Biometric in Motion: Identification Using Acceleration Data

Zexi Mao
zexim@andrew.cmu.edu

Siqi Tan
siqitan@andrew.cmu.edu

Yuchen Wu
yuchenw@cs.cmu.edu

Yimeng Zhang
yimengzh@cmu.edu

Abstract

As cellphones and other smart devices are more closer to our daily life than ever, we are curious to know how the data gathered by sensors on the devices, especially identification via accelerometers, impacts our lives. In this project we try to answer this question with the tools machine learning provides. We will first study how to implement such biometric technique by retrieving useful identification information from raw accelerometer data. And then we will show how such technique impacts our lives.

1 Introduction

1.1 Motivation

Ubiquitous computing had never been so close to us as our phones become more powerful and smarter today. Sensors equipped in our cellphones like GPS, gyroscope, accelerometer and even barometer collect data from the surrounding environment and from ourselves. This information collected from our cellphones and other mobile devices can boost many more applications than just adjusting the brightness and rotating the screen. Google uses the GPS data from our phones to provide real-time traffic conditions. Apps can track how well you sleep by just putting your phone beside you on bed. Indoor localization takes advantage of wireless fingerprint, sound from microphone, footsteps from accelerometer and even colours from the camera to tell you where you are when GPS signal is weak inside buildings. More fancy applications far beyond the original purposes of these sensors are on the way. How to retrieve more information from the raw sensor data is one of the hot and promising topics today.

Accelerometer is one of the most interesting sensors in our cellphone which records the acceleration data in 3D. Posture of the phone, gestures and footsteps of the user, and even the user's sleeping condition can be measured by the accelerometer.

In this project, we are trying to develop a novel usage of the accelerometer: biometric identification. In other words, can we identify the user by only looking into how he/she moves? We believe that everyone has his/her own unique pattern of movement. If this assumption is true, we can identify the person when we match the current data from the accelerometer to the historical data we have learned.

To achieve this, we adopt the already available accelerometer dataset gathered by Kaggle. By simply looking into the raw dataset, we can tell not only noise stands between us and our goal, quantity and quality of the raw data also make it harder and more necessary to refine the data before we apply any powerful classifier on it. Our first effect will be on preprocessing data. Then we will adopt several useful technique learned from machine learning to build a classifier for identification. With

this preliminary approach, we will further revise and refine both the preprocessor and classifier. Hopefully we will finally achieve a pretty well identification technique in this iterative way.

If possible, we will evaluate our technique not by testing it on the Kaggle dataset, we will also develop apps for our smart phones to collect real time data for learning and evaluation. We hope this approach will give us the clue of how to protect our id privacy.

We will learn a few things from this project besides a better understanding of machine learning itself if our machine learning algorithms finally identify users. First, the assumption about pattern of movement could be true. Second, applications such as anti-theft, health monitoring and emergency detection that adopt this novel identification technique will be available in the near future. Third, we will be able to know how much data from the accelerometer is sufficient to cause the leak of one's identity, which can trigger a serious privacy issue.

1.2 Related Work

A human beings walking gait can reflect the walkers physical characteristics and psychological state, and therefore the features of gait can be employed for individual recognition. The use of accelerometer data for biometric identification is relatively new but has been increasingly explored in recent years. Existing methods for gait recognition have shown good performance.

Gafurov et al. [1] attach multiple sensors to a subject at different body parts. Xu et al. [2] developed an Android App to collect gait acceleration data. With a reasonably sized dataset, by matching gait patterns across different paces, they show preliminary results indicating that not only can smart phones be used to identify a person based on their normal gait but also that there is potential to match gait patterns across different speeds.

Tao et al.[5] focus on the representation and pre-processing of appearance-based models for human gait sequences. Two major novel representation models are presented, namely, Gabor gait and tensor gait. Experiments show that the new algorithms achieve better recognition rates than previous algorithms.

Pan et al. [4] proposed algorithm based on signature points, instead of the whole gait signal. They consider acceleration-based gait recognition insensitive to changes of lighting conditions and viewpoint. Their algorithm firstly extracts signature points from gait acceleration signals, and then identifies the gait pattern using a signature point-based voting scheme. The experimental results shows the accelerometer-based gait biometrics is promising.

Kwapisz et al.[3] collect some data and also perform identification experiments . Based on the 600 raw accelerometer readings, they generated 43 features, which are variations of 6 basic features including average acceleration value, standard deviation, time between peaks and so on. They applied two classification techniques decision trees neural networks to classify and the identification performance turned out to be fairly good.

2 Methodology

In this section, we describe the methods we have used to pre-process the raw data and classify the refined data. This is not necessarily the final version of our approach.

2.1 The raw data

The data set can be easily acquired from Kaggle which consists of 3 parts: a training dataset, a testing dataset and a question set. Each single sample in the training and testing dataset contains a time stamp in milliseconds, acceleration measurements in 3 dimensions, and an associated DeviceId. There is an Android application published on Google Play for such data collecting purpose. This application runs in background while reporting accelerometer data and the corresponding device id to a central server.

In the training dataset, there are 30 million samples, which are collected from 387 different devices as labelled. In the testing data set, there are also 30 million samples, but without DeviceId. These

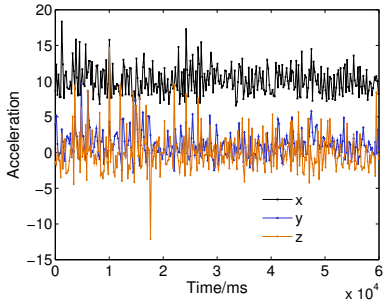

Figure 1: A piece of raw data from a good trace. Data is almost uniformly distributed along time.

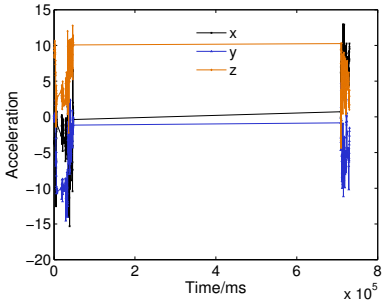

Figure 2: A piece of data from a bad trace. There is no useful data in most of the samples over time.

samples are demarcated into 90,000 sequences, each with a SequenceId. In the question set, for each SequenceId, there's a proposed DeviceId.

The challenge on Kaggle is to tell whether each of the SequenceId is mapped correctly to its DeviceId. For our study, we aim at a more general version of the problem of classification, which is telling the correct DeviceId for any given sequence of sample.

2.2 Refinement

The raw dateset has already be prepared preliminarily by Kaggle. First, the devices with less than 6000 samples are removed ahead. Second, the sequences shorter than 10 seconds are eliminated as well.

However, the dataset is still far from beautiful.

First, the accelerometers have various sampling rate. A larger interval between two samples may leads to much information lost between the two sample point. Furthermore, the acceleration on each dimension changes rapidly in such low sample rate. A variance of sampling interval within one device's trace even makes it harder to retrieve features from the trace.

Second, according to the nature of Android system and the low privilege of an application, sometimes the app cannot access accelerometer, sometimes the app itself is swapped out or killed by the operating system. Thus, a great part of the sample data is not well distributed over time.

Figures 1 and 2 show the read untouched raw data from two outstanding device.

To overcome the two challenge above. We propose our three phase refinement: splitting, smoothing and re-sampling.

The idea of split is very simple. For the first step of splitting, we scan a sequence of data from a identical device in chronological order. Once the interval between our current sample point and the previous one is greater than a threshold T_{split}, the samples scanned before are grouped into one subsequence. This process continues until all the data from each identical device is split well. Then we go to step two of splitting. In this step, if any of the sub-sequences is shorter than threshold T_{short}, we simply eliminate it as we believe a too short sequence brings more noise than useful information. If, unfortunately, all the data from a particular device is abandoned, it left us no choice but keeping the longest among the sub-sequences of it.

The next phase is to make the sequence smoother, which hopefully could reduce the level of noise. To this end, interpolation is the idea comes to us so far. We simply adopt cubic spline as our interpolating function. Thus, we transform a sequence of data into a continuous curve. Some missing pieces of data are "reproduced" in this way. This phase makes it easier for feature retrieval as we can get a sequence of data by any uniformed time interval. We should notice that noise is not necessarily reduced well in this manner. The curve we get contains almost the whole noise from the original data as it is with respect to every single data point from the raw data. We believe a better smoothing function will help while we are still looking for such one.

The third phase benefits from the smoothing phase as it is quite easy to re-sample data point from a smooth curve. We set a parameter, $\tau_{resample}$, to be the re-sampling interval. With this interval, a new sequence of data is generated based on the original data trace.

After the three phases we introduced above, the refined data trace is produced from the raw data. Next, we will propose the way we select features from such refined trace.

2.3 Feature extraction

Feature selection as well as extraction is the most important part of the entire classification method. A great feature should be pinpoint to reveal the underlying differences between the users (also known as DeviceId in this project). In this section, we describe how we extract features from the preprocessed data.

The only assumption we made so far in this research project is that a human being moves in his/her own identical pattern. This pattern could be a sequence of outstanding events, it also could be some repeatable events as well. Noticing that accelerometer records data with an unpredictable length and quality, we cannot assume that the complete sequence of events could be well recorded. Thus, our hope lies on the latter one.

It is nature to use frequency analysis for repeatable events. In our approach so far, simple Fast Fourier Transformation is adopted. The basic idea is just making FFT on the refined data sequence we talked about in previous section.

A better idea is to make windowed FFT on them. The reason is, as each of the sequence has different length, if we make full FFT on each sequence, the frequency components produced by FFT have different meanings on different lengths of FFT. For example, we make full FFT on two sequence with length 17 and 29 respectively, some of the point in frequency domain are not matched. The results will be hard to interpret and compared. To this end, we make the transformation in a given window size ,w_{fft}. Thus, multiple transformations are performed in a sliding window manner. We can average each corresponding point from different transformation on the sequence, as they are perfectly aligned.

However, another concern arises upon us as if we make one windowed FFT on a pretty long sequence, some information is erased by the sliding window averaging. This is true if we believe that a longer sequence could contain more information than a shorter one.

To this end, we prefer to perform partitioning technique on the sequence before the windowed FFT. There are two threshold in the partitioning process, one is the lower bound of number of generated sub-sequences, $k_{\#seq}$, the other is the upper bound of the maximum length among all the generated sub-sequences, l_{maxL}. The procedure of partitioning goes as follows. The input sequence is halved into two sub-sequences until either of the following conditions satisfied. A) the number of sub-sequences is greater than $k_{\#seq}$. B) all the sub-sequences are short than l_{maxL}. In this manner, we make more independent sequences, hopefully keeping more helpful information along. We also prevent over cutting by using the second threshold.

So far, the results of FFT is the extracted features as we shown above. As there are three dimensions, and the independent values of FFT with window size w_{fft} is $1/2w_{fft}$, we build a feature space with $3/2w_{fft}$ dimensions

2.4 Classification

This is the ongoing part of out work. The basic idea is to leverage various kinds of classifiers and optimize their parameter for more accurate results.

We start from some naive classifiers. The performance of K-Nearest-Neighbor depends on the number of K and the definition of distance function. This classifier leads us to a baseline result. Then we move on to popular classifier such as support vector machine. Optimizing SVM via kernel and parameter selection will be discussed in next section.

3 Implementation and Evaluation

In this section we show how our approach performs. We will also manipulate and justify the selection of parameters of both reprocessing and classifiers.

3.1 Parameter selection

In this subsection we discuss how to determine the parameters and thresholds we defined in the previous sections.

T_{splite} is set to be 10 second in consistent with the preprocessing setting done by Kaggle. It is pointless to set the threshold above 10 seconds as those sequence shorter than 10 seconds have already been eliminated. We keep this threshold as large as possible to generate longer continues sequences. T_{short} is set as 4.8 seconds for training set and 3.2 seconds for test set based on empirical study.

We let $\tau_{resample}$ be 200 millisecond. This value is chosen based our statistic result of the distribution of the original data. Any value near 200 ms is a reasonable choice because most of the intervals lie in this range. Choosing a value deviating from 200 ms could misinterpret the data. (put a figure here for CDF of interval?)

The window size of FFT, w_{fft}, is 64 sample points, which means a window with length 12.8 seconds. We chose this value to balance the details of data and dimensions of feature space.

$k_{\#seq}$ and l_{maxL} are more flexible as the result is less sensitive on these values.

3.2 Optimizing classifier

This is the To Be Done part.

3.3 Results

Currently we are ranked at 250 out of 500 competitors on Kaggle.

Appendix

Our plan for future goes here.

References

[1] D. Gafurov, E. Snekkenes, and P. Bours, "Gait authentication and identification using wearable accelerometer sensor," in *Proceedings of IEEE Workshop on Automatic Identification Advanced Technologies*, June 2007, pp. 220–225.

[2] F. Juefei-Xu, C. Bhagavatula, A. Jaech, U. Prasad, and M. Savvides, "Gait-id on the move: pace independent human identication using cell phone accelerometer dynamics," in *Proceedings of IEEE 5th International Conference on Biometrics*, 2012, pp. 8–15.

[3] J. R. Kwapisz, G. M. Weiss, , and S. A. Moore, "Cell phone-based biometric identification," in *Proceedings of Fourth IEEE International Conference on Biometrics: Theory Applications and Systems*, September 2010, pp. 1–7.

[4] G. Pan, Y. Zhang, and Z. Wu, "Accelerometer-based gait recognition via voting by signature points," in *Electronics Letters, 45(22)*, 2009, pp. 1116–1118.

[5] D. Tao, X. Li, X. Wu, and S. Maybank, "General tensor discriminant analysis and gabor features for gait recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence ,29(10)*, October 2007, pp. 1700–1715.