Detecting ASL gestures using KNN with Select K Best

Selene Moraga, Alexander Taffe, Anthony Gonzalez
CSE 450 Design and Analysis of Algorithms

Introduction
American Sign Language(ASL) is a widely used method of communication. The World Health Organization (WHO) has listed under key facts about deafness and hearing loss that, “ 360 million people worldwide have disabling hearing loss, and 32 million of these are children”. ASL is a communication option for someone who was born deaf or with hearing loss. Some parents decide to learn ASL with their children at the same time. But with the growing popularity of the internet of things (IOT), people who use mainly or only ASL for communication should be able to extend the audience of who they can talk to in a convenient way. One example of this is a mobile application that allows the conversion of sign language to a text message. This could be a possible replacement for dictation. Another mobile application that could add convenience to the lives of ASL only users is a sign language to voice application that speaks the letters that are signed. We offer one solution to this problem by using data from two MYO gesture control armbands.

We propose a method to recognize an ASL finger spelled letter using a combination of Select K Best and K Nearest Neighbors (KNN) classification on preprocessed data from the MYO armbands. This approach is designed to be able to be used in a mobile computing setting. In this project we started with preprocessed data, but when implementing this algorithm in an application for example, data would be collected from the armbands during a listening period. One example listening period like this occurs when using the Siri feature on an iphone. When the user would like to make a request they say the phrase “Hey Siri.”; then voice their request. The same thing would occur when using our algorithm in a mobile application. The user would use the American Fingerspelled Alphabet(AFA)[2] to make the hand sign for the desired letter during this period. Data on the hand sign would be collected during the listening period and preprocessed after this period has ended. This preprocessed data is then passed to Select K Best for feature selection and KNN for classification and letter output.

Usage however requires one training session, the first time the user uses the armbands with the application. This training is needed mainly for sensor data variation between users and training KNN. This will be discussed in detail in later sections. A way to mitigate this, could be to store this training data in device setting. This allows all applications to have access to this data. This can also be compared to the thumb print security feature on the iPhone 6 and later. Instead of having to train possibly multiple applications, the user could provide one training instance for each letter as a part of an optional setup sequence. This training data is then stored and is globally accessible to any application on the device.

After training, recognition can be performed within our time constraints with a peak accuracy of 81.5%. This is discussed in detail in the problem results sections. Testing was conducted on pre collected data in the format of a comma separated value (CSV) file for three different people.

Problem Description

Our intention is to be able to classify the AFA by using the MYO wristbands. The wristband will give a time-series dataset that contains readings for 8 EMG sensors, 3 Accelerometer sensors, 3 Gyroscope sensors and 4 Orientation sensors. Our algorithm will compare the readings between training and testing sets in order to give us the highest possible accuracy to classify each gesture to its corresponding letter. The training and testing datasets will be further explained in the implementation section. The algorithm must also be efficient in order to classify gestures in real time, to keep up with the natural flow of conversations.

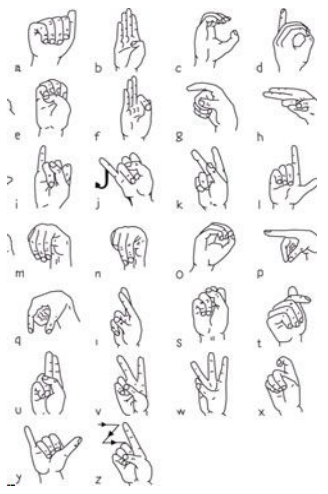

Figure 1

Related Work

Other approaches to this problem include the use of data gloves, cameras and also a MYO device. This project itself has branched from the other work that has used a MYO device for detection of AFA alphabet hand signs [2][3]. As mentioned before we approached this by looking at other methods that use supervised learning for gesture detection.

One of these methods used a decision tree with a data glove. [4] In this approach a data glove with 18 sensors and a Pohelmus 3-D tracker was used, along with one receiver on each wrist of the data glove. The alphabet included up to sixty-five unique chinese sign language gestures. A decision tree was chosen as the means to classify the signs, and compared to an Artificial Neural Network(ANN) and Hidden Markov Model(HMM). A decision tree was chosen over ANN because of the “tedious and time consuming training process” [4]. This is also the part of the reason we chose KNN over ANN. The nature of our problem requires that the algorithm work for multiple users and allow mobility of the user. We believe that ANN would not fit this criteria. HMM was not used because of difficulty with recognition of hand signs with little motion. This was analyzed in the previously mentioned work as well. An example of why this could prove to be difficult with the ASL as well, is shown in the Figure 1. The difference between the letters “m” and “n” is much smaller than other letters. To have modeled this distinctively would have proven difficult for HMM. The decision tree approach yield

recognition accuracy of 92.3% when trained with noise.
Other approaches include the use of Dynamic Feature Selection and Voting(DyFAV), and Dynamic Time Warping and ranking(SCEPTRE). These methods both used the MYO device used in this project, and reported accuracy rates of 95.36% and 97.72% respectively.[2][3]

KNN was compared to a decision tree and chosen to be the most appropriate algorithm to use. Based on prior work, the build time for a decision tree can be as high as $O(A^2N)$ for a tree of height A and N items. [9] KNN using the brute force approach has a worst case running time of O(nk + nd) where d is the number of dimensions for each point, n is the number of samples and k is the number of points that must match for recognition. This is one reason why we chose KNN.

Our Approach
There is three significant components contributing to the algorithm's time complexity: preprocessing, feature selection, and identification. Preprocessing consists of taking the raw data from the MYO armbands and converting it into various formats such as the maximum, minimum and average of each sensor. The preprocessed data is formatted in a CSV file which simplifies the scope of our project. This allowed us to focus on the algorithm to select the best features for our data in order to classify them as the correct hand sign. The data is then passed to Select K Best for feature selection.

Feature selection involves systematically reducing the number of features in hopes of increasing accuracy and reducing computation time. This reduces the dimensionality of the data before passing it to KNN for training. This set of features is also used during recognition. It was decided that feature selection would be necessary due to the fact that KNN suffers from the so called "Curse of Dimensionality". This is a problem that occurs when attempting to determine "structure in data embedded in highly dimensional space". [8] In the case of KNN this affects the process of trying to find the nearest neighbor of the point that is being tested, using euclidean distance. We anticipated this to be a problem based on the structure of our data. Each of our data points for training and testing have over five hundred features. This results in a sparse dataset, and lead us to decide to use some type dimensionality reduction, and for that we chose Select K Best. Select K Best was chosen as one of the most straightforward approaches. It allowed for removal of superfluous features and reduced the overall computation time of our algorithm.

Identification is the act of taking raw input data and determining which letter the data represents. Along with these three components is two phases: training and testing. The training phase consists of the preprocessing and feature selection components, and is used for preparing the system, while the testing phase consists of the preprocessing and identification components, and is used for active use of the system. As mentioned in related work KNN was chosen as the method of identification based on its running time for our data set and its overall simplicity when compared to other methods of identification. We used the brute force approach, although it still has the option of being implemented using a K-dimensional tree or a ball tree, so this provided flexibility.

Implementation

The results were gathered through the use of the Python programming language with help from a machine learning library known as Scikit Learn. Python was chosen for its simplicity and its ability to perform as an excellent platform for testing. After Python was decided upon as the implementation language, the next step involved research into systems that would support the task at hand. Scikit Learn was discovered and implemented as it allowed for more focus into the research and analysis of algorithms more so than implementation strategies. The results were also obtained via executing the Python code on a laptop with an AMD A10-4600M processor. This processor has a clock rate of approximately 2.3 GHz and four cores available for computing. This has potential to produce better performance over mobile processors which likely have lower clock rates and less cores. Normalized data was provided for several individuals, and much of the Python code involves formatting this data. The data consisted of approximately five readings for each letter in the alphabet. First, the csv file of normalized data for a single individual is stored into a matrix where it is then split into two groups: training data and testing data. The training data consisted of three readings for each letter, whereas the remaining readings for each letter were used for testing. The testing data was then run through the feature selection algorithm, Select K Best, to determine which features provided the best information. Afterwards, the kNN model was fit to the training data. Next, the testing data was reduced from the original *m* features to the selected *k* features. At this point the testing data was ready to be scored. It was unclear what K was appropriate for Select K Best, and what K was appropriate for KNN. Thus, each individual's dataset was scored with the amount of neighbors ranging from 1 to 10, and at 15 to 510 features each. The scoring was done in bulk with two loops iterating through the number of neighbors to use, and the number of features to use. It took approximately 40 seconds for each individual's data to be formatted, tested for each combination of neighbors and features, and then written to a CSV file. Figures 2- 4 display the results of the scoring, and the pattern of each individual's data.

Analysis

With *m* features, the preprocess running time will be denoted as O(m) due the data being provided. The algorithm for feature selection is known as Select K Best, which performs a univariate statistical test on each feature, and selects the K features which performed the best. A univariate statistical test performs some statistical test on a single variable (in this case a feature) at a time. The test chosen was Analysis of Variance

(ANOVA), which produces an F-value that is used to score a feature. In short, the F-value describes the how similar features are to one another. This is a useful measure as features similar to each other offer no new information, and could potentially be cause for overfitting. Calculating an F-value requires calculating two related values and taking their ratio. First is the variation between sample means. This requires calculating an overall mean involving each feature, and then calculating a sum of squares. Producing an overall mean simply requires taking the average over each data point; this requires a quadratic asymptotic time complexity. Essentially, if there are *m* features, and *n* samples (representing the amount each letter was trained collectively), then it forms an n x m matrix which must be iterated through. This results in a worst case running time of O(nm). Next, the variation within individual samples must be calculated. This requires iterating through each feature, calculating its mean, followed by calculating a sum of squares. Similarly, this will require a O(nm) time complexity to calculate the variation within each feature. The final step of the ANOVA process requires a division operation to repeat *m* times. In short, the feature selection algorithm Select K Best has an asymptotic time complexity of O(nm); the number of features derived from raw input data times the number of times each letter was trained collectively.

The algorithm for classification is kNN which performs a brute force search by calculating the euclidean distance to every data point and queries the K nearest points for classification. The brute force algorithm for kNN has a time complexity of O(Dn + kn). D represents the number of dimensions in a sample and n represents the number of samples. This would require O(Dn) time to calculate the distance to each sample. After the distances have been calculated, it takes O(kn) time to determine the k closest neighbors. Thus, kNN has a time complexity of O(Dn + kn). It should also be noted that the dimensionality D is the number of features selected by Select K Best. Thus, if the Select K Best returns a number of features C, they must be no greater than D, and the running time is more accurately O(Cn + kn).

In summary, the training phase of the algorithm must perform preprocessing and feature selection, giving the training phase a time complexity of O(nm). The testing phase then requires preprocessing and classification, giving the testing phase a time complexity of
O(Cn + kn).

Results

The results from the algorithm were produced from three individual datasets, which were provided in a normalized format. The datasets consisted of approximately five samples for each letter, and nearly five-hundred features for each sample. The data was then split into training and testing data. The accuracy for each dataset is summarized in Figures 2 - 4. Each figure corresponds to an individual, and each line on the figure corresponds to the amount of K neighbors used in kNN. The legend on each graph indicates how many neighbors a line represents. The x-axis indicates the amount of features used, has a range of 15 to 510 features, and is incremented in powers of five. The y-axis indicates the accuracy, and is normalized between 0 and 1. Thus, an accuracy of 0.53 on the graph translates to an accuracy of 53%. The results show that each individual's data is capable of varying widely from one another. The highest accuracy for individual one was 61.8% when using only a single neighbor, and a range of features from 185 to 200. The worst accuracy was 29.1% when using ten neighbors, at 25 features. Individual two had an accuracy significantly better at 81.2% with 2 neighbors, at 100 features. The worst accuracy was 25.5% at 10 neighbors, and a range of features from 130 to 155. Individual three performed similarly to individual two with the highest accuracy at 81.5% with a single neighbor, and a range of features from 425 to 450. The worst accuracy was 24.1% at 9 neighbors with a range of features from 15 to 30. This data suggests that a low number of neighbors is best for classification, however, the number of features may depend completely on the individual.

The running time for each input file was measured as the time to perform the feature selection and the time to score the accuracy of KNN on the testing data. The average time was 23.6ms, 13.4ms, 68.8ms for individual's one, two and three respectively. Each of these times are well within the demands for a real time application. Since the data suggests that a small number of neighbors is favorable, computation will likely be dominated by the dimensionality of the data. The data suggests this since individual three's computation time is 2.91, and 5.13 times larger than individual one and two's computational times, respectively. This is likely due to the fact that individual three requires at least 425 features, a significant increase over the other two individuals. However, even with this larger set of features, the computation time of 68.8ms is highly acceptable.

An interesting observation to note from the graphs is that each follow a similar trend. As the number of neighbors used increases, the accuracy tends to decrease. This is suspected to be caused by the curse of dimensionality, and the variability of the EMG sensors.

Conclusion

In order to do real-time classification of the AFA, we decided to use Select K Best to pick out the identifying features in order to classify them using KNN. Our final results did vary in accuracy, but we did get an acceptable computational time for real time applications. We can conclude that a low number of neighbors is best for classifying and the best number of features solely depends on the user. The future of this project is to increase the accuracy by trying Principle Component Analysis (PCA) over Select K Best. Instead of picking out a subset of the best features from our data, PCA will transform our a highly dimensional data into an artificial set with a lower dimensionality while retaining most of these features.

Figures

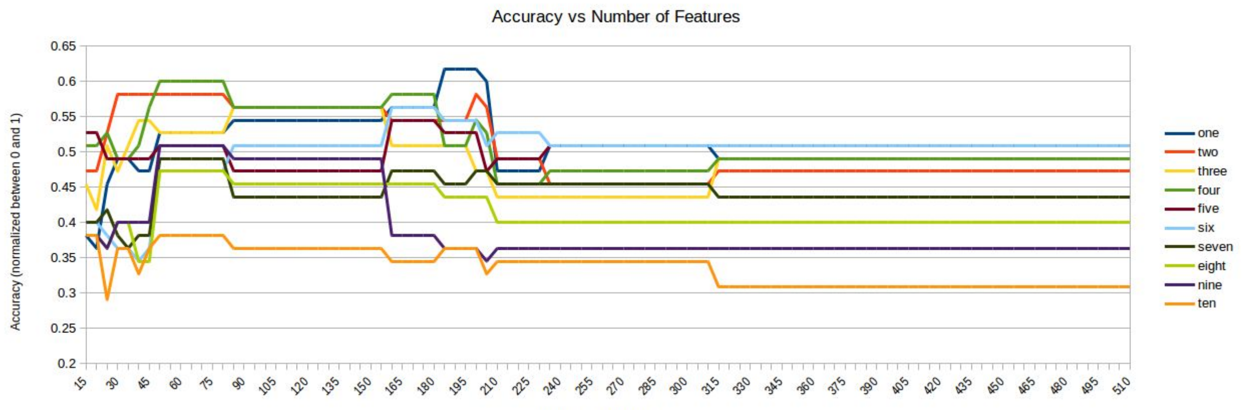

Figure 2

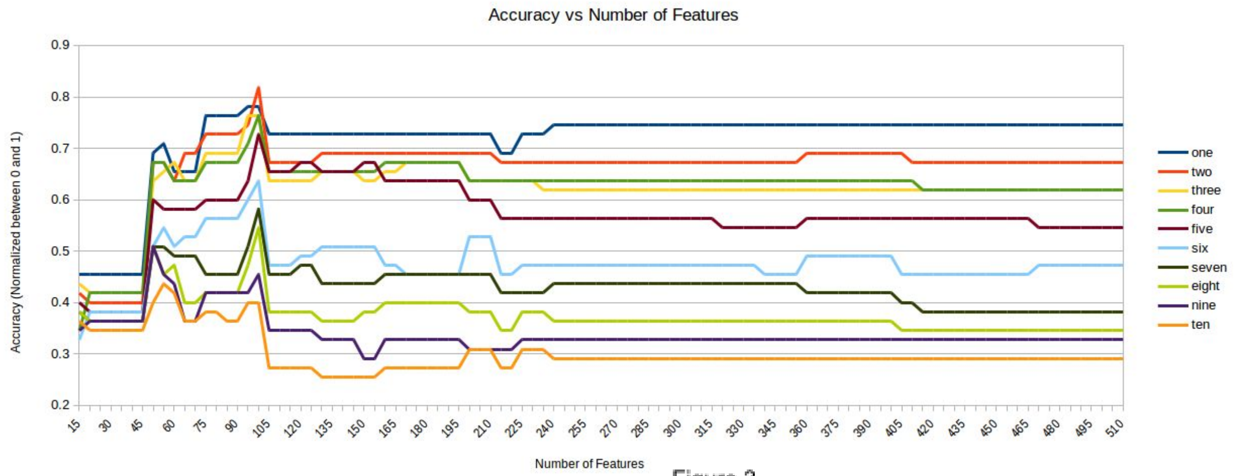

Figure 3

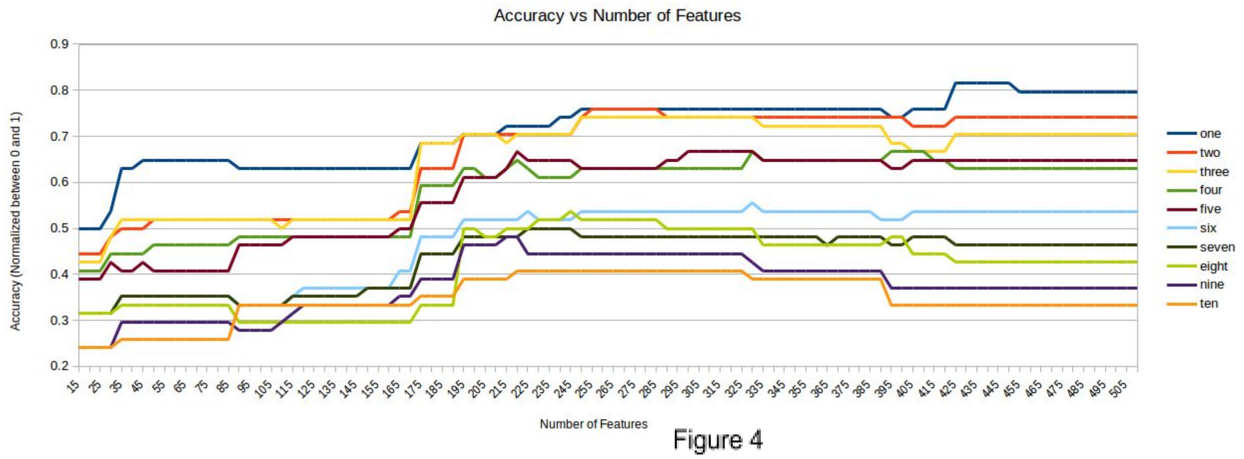

Figure 4

References

1. World Health Organization. (2017). Deafness and hearing loss. Retrieved from http://www.who.int/mediacentre/factsheets/fs300/en/

2. Paudyal, P., Banerjee, A., & Gupta, S. K. S. (n.d.). DyFAV : Dynamic Feature Selection and Voting for real-time recognition of fingerspelled alphabet using wearables.

3. Paudyal, P., Banerjee, A., & Gupta, S. K. S. (n.d.-b). SCEPTRE: a Pervasive, Non-Invasive, and Programmable Gesture Recognition Technology. https://doi.org/10.1145/2856767.2856794

4. Yiqiang, C., Wen, G. A. O., & Jiyong, M. A. (2000). Hand Gesture Recognition Based on Decision Tree. *International Symposium on Chinese Spoken Language Symposium*. Retrieved from http://www.isca-speech.org/archive_open/archive_papers/iscslp2000/or03/027.pdf
5. National Institute of Deafness and Other Communication Disorders(NIDCD). (2014). American Sign Language. Retrieved from https://www.nidcd.nih.gov/health/american-sign-language

6. Smith, L. I. (2002). A tutorial on Principal Components Analysis Introduction. *Statistics*, *51*, 52. https://doi.org/10.1080/03610928808829796

7. Course, A. 2002-. (2002). The Curse of Dimensionality. *Most*, 1–3.

8. Martin, J. K., & Hirschberg, D. S. (1996). On the complexity of learning decision trees. *International Symposium on Artificial Intelligence and Mathematics*, 112–115.

9. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011