Accelerometer Gesture Recognition

Michael Xie
xie@cs.stanford.edu

David Pan
napdivad@stanford.edu

December 12, 2014

Abstract

Our goal is to make gesture-based input for smartphones and smartwatches accurate and feasible to use. With a custom Android application to record accelerometer data for 5 gestures, we developed a **highly accurate SVM classifier** using only 1 training example per class. Our novel **Dynamic-Threshold Truncation** algorithm during preprocessing improved accuracy on 1 training example per class by 14% and the addition of axis-wise **Discrete Fourier Transform** coefficient features improved accuracy on 1 training example per class by 5%. With 5 gesture classes, **1 training example for each class**, and 30 test examples for each class, **our classifier achieves 96% accuracy**. **With 5 training examples per class, the classifier achieves 98% accuracy, which is greater than the 10-example accuracy of other efforts using HMM's**[1, 2]. This makes it feasible for a real-time implementation of accelerometer-based gesture recognition to identify user-defined gestures with high accuracy while requiring little training effort from the user.

1 Introduction

The touchscreen interface of smartphones and smartwatches limits the interaction with the phone or watch to the GUI-based point-and-click interactions. Similarly, current life-tracking software largely relies upon user input to collect data, which is often not effective. Software that collects and interprets data should not have the deficiency of needing manual input of data. By capturing acceleration data, it is possible to respond to gesture-based commands and to interpret daily activities as an automated data tracker. The objective of this project is to interpret and respond to gestures using accelerometer data. The real-life implementation of this project would be to automatically detect gestures in a stream of accelerometer data and respond to user-defined gestures. One major challenge for this implementation is processing the data so that the classifier only takes into account the segment of data that corresponds to the gesture and is invariant to gestures starting and ending at different times. The second major challenge is to achieve high accuracy on 1 to 3 training examples per class; the user would then only be required to define their gesture a few times before being able to use the gesture. We assume that in the real life implementation, the training set would grow with time with correct classification of new instances of the gesture, improving the performance over time.

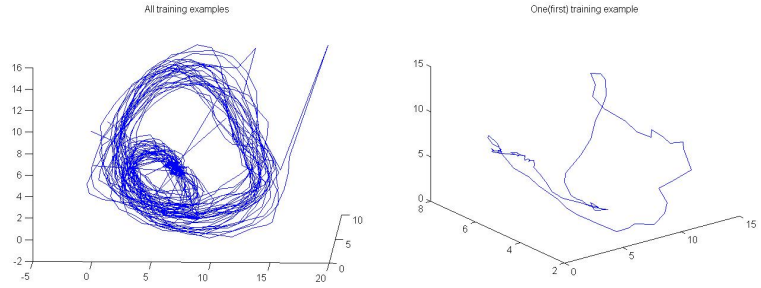
In this project, we define a **basic gesture** to be a continuous sequence of movements. **Compound gestures** can be made of several basic gestures in sequence, which would be a layer of classification above the classification of basic gestures. In this project, we focus on the classification of basic gestures.

2 Data and Preprocessing

To collect data, we constructed an Android application that records accelerometer data upon the click of an onscreen button and stores the data in a file with the gesture label provided by the user. The application collects a fixed-length matrix of 100 points of acceleration data from the x, y, and z axes sampled at 50Hz over a 2 second window. Let $x(t), y(t), z(t)$ be discrete time functions that return the acceleration in each axis at time t. We reshape this 3-by-100 matrix such that each raw training example $e^{(i)}$ is as follows:

$$e^{(i)} = [x(1), ..., x(100), y(1), ..., y(100), z(1), ..., z(100)]$$

Each raw example and label is a pair $(e^{(i)}, l^{(i)})$ where $l \in \{0, ..., n-1\}$ for n different gestures in the training set. The gestures that the training examples represent are labeled by the user and represents the ground-truth data from which we build our model. For the current purposes of this project, we assume that when acceleration data is being recorded, the phone is in the right hand, upright, and screen facing away from the palm. Our model



(a) Aggregate raw data for class 'O' (b) A single 'O' example

Figure 1: Accelerometer data for the 'O' gesture

further assumes that a gesture starts from rest and that the starting position of a gesture is irrelevant, which are reasonable assumptions for gestures based on our definition in the previous section. Our dataset consists of 31 examples each of 5 different classes of gestures, 'O', 'X', 'Z', 'W', and 'V'. The names of the gestures correspond to the movement the gesture encodes in the air, i.e. gesture 'O' is a counter-clockwise circle drawn in the air using the device.

2.1 Filtering

To decrease noise from natural shaking of the hands and from idle moments in the beginning and end of the gesture, we employ a low-pass filter to smooth the data by axis. Initially, we experimented with a box filter of window size 10, 15, 20, 25, and 30. The average error after cross validation decreased with increasing window size until converging at around 20-25. A Gaussian filter was also tested, and while it represented an increase in accuracy of the classifier, it yielded poorer results. For our final results, an exponentially weighted moving average filter with $\alpha = 0.3$ outperformed the rest. An exponentially weighted moving average filter is defined recursively as

$$s_t = \alpha \cdot x_{t-1} + (1 - \alpha) \cdot s_{t-1}$$

where s_t is the smoothed point at time t and x_t is the actual data at time t. The exponentially weighted moving average filter weights the time-step before the highest, with exponential decay in weight for further time steps in the past. This is intuitive, as the position of the gesture can be modeled as obeying the Markov property, where the position of the next time step is dependent only upon the state in the previous time step. For this reason, there are many approaches to this problem that use Hidden Markov Models [1]. We have found that the SVM approach we propose here gives better performance on a small number of training examples and has faster runtime [2].

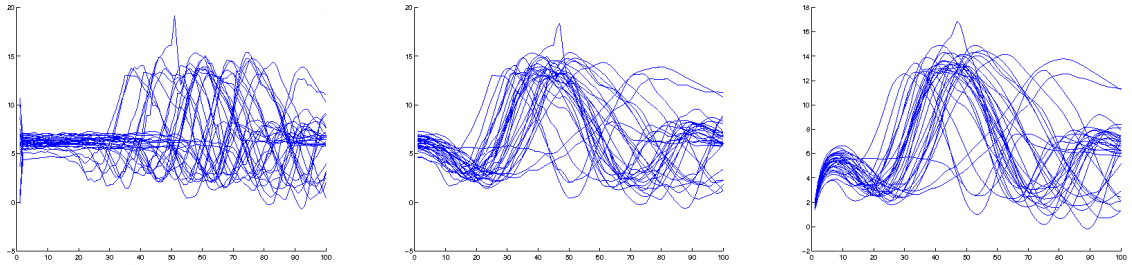
2.2 Dynamic-Threshold Truncation

We want to remove data recorded before the gesture actually starts and after the gesture actually ends in order to capture the "essence of the gesture." In typical truncation, the longest continuous data segment starting at the beginning of the data that is below some fixed threshold is removed, and so is the longest continous data segment ending at the end of the data that is below the threshold. Specifically, a data point (an acceleration vector) is below the threshold if the norm of the difference between the data point and the intial point (first point or last point of the data depending on which side the segment is on) is below the threshold. However, fixed-threshold truncation had inconsistent peformance across the number of training examples.

We modified truncation to use a dynamic threshold that depends on the standard deviation of the data to be truncated:

$$\textit{threshold for data} = \alpha\sqrt{\textit{standard deviation of data}}$$

where α is a nonnegative constant. The intuition is that a higher standard deviation means that the essence of the gesture completes in a shorter time interval, so the threshold should be increased to improve the chance that the data is truncated around that time interval; this helps overcome the problem that noise may stop truncation prematurely before the essence of the gesture is captured in a tight window. We chose $\alpha = 0.22$; any α between 0.20 and 0.37 increases SVM accuracy by at least 10% with one training example and no additional preprocessing or feature extraction.



(a) Raw x-axis data for gesture 'O' (b) After Dynamic Truncation (c) After smoothing

Figure 2: Preprocessing visualized for x-axis data of 31 examples of the 'O' gesture

2.3 Interpolation

After truncating the data to capture the essence of the gesture, we linearly interpolate the data back to the 100 point length. By doing this, much of the phase differences between gestures are gone.

2.4 Normalization

Mean and variance normalization was tested due to the possibility of gestures being completed at different speeds, and hence resulting in acceleration vectors that represent the same gesture with differing acceleration magnitudes. However, standard score and feature scaling normalization decreased classifier accuracy.

3 Feature Selection

We can see after preprocessing the data that each gesture has a characteristic "signal" for each axis. However, there may be phase differences due to starting the gesture at slightly different times that would shift the acceleration values at each time point slightly for each example. We aim to add features that take into account the time-series data as a whole to combat these differences.

3.1 Discrete Fourier Transform

The main intuition for the FFT is that, if a gesture has many movements within the same time period, then the individual axes will have higher frequency components. We posit that if acceleration data over a 2-second window for a given gesture is repeated over consecutive 2-second windows, then this time-series data is periodic over a period of 2 seconds, or $N = 100$ time points. Using a 21 point Fast Fourier Transform, we output N 100-periodic complex coefficients X_k that encode the amplitude and phase of sinusoidal components. We take the amplitude $a_k = \frac{|X_k|}{N} = \frac{\sqrt{Re(X_k)^2 + Im(X_k)^2}}{N}$, which is magnitude of each complex number, as features. Thus, we ignore phase differences in each example. We append these a_k to each example, resulting in 363-dimensional example vectors, where each spatial dimension has $100 + 21$ features. We chose a FFT with a small number of points to limit the dimensionality of the data while adding valuable information; we found that a 21-point FFT performed well. Adding n-point FFT where n = 0 through 36 does not decrease classifier accuracy.

3.2 Mean and Variance

Adding the mean and variance of each training example was also tested, but did not have any noticeable effect on the accuracy. The intuition behind the attempt is that more variance implies more dynamic gestures and differing mean in each axis implies a gesture that tends to accelerate in a direction over another. The Naive Bayes classifier uses these features to classify with reasonable performance, so we believe that while these features encode information about the gestures, this information is encoded in other features when using the SVM.

4 Classification Techniques

4.1 Nearest Centroid Classifier

In the Nearest Centroid Classifier, we create a "mean" gesture for each class that represents the average of all examples of the class in the training set. Given a test example, we then predict by outputting the class of the "mean" gesture with the shortest Euclidean distance to the test example. This represents a very simple baseline performance. This classifier had very good performance for such a simple solution, which suggests that Euclidian distance is a good measure of similarity between gesture time-series data.

4.2 K-Nearest Neighbors

Given a test example, the K-Nearest Neighbors algorithm outputs a classification based on which classes the majority of the "closest" K neighbor training examples belong to. This is essentially seeing which known gestures the new example looks like, and outputting that result. This is sensitive to time shifts and scaling differences (fast or slowly executed gestures). In our testing, we used $K = 4$ neighbors and the Euclidean distance function as the measure of distance. This classifier had reasonable performance on five classes that was comparable to the SVM with 10 training examples per class. However, this algorithm does not have enough information to classify gestures with only 1 training example per class, as the 4 neighbors were likely all of different classes.

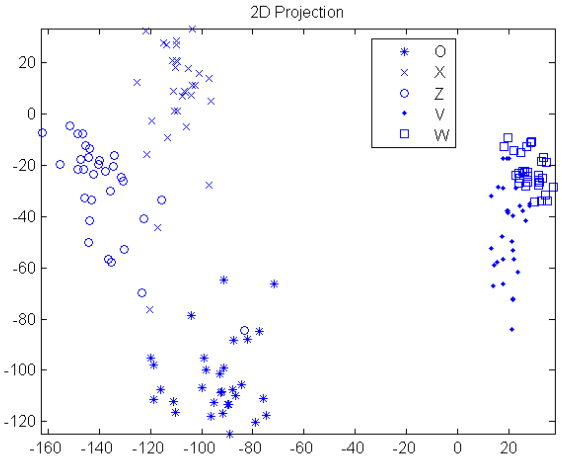


Figure 3: 2D low-rank projection using SVD with feature extraction and Dynamic Truncation, maxmimizing differences between data points. Preprocessing/Feature Extraction separates the data into well-defined clusters.

4.3 Naive Bayes

The features used are the mean and standard deviation of the accelerometer data. Assuming conditional independence, we weighted the probability of these features against each class, as well as the probability of each gesture class in total. However, if this is used in gesture detection in a smart device and online learning is used, input from the user provides more accurate "custom" probabilities up to an extent, where a user that heavily prefers one or two gestures may cause the overall probabilities of each gesture to be skewed. Interestingly, filters had little effect on the accuracy of the Naive Bayes classifier. This may be because the filters have less overall effect on the features that Naive Bayes uses, such as the mean and standard deviation.

4.4 SVM

From the perspective of the SVM, each example is a 363 dimensional data point. In this case, we have a large number of features in comparison to the number of examples, but the large margin properties of the SVM allows it to learn a classifier that does not overfit. We find that a linear or polynomial kernel performs well for the data, but the Gaussian kernel doesn't perform as well.

5 Results and Ablative Analysis

The above techniques were tested using 31 examples each of 5 different classes of gestures, 'O', 'X', 'Z', 'W', and 'V'. With the small size of our dataset, 10-fold cross validation tended to leave out classes in each fold. Instead, 1000 iterations of bootstrapping cross validation was used, in which we randomly sampled a given number of examples from each class and tested on the rest of the data. This simulates the real-life application, in which the user would supply the application 1 to 3 examples to define each new gesture. With subsequent correct classifications, the training data for each gesture class grows. Out of the above techniques, the SVM classified the gestures the best,

especially with a low number of examples; it was able to achieve 96% accuracy on 1 training example per class and 99% accuracy with 10 training examples per class. Many other implementations use Hidden Markov Models, which achieve 98% accuracy on 10 or more training examples [1, 2]. Other implementations, using Dynamic Time Warping, achieved 98% on one training example. However, the gestures used are simpler (move left, move right) and perhaps more distinguishable then the ones we used [3].

Accuracy of classifiers given a uniform, randomly sampled number of examples per class:

Classifier	1	5	10
SVM	0.9630	0.9872	0.9909
Naive Bayes	NaN	0.9645	0.9838
K-Nearest Neighbors	0.2000	0.9827	0.9846
Nearest Centroid Classifier	0.9523	0.9848	0.9859

Accuracy of classifiers given a uniform number of examples per class without features from FFT:

Classifier	1	5	10
SVM	0.9081	0.9648	0.9784
Naive Bayes	NaN	0.9605	0.9804
K-Nearest Neighbors	0.2000	0.9351	0.9494
Nearest Centroid Classifier	0.9001	0.9524	0.9562

Accuracy of classifiers given a uniform number of examples per class without Dynamic Truncation:

Classifier	1	5	10
SVM	0.7718	0.8881	0.9291
Naive Bayes	NaN	0.8773	0.9286
K-Nearest Neighbors	0.2000	0.8520	0.9173
Nearest Centroid Classifier	0.7604	0.8723	0.8960

Baseline accuracy of classifiers without preprocessing or additional feature extraction:

Classifier	1	5	10
SVM	0.7729	0.8925	0.9271
Naive Bayes	NaN	0.8871	0.9340
K-Nearest Neighbors	0.2000	0.8484	0.9167
Nearest Centroid Classifier	0.7807	0.8668	0.8840

6 Conclusions and Discussion

Our results suggest that highly accurate, low-training accelerometer based gesture recognition is feasible. Importantly, Dynamic-Threshold Truncation results in a 14% increase in accuracy with one training example and increases performance of every classifier; FFT features add another 5%. However, without Dynamic-Threshold Truncation, FFT features have no effect. The main topic of discussion is the relationship between standard deviation and truncation threshold.

For future work, we will explore the nuances of Dynamic-Threshold Truncation further, experimenting with more threshold functions. We will consider using rotational normalization to account for gestures done at different orientations [4]. We aim to test robustness further by adding more gestures and more training examples. Finally, we'd like to test performance on real-time data by implementing this in our data-collecting Android application.

References

[1] Klingmann, Marco. "Accelerometer-Based Gesture Recognition with the iPhone."

[2] Prekopcsak, Zoltan (2008). "Accelerometer based real-time gesture recognition". In Proceedings of the 12th International Student Conference on Electrical Engineering, Prague, Czech Republic.

[3] Jiayang Liu, Jehan Wickramasuriya, et al (2009). "uWave: Accelerometer-based Personalized Gesture Recognition and Its Applications." In Pervasive Computing and Communications.

[4] David Mace, Wei Gao, Ayse Coskun (2013). "Accelerometer-Based Hand Gesture Recognition using Feature Weighted Nave Bayesian Classifiers and Dynamic Time Warping." In the Proceedings of the 2013 International Conference on Intelligent User Interfaces.

[5] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011.