so on. For training and testing purposes, we created two 1800×1 "label"s for the data to indicate which gestures they represented. The first label consisted of numbers from 0 to 5 with each number corresponded to one gesture. The first 300 entries of the label were 0s which indicated gesture 1, the next 300 entries were 1s which labeled gesture 2, and so on. The other label consisted of one-hot line coded labels. 100000 represented gesture 1, 010000 gesture 2, and so on. The two labels were selected to fit into different learning algorithms.

4 Methods

Neural Network was our first attempt at our project. We then tried other supervised learning algorithms including Adaptive Boosting and Random Forest. We also learned from [8] to transform the data with EMD and feature extraction functions. After that, we trained and tested the transformed data with a linear classifier as mentioned in [8] and [5], as well as with Neural Network and Random Forest. We made our last guess of implementing PCA to the extracted features in addition to Random Forest and achieved our best results.

4.1 Neural Network

We set 80% of the dataset for training and the rest 20% for testing. Among the 80% training data, we randomly selected three fourths to be the training sets and the remaining to be the validation set. Cross-validation would be performed, that is, 60% of the entire dataset was used for pure training, 20% for validation, and 20% for testing. Following the guidance of [3], we used the toolbox GUI and trained our dataset by adjusting the hidden layers. To be more ambitious, we overcame the limitations of the GUI by changing the default parameters that could not be adjusted on GUI. To accomplish this, we took the running script generated by GUI, named it "runNN", and manually modified its content such as the pre and post-processing functions and the performance function. Then we wrote a method called "loopNN" that took the dataset and its label as the inputs, called runNN repeatedly while looping through all combinations of the variables, and output each average accuracy rate in a matrix for direct comparison and analysis. Since the original script did not output the training, validation, or test rates of accuracy individually, we modified runNN to capture the three groups of data using masks and recalculate all the three accuracy rates for the output matrix. With the scaled conjugate gradient algorithm which had low requirements for the memory storage, we changed the number of neurons

in the hidden layer. Each setting was run for five times and an average was calculated.

4.2 Adaptive Boosting

During the experiment, we were informed that Decision Tree performs better with relatively small datasets like ours than Neural Network. Thus we switched our methods to Decision Tree along with two other stronger learners. One of them was Adaptive Boosting (AdaBoost). AdaBoost was relatively more resistant to overfitting problems. We learned that when AdaBoost was used with Decision Tree learning, the tree growing algorithm is fed with the information on relatively "hard" training samples such that later trees tend to focus on examples that are harder to classify.

In this case, we imported the train_test_split function and set the test size to 0.1. We used the Python libraries to perform AdaBoost. The Decision Tree classifier is one of the parameters of AdaBoost classifier. We tried max depth of 4 for the Decision Tree classifier, set 2000 trees, and a learning rate of 0.1. Then we calculated the test errors as the trees grew and put into an array. We also calculated the training error.

4.3 Random Forest

The second strong learning algorithm we used with Decision Tree is Random Forest. We figured that this method works better on dataset with relatively small sizes. Random Forest corrects for Decision Trees' habit of overfitting to their training set, which was what we wanted.

After importing the relevant Python libraries, we defined a Random Forest classifier, setting 5000 trees, max depth of 20, and min sample split of 5. Then we calculated our accuracy rates and plotted the results.

4.4 Empirical Mode Decomposition and Feature Extraction

With such a small-sized dataset of huge dimension and weak signals, we wondered if we could transform the 5000 features of each data trial into more meaningful features with a lower dimension. We tried PCA but was unsuccessful in improving the accuracy. [8] led us to the method of EMD. EMD acts as a non-linear filter that can decompose signals into Intrinsic Mode Functions (IMF) [8]. IMF represents a simple oscillatory function that helps us extract oscillatory information of the movement of the gestures. We then took this information and plugged into the feature extraction functions to obtain the new feature for classification.

The first step was to use MATLAB EMD Toolbox to perform the transformation. We plugged the 2500 features from Channel 1 of our EMG dataset into the emd function, and a list of new data vectors transformed by IMFs was

returned for each original data vector. Note that by default, the emd function will keep decomposing signals to IMFs and will stop when the residual becomes a monotonous function [8]. That is to say, the residual function of each data vector will be returned at the end of the IMF list. We plotted the first original data vector in our EMG dataset, its three IMF transformed values and the residual. The resulting Figure 2 gives a clear graphic representation of the differences between EMG and IMF functions. We can see that IMF has fewer fluctuations comparing to the EMG function, and the IMF functions were decreasing in value.

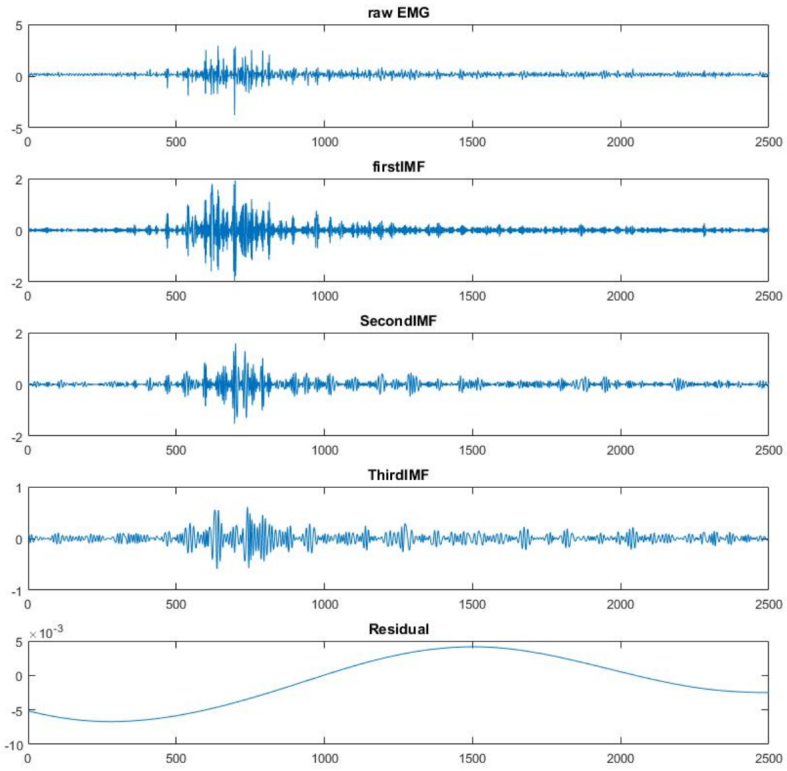

Figure 2: Data projected by EMD and IMF

The second step was to implement 8 feature extraction functions [8] on each EMG and IMF transformed data to obtain the new features for each gesture. These 8 functions are mostly simple calculations. One of them is the Integrated EMG which calculates the average value of the absolute values of the raw EMG data. The Waveform Length function is to calculate the absolute difference between two adjacent data point and sum up all the differences. The intuition behind these 8 function is that, by observing different extracted features

of different gestures, machine learning algorithms can easily distinguish them. We expected that the newly extracted features could improve the classification accuracy.

Figure 3 is the plot of 8 feature extraction methods applied individually to the entire dataset. Through each of the functions, the 2500 original features of each data vector were projected into one single point as a new feature. The new features were projected in order: the first 300 feature points belonged to gesture 1, the following 300 belonged to gesture 2, and so on. We can use Figure 3 to study the differences between the gestures. For example, we see that the first 300 points (gesture 1) in the third function (the Variance function) has generally higher values than other points. This means that when dealing with a relatively high value returned by the Variance function, we expect it to be gesture 1.

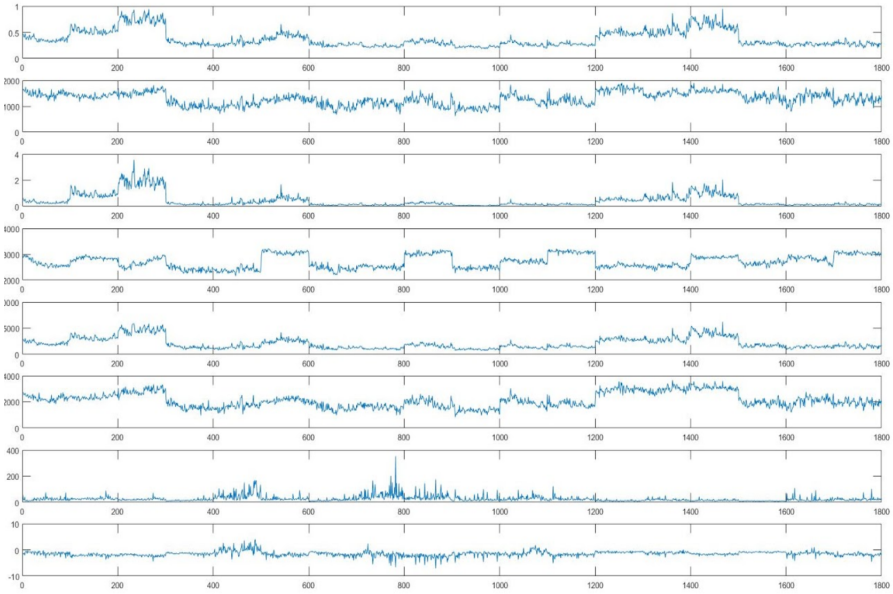

Figure 3: Eight feature extraction functions applied to all the 1800 data

So after the EMD process, each data vector had a total of five forms: its original EMG form, three IMF transformed forms and a residual form. Then we plugged the five forms into the eight feature extraction functions to obtain a total of 40 new features for each data vector. Figure 4 is a visualization of how the 40 features of the first data vector of each of the six gestures look like. They are somewhat in a similar shape but different in detail.

Finally, we put all the features of all the data into a 1800 by 40 matrix for the new round of classification. We used a simple linear classifier as introduced in [8][5]. We also put the new dataset back to the previous methods we used: Feedforward neural network, Adaptive boosting, and Random forest.

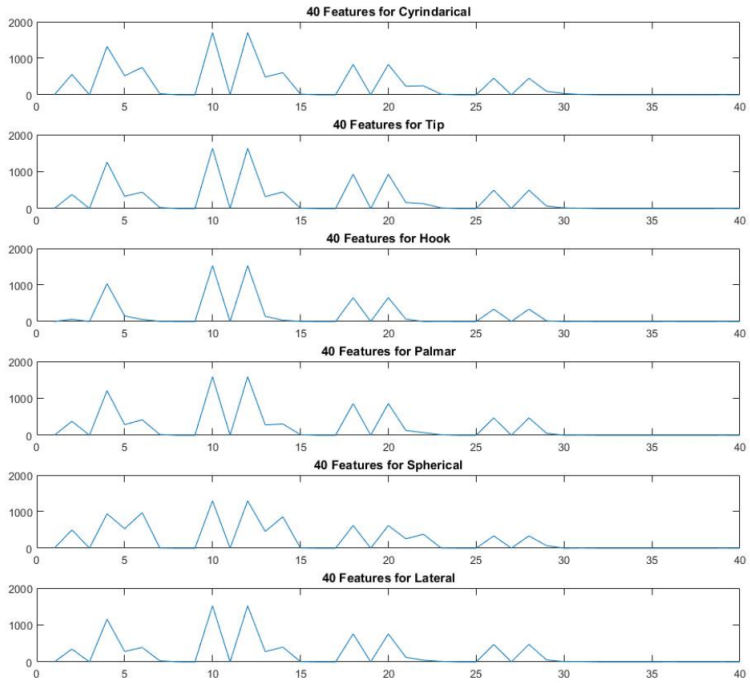

Figure 4: Each data vector now has only 40 features

5 Results

The dataset was much better classified with the EMD and feature extraction process than without.

5.1 Neural Network

We spent a large amount of time tuning the parameters, changing the training algorithm as well as the performance function. We figured that setting the training function to "trainscg" and the performance function to "mse" gave relatively better results. Then we set the number of hidden layer neurons to 10, 40, 160 and 640 respectively. The test accuracy rate turned out to be only about 20%. We, therefore, tried larger numbers of neurons such as 2560 in the hope that the function would fit better. The average result grew a little to around 23%, which was still not satisfying. We also tried the python library of Neural Network, and the results improved a little to 30%. We found out that our highest training accuracy actually got to 99.8%, but the testing accuracy was as low as 20-30%. We wondered if the model over-fitted the dataset.

5.2 Adaptive Boosting

We tested our data using the staged_predict function of the AdaBoost classifier. Figure 5 is the plot of our resulting test and training rate of error. As we can see, the error rate for testing was dropping as the tree grew but very slowly, and the rate stabilized at around 40-50%. That is to say, we had an improved test accuracy rate of around 50-60%.

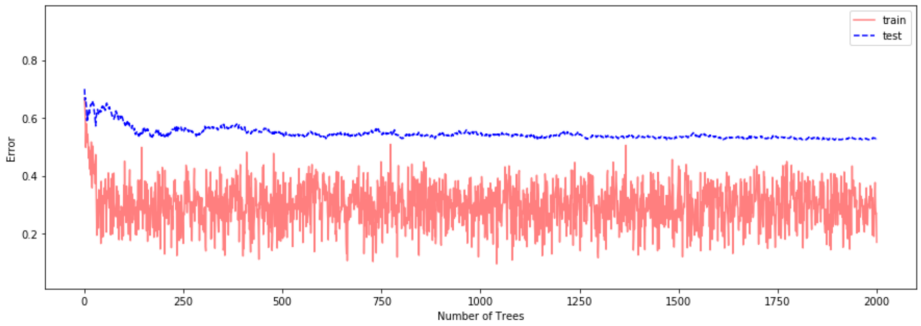

Figure 5: Error rate for AdaBoost

Figure 6 is our corresponding confusion table. We found out that this algorithm had a hard time classifying certain pairs of the gestures. For example, from the first row, gesture 1 was confused with gesture 6, and from the last row, gesture 6 was confused with gesture 4, etc. We wondered if we only train and test only these confused pairs, the result would improve. Also, parameter selection could also be an important factor.

```
[40,  3,  1,  2,  0, 43]
[ 0, 55,  5, 31,  0,  4]
[ 0,  1, 30, 62,  0, 17]
[ 0,  0, 20, 67,  0,  6]
[ 4,  0,  0,  2, 51, 52]
[ 0,  0, 11, 48,  0, 39]
```

Figure 6: Confusion matrix for AdaBoost

5.3 Random Forest

Our training accuracy reached 100%, but most excitingly, with the predict function of the Random Forest classifier, our test accuracy rate reached 75%! . Figure 7 is the resulting confusion matrix that looks pretty positive. Compared with the confusion matrix of Adaboost in Figure 6, we got predominantly larger percentage of numbers on the diagonal, which indicated higher test accuracy.

```
[26,  0,  0,  0,  3,  0]
[ 1, 26,  1,  3,  0,  0]
[ 0,  1, 24,  5,  0,  4]
[ 0,  3,  6, 16,  0,  2]
[ 0,  0,  0,  0, 28,  0]
[ 1,  0,  9,  5,  3, 13]
```

Figure 7: Confusion matrix for Random forest

5.4 Empirical Mode Decomposition and Feature Extraction

5.4.1 Linear Discriminant Classifier

The accuracy was around 65%. The classifier we constructed seemed not suitable enough.

5.4.2 Neural Networks

With "trainscg" as the training function, "mse" as the performance function, and 10 hidden layer neurons, the test accuracy reached 80%. As we increased the number of neurons to 80, the rate reached 88%.

5.4.3 Random Forest

In addition to the 40 features extracted through EMD, we used the 2500 features in Channel 2 and extracted 40 more features. Now each data vector had a total of 80 features. The test accuracy reached 91.9%, which was a great result compared to our previous rates. Then we tried PCA to reduce the dimension of data set, and the average accuracy rate reached over 94%.

6 Discussion

All the approximate average test rates of accuracy we obtained are summarized in Table 1.

	Before EMD	After EMD
Neural Network	25%	85%
Adaptive boosting	55%	-
Linear Discriminant Analysis	-	65%
Random Forest	75%	91%
Random Forest + PCA	43%	94%

Table 1: Average test rates of accuracy

6.1 Analysis

After using EMD, the accuracy was improved significantly. We tried to deduce the reasons that could possibly explain the differences before and after EMD. For Neural Network, we believed that our original dataset had too few data and too many features. Neural Network usually works the best on big data. But if we extracted features and reduced the dimensions, the results would improve. Similarly, Random Forest performs better with small datasets but may over-fit the datasets that are particularly noisy. But after extracting only 80 features out of 5000, the number of features was a lot lower than the number of data. We believed that the not only EMD but also the ratio between the dataset size and dataset dimension contributed to the huge improvement in the accuracy in both Neural Network and Random Forest.

6.2 Potential Improvements and Extensions

In Section 5.2, we mentioned that if the pairs of classes are selected based on the confusion matrix and are trained separately, we wonder if AdaBoost will do a better job distinguishing within each pair. Also, implementing AdaBoost on the new dataset we obtained after EMD and feature extraction is worth trying. We may also use cross-validation for a more thorough training while implementing these supervised learning algorithms.

For now, we used all eight popular features and the classification reached 90%, which was pretty satisfactory. However, some features may not be important and would, in fact, be a distraction during the classification process. We may try selecting a couple "essential" features and see if it improves the accuracy. We may also try new feature functions that would potentially be helpful to distinguish those easily confused gesture signals. We can as well try Band-Pass Filter (BPF) in different frequency bands (e.g. wavelets or just Fast Fourier Transform (FFT) in different sub-bands).

Our project of gesture recognition may be extended to more EMG functions or similar datasets.

7 Conclusion

EMG based hand movement recognition is a useful technique that can be widely applied in our life. It can help robotic exoskeleton hand move more autonomously. For a hand amputee, the technique is also helpful since it is easier to use an EMG glove than wearing an EEG helmet. In our project, we aimed to develop the best machine learning model for the classification of the sEMG Hand Gesture dataset. The dataset contains the signals of six daily hand gestures in three files, and we re-organized our data through MATLAB concatenate commands. We had hands-on experiments with MATLAB Neural Network Toolbox and the EMD Toolbox. They are well-developed for machine learning purposes. We also explored the actual algorithm of Neural Network by modifying its components for higher accuracy rates. We experienced other

supervised learning methods such as Python AdaBoost classifier and Random Forest classifier to solve the problem of overfitting occurred while using Neural Network. We had a general idea how each algorithm works together with its parameters and how well it fits into our data set. Under the guidance of [8], we used EMD to decompose our data signal into a list of IMFs whose maxima are all positive and its minima are all negative. We then extracted eight popular features through eight simple calculations on the original features to reduce the dimension of our dataset. This step aimed to further differentiate each gesture class by giving it less but more unique patterns.

Before EMD, the test accuracy rate for Neural Network ranged from below 20% to around 30%. Adaptive Boosting gave an improved rate of 50-60%. Random Forest later raised the accuracy to over 75%. After EMD and feature extraction, Neural Network improved the accuracy to over 80%, and Random Forest to over 90%. With an additional PCA, Random Forest obtained the test accuracy of 94%, which was the best result throughout our project. We thus inferred that, for small datasets with large data features, Random Forest with Decision Trees has a generally better performance than Neural Network, and the EMD and feature extraction become the crucial step of the classification process.

8 Acknowledgement

We would like to give our special thanks to professor Virginia de Sa, teaching assistant Sai Gullapally, and Ph.D. candidate Alex Li for advising and supporting us on this project.

References

[1] N. C. Camgoz, A. A. Kindiroglu, and L. Akarun. *Gesture Recognition using Template Based Random Forest Classifiers*. Istanbul.

[2] S. E. Fahlman. *An Empirical Study of Learning Speed in Back-Propagation Networks*. Pittsburgh, PA, Tech. Rep., 1988. CMU-CS-88-162.

[3] R. Lieu and W. Cui. *Human Activity Recognition with Smartphones*. https://docs.google.com/document/d/1UefI9Jph98LxvHI5vf7fTx8TeOmV7wYr99rmFTABLP0/.

[4] L. Ma and K. Khorasani. Facial Expression Recognition Using Constructive Feedforward Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics*, 34(3), 2004.

[5] I. Naseem, R. Togneri, IEEE Senior Member, and M. Bennamoun. Linear Regression for Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(11), 2010.

[6] UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/sEMG+for+Basic+Hand+movements#.

[7] C Sapsanis. *Recognition of basic hand movements using electromyography*, 2013.

[8] C. Sapsanis, G. Georgoulas, A. Tzes, and D. Lymberopoulos. Improving EMG Based Classification of Basic Hand Movements Using EMD. *35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society 13 (EMBC 13)*, pages 5754 – 5757, 2013. doi:10.1109/embc.2013.6610858.