SMART DEVICE

USER BEHAVIOR CLASSIFICATION AND LEARNING

WITH HIDDEN MARKOV MODEL

A THESIS

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE DEGREE

MASTER OF SCIENCE

BY

RAN AN

DR. WU SHAOEN – ADVISOR

BALL STATE UNIVERSITY

MUNCIE, INDIANA

MAY 2016

Contents

1 INTRODUCTION

In 2016, there expects two billion people using smartphones worldwide [1]. The main reason of the popularity of the smartphones is that smartphones are convenient for users to access various online services. Every day, people use smartphones to check emails, log on social networks applications, such as Facebook, store personal information, data and files onto the clouds and etc. During the use of smartphones, people likely save their personal information, photos, and passwords into the websites and applications. So, nowadays, smartphone is not only a communication device for calling and texting, but a personal assistant device which is full of personal and private information. Obviously, the devices cannot be guaranteed of complete safety. Unreliable and simple passwords, and ubiquitous attackers present severe threatens to users' personal and sensitive information stored on the devices. Therefore, providing a reliable authentication to these devices is an essential requirement.

Passwords used to be the only option to authenticate people to access their devices. In recent years, more and more devices provide biometric sensors such as fingerprints or drawing on screen as alternative authentication options to unlock the devices, but still require passwords as the last help resource in case that biometric solutions fail to work with repeated tries. Even though, these new options cannot avoid the usability issue. A device still requires its user to repeatedly enter their passwords or pins, touch the screen with drawing, or place a finger on the fingerprint sensor. Therefore, how to minimize the usability issue of inconvenience is significant, essential but challenging. In this thesis work, we propose a non-password software-only solution which is a passive and continuous authentication. It does not rely on the traditional authentication inputs such as password or biometric information, but on user's historical behaviors in using a device.

In recent years, technology companies such as Google and Apple provide more and more built-in sensors to increase user experiences. The larger amount of data collected by sensors in modeling user behavior to creates a lot of opportunities for improving mobile device security.

In this paper, we propose a multi-sensor-based authentication framework for smartphone users. The framework leverages accelerometer, orientation, and touch size data which are gathered from an Android smartphone, and then, it uses Hidden Markov Model to train a user's figure gesture and handholding pattern, which is dynamically authenticate the legitimate user of the device and distinguish the user from other unauthorized users.

Results from the experiments show that our system can achieves 0.89 accuracy in identifying users. Also, the system can detect 65 cases when the device is operated by non-owners. Only 25 cases trigger false alarm when it's operated by the owner.

The rest of this thesis paper is organized as follows. We introduce built-in sensors which are supported by the Android platform, summarize the past wok on sensor-based authentication, and describe two major methods we adopt for this thesis work in Section 2. In Section 3, we present our proposed approach to authenticate user from various sensors, including sensor data collection, data preprocessing, modeling, and authentication. In Section 4, we present the performance of our work. The thesis is concluded by Section 5.

2 BACKGROUND

2.1 Sensors

Smart devices have built-in sensors that can measure motion, orientation, and various environmental conditions e.g. temperature, ambient light, etc. These sensors provide raw data with high accuracy and are useful to perform a wide variety of sensor-related tasks. For example, a

game application can track readings from a device's gravity sensor to infer complex users' gestures and motions, such as tilt, shake, rotation, swing [2]. Table 1 summarizes the sensors that are supported by Android platforms including descriptions and common usage. There are three broad categories of sensors: motion sensors which are used to measure acceleration forces and rotational forces including accelerometers, gravity sensors, gyroscopes, and rotational vector sensors; environmental sensors which includes barometers, photometers, and thermometers to measure various environmental parameters; and position sensors which is to measure the physical position of a device including orientation sensors and magnetometers.

Table 1: Sensor types supported by the Android platform [2].

Sensor	Description	Common Use
Accelerometer	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z)	Motion detection
Gravity	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, and z)	Motion detection
Gyroscope	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z)	Rotation detection
Rotation vector	Measures the orientation of a device by providing the three elements of the device's rotation vector	Motion detection and rotation detection
Light	Measures the ambient light level in lx	Controlling screen brightness
Pressure	Measures the ambient air pressure in hPa or mbar	Monitoring air pressure changes
Temperature	Measures the ambient room temperature in degree Celsius (°C)	Monitoring air temperatures
Orientation	Measures degree of rotation that a device makes around all three physical axes(x, y, and z)	Determining device position
Proximity	Measures the proximity of an object in cm relative to the view screen of a device	Phone position during a call

2.2 User Authentication based on Mobile Sensors

With the rapid development of mobile sensory technology, it is easy to collect data from many types of sensors in a smart device. These sensory data are extensive and personal. How to learn a user's behavior from the sensor data is a hot research area today.

For user authentication, the traditional approach relies on password. Also, biometric authentication is also popular to apply context-awareness to security applications [3, 4]. It verifies user identity based on the unique biological characteristics such as fingerprints, voice waves, retina and iris pattern, face recognition and etc. Except those, there are other biometric authentications including signatures and blinking pattern which are studied. But now, the sensor data we obtain can be a possible way to establish a user's portfolio and learn his/her behaviors. There is already significant amount of work performed on sensor-based authentication. Then, we summarize the latest approaches below.

In [5], they propose a multi-sensor based system to achieve continuous authentication on smart devices. Also, they employ Support Vector Machines (SVM) as their learning algorithm. Through a lot of experiments, they find that the combination of more sensors can provide better accuracy. They can achieve more than 90% accuracy in detecting the owners.

Also, in [6], they develop a lightweight, temporally and spatially aware user behavior model using light sensor, orientation, magnetometer, and accelerometer to authenticate users. However, it requires much more detecting time.

SenSec is a mobile system framework, which is proposed by [7]. They use the data which are collected from the accelerometer, gyroscope, and magnetometer to construct the gesture model of how a user uses the mobile device. They ask users to perform five tasks such as picking up the

device, and model a user' gesture patterns though a continuous n-gram language model. The results of using SenSec system show that it can achieve 75% accuracy in user classification and 71.3% accuracy in user authentication.

2.3 K-means Clustering Algorithm

Cluster analysis is the process of partitioning a set of data objects (or observations) into subsets [8]. Each subset is called a cluster, such that objects within a cluster are similar to one another, yet dissimilar to objects in other clusters. Cluster analysis has been widely used in many applications such as image pattern recognition and Web search. There are many clustering algorithms in the literature: partitioning methods, hierarchical methods, density-based methods, and grid-based methods [8]. Partitioning methods, which are distance-based and use mean to represent cluster center, are effective for small- to medium-size data sets. Based on the general characteristics of partitioning methods, we employ K-means clustering algorithm which is the most well-known and commonly used partitioning method to pre-process the raw sensor's data sets before they are used for authentication.

K-means clustering algorithm is a centroid-based technique [8]. Suppose there is a data set, D, contains n objects in Euclidean space. Partitioning methods distribute the objects in D into k clusters, $C_1, \dots, C_k$, that is, $C_i \subset D$ and $C_i \cap C_j = \emptyset$ for $(1 \leq i, j \leq k)$. This technique uses the centroid of a cluster, C_i, to represent cluster. The centroid can be mean or medoid of the objects assigned to the cluster. The difference between an object $p \in C_i$ and c_i, is measured by $dist(p, c_i)$, where $dist(p, c_i)$ is the Euclidean distance between two points x and y. The within-cluster variation is used to measure the quality of cluster C_i, which is the sum of $squared\ error$ between all objects in C_i and the centroid c_i, defined as

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)^2, \quad (2.1)$$

where E is the sum of the squared error for all objects in the data set; p is the point in space representing a given object; and c_i is the centroid of cluster C_i. The k-means algorithm uses the mean value of the points within a cluster as the centroid. It works as follows. First, it randomly

Algorithm: k-means. The k-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

k: The number of clusters,

D: A data set containing n objects.

Output: A set of k cluster.

Method:

(1) arbitrarily choose k objects from D as the initial cluster centers;
(2) repeat
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) Update the cluster means, namely, calculate the mean value of the objects for each cluster;
(5) **until** no change;

Figure 1: The procedure of k-means algorithm

chooses k of the objects in D and each of them represents initially a cluster mean. Based on the Euclidean distance between the object and the cluster mean, each of the remaining objects can be assigned to the cluster to which it is the most similar (or closest in Euclidean distance in this case). Then, the algorithm iteratively furthers the within-cluster variation. In each iteration, the new mean

is computed by each cluster from the previous iteration. Then, the remaining objects are reassigned using the new mean as the cluster centers. Until the assignment is stable or converges, the iterations stops. The *k*-means procedure is summarized in Figure 1.

2.4 Hidden Markov Model

In this thesis, we use the Hidden Markov Model algorithm for authentication. Hidden Markov Model (HMM) is a stochastic model which predicts the hidden state of a system given an observation at a certain moment [9, 10, and 11]. It consists of a collection of finite states connected by transitions. Compared to a Hidden Markov Model, the Markov Model is a collection of *visible* states to the observer. The parameter of a Markov model is the state transition probability. However, in an HMM, the states are not visible (thus hidden), rather are implicitly related to something which can be observed. With visible observations, the state can be stochastically inferred.

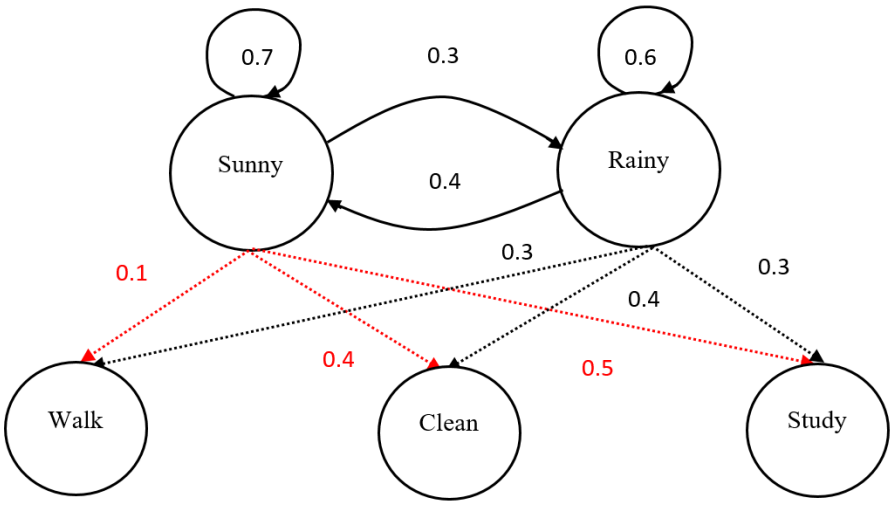


Figure 2: An example of the Hidden Markov Model

Assume the following scenario. There are two people, A and B, who live in different cities. Every day they talk to each other about what they did over the phone. A only has three activities

to do each day: walking on campus, cleaning his apartment, and studying. Each activity is determined by the weather on a given day. B has no information about the weather in A's city, but he believes that he can guess the weather based on A's activity. There are two different weathers, "Sunny" and "Rainy", but B cannot observe them directly. They are hidden from B, so they are considered as *hidden* states. "Walking", "cleaning", and "studying" are shown a certain chance which will happen on each day. These three activities are the *observations*, because A tells to B. The entire system is a Hidden Markov Model. It is given in Figure 2.

HMM has been widely used in speech and image-based gesture recognitions [9, 10]. An HMM can be defined by:

- S – A set of hidden states in the model, we denote the each state as $S = \{S_1, S_2, \dots, S_N\}$ and N is the number of hidden states;
- V – A set of distinct observation symbols in the model, we denote the each individual symbol as $V = \{V_1, V_2, \dots, V_M\}$ and M is the number of distinct observation symbols in each hidden state;
- A – The transition probability matrix, $A = \{a_{ij}\}$, where a_{ij} is the transition probability of taking the transition from state i to state j,

$$a_{ij} = P\left[S_j \middle| S_i\right], 1 \ll i, j \ll N; \tag{2.3}$$

- B – The output probability matrix, $B = \{b_j(V_k)\}$ for a discrete HMM, where V_k stands for a discrete observation symbol,

$$b_j(V_k) = P\left[V_k \; at \; t \middle| S_j\right], 1 \ll \; j \ll N, 1 \ll k \ll M; \tag{2.4}$$

- π – The initial state probabilities.

It can be seen that there are five elements for a complete HMM. For convenience, we write an HMM in a compact notation

$$\lambda = (A, B, \pi) \tag{2.5}$$

to represent a HMM with a collection of different hidden states and a sequence of observations [9].

In general, HMM can solve three common problems: evaluation, decoding, and learning problems [9]. The evaluation problem uses a forward-backward algorithm [12, 13] to evaluate the probability of efficient computation based on a given set of observations and a model. The learning problem optimizes parameters of a model to better describe the observations using the expectation-maximization (EM) algorithm [14]. The decoding problem uses the Viterbi algorithm [9, 15, and 16] to find the most likely state sequence given an observation sequences.

In this work, we use the expectation-maximization (EM) algorithm, which is primarily used to solve two types of problems. One is the maximum likelihood estimation (MLE) problem. The other is to address the problem of missing data values due to limitations of observation process. Baum-Welch (BW) algorithm [9] is the most often used EM algorithm. Using training data, the BW algorithm employs the forward-backward algorithm and computes MLE and posterior mode estimation for the probability parameters in an HMM. We employ the BW algorithm to learn finger gestures and handholding patterns.

Baum-Welch algorithm, introduced by [11] in 1970, has two steps: the E-step and the M-step. The E-step uses the forward-backward algorithm to determine the probability of various possible state sequences that generated the training data. The M-step uses the maximum likelihood algorithm (MLE) to re-estimate values for all of parameters.

Assume an HMM with N states, and set its parameters $\lambda = (A, B, \pi)$.

Given each s and t, it computes the probability of being in state $S_t = s$ at time t, and we use $P(S_t = s|o_1, o_2, \dots, o_T)$ to represent it; then, given each s, s' and t, it computes the probability of being in state $S_t = s$ at time t and state $S_{(t+1)} = s'$ at time t +1: $P\left(S_t = s, S_{(t+1)} = s' \middle| o_1, o_2, \dots, o_T\right)$.

The forward procedure can be stated as follows:

(1) Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), 1 \le i \le N \tag{2.6}$$

(2) Recursion:

$$\alpha_{i+1}(j) = b_j(O_{T+1}) \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right], 1 \le t \le T-1, 1 \le j \le N \tag{2.7}$$

(3) Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.8}$$

The backward procedure is very similar to the forward procedure. The only difference is that the recursion is backward.

(1) Initialization:

$$\beta_T(i) = 1, 1 \le i \le N \tag{2.9}$$

(2) Recursion:

$$\beta_i(t) = \sum_{j=1}^{N} b_j(O_{t+1}) a_{ij} \beta_{i+1}(j), 1 \le j \le N, 1 \le t < T \tag{2.10}$$

(3) Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \beta_T(i) \tag{2.11}$$

After computing α and β, we define $\xi_t(i,j)$ to describe the procedure for re-estimation of HMM parameters, which is the probability of being in state $q_t = S_i$ at time t and state $q_{t+1} = S_j$ at time t+1 respectively given the observed sequence O and parameters λ.

$$\begin{aligned}\xi_t(i,j) &= \mathrm{P}\left(q_t = S_i, q_{t+1} = S_j \middle| \mathrm{O}, \lambda\right) \\ &= \frac{\alpha_t(i)\beta_{t+1}(j)a_{ij}b_j(O_{t+1})}{P(O|\lambda)} \\ &= \frac{\alpha_t(i)\beta_{t+1}(j)a_{ij}b_j(O_{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\beta_{t+1}(j)a_{ij}b_j(O_{t+1})}\end{aligned} \tag{2.12}$$

Also, we denotes $\gamma_i(t)$ as the probability of being in state $s = S_i$ at time t, given the observation sequence and the model.

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{2.13}$$

After we get ξ and γ, λ can be updated:

$$\pi^* = \gamma_t(i) \tag{2.14}$$

$$\begin{aligned}a_{ij}^* &= \frac{expected\ number\ of\ transitions\ from\ state\ S_i\ to\ S_j}{expected\ number\ of\ transitions\ from\ state\ S_i} \\ &= \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}\end{aligned} \tag{2.15}$$

$$b_i^*(v_k) = \frac{expected\ number\ of\ times\ in\ state\ S_j\ and\ observing\ symbol\ v_k}{expected\ number\ of\ times\ in\ state\ S_j}$$

$$= \frac{\sum_{t=1}^{T} 1_{y_t=v_k} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{2.16}$$

where $1_{y_t=v_k} = \begin{cases} 1, if\ y_t = v_k \\ 0, otherwise \end{cases}$.

These steps are now repeated iteratively until a desired level of convergence. We present the pseudo code of Baum-Welch algorithm in Figure 3 [11].

Assume a HMM with N states.

Randomly set its parameters $\lambda = (A, B, \pi)$.

Until converge (i.e. λ no longer changes) do:

E Step:

Compute values for $\gamma_t(i)$ and $\xi_t(i,j)$ using current values for parameters A and B.

M Step:

Re-estimate parameters:

$a_{ij} = \hat{a}_{ij}$

$b_j(v_k) = \hat{b}_j(v_k)$

End.

Figure 3: Pseudo code of BW algorithm

3 SYSTEM DESIGN

In this section, we discuss our proposed solution to the mobile user authentication based on various sensors on mobile devices.

The proposed solution consists of four components. First, we collect the data from the selected sensors. Then, we use the K-means clustering algorithm and the normal distribution to preprocess the raw sensor data. After that, we use Hidden Markov Models (HMMs) to learn a user's profile based on the finger gestures and handholding patterns. Finally, we employ the Least Mean Square (LMS) algorithm to adaptively combine the outputs of multi-sensors' HMMs to authenticate the user of the device. Figure 4 shows a high-level design architecture of our proposed authentication solution.

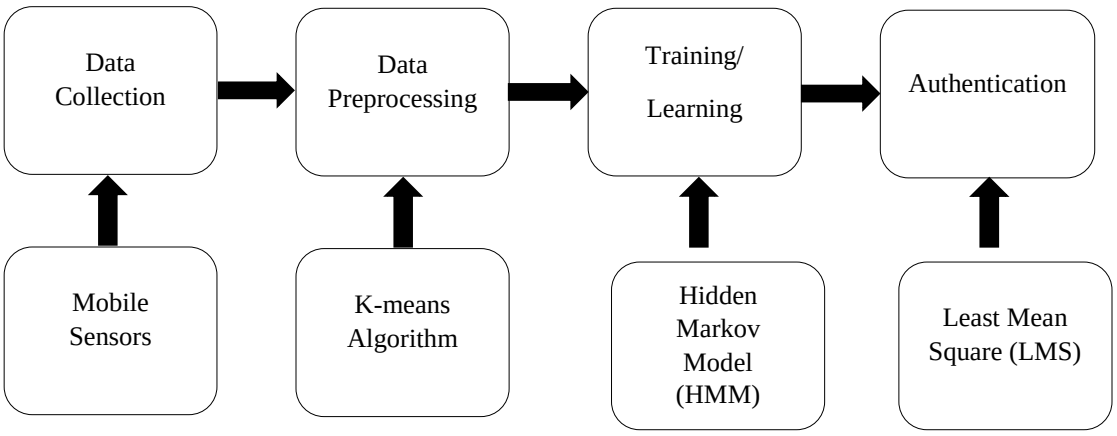


Figure 4: Block diagram of our user authentication system

3.1 Sensor Selection

As shown in Table 1, there are many built-in sensors in a smart device today. In this thesis work, the foundational hypothesis is that each user has a unique set of combinational features of finger gestures and handholding patterns in using a smart device. For example, a user Bob normally uses his right thumb resulting in 0.38 cm touch size on the screen to unlock his Google Nexus 5X and picks his phone from the table at a certain speed.

In our work, we experiment with three sensors that are available in a Google Nexus 5: accelerometer, orientation sensor, and touch screen input. First, we characterize finger gestures

with two features: finger touch size and finger touch position. Finger touch size and touch position on screen both can be obtained through touch screen input measurements. Second, users tend to pick up and hold devices in various patterns, which can be learned from various motion detecting sensors such as accelerometer and orientation sensors. These sensors can report users' information about their behaviors. The accelerometer measures coarse-grained motion of a user like how he walks (multi-sensor). The orientation sensor can detect fine-grained motion of a user, e.g. how he holds a smartphone (multi-sensor). We will focus on two factors of holding pattern: speed of device motion and device orientation during moving. Device motion reflects the pattern that a user picks up a smart device to use. The accelerometer and orientation sensors can provide the data that can characterize the motion. Furthermore, these sensor data can be easily obtained on Android devices today.

3.2 Data Preprocessing

We select three sensors which are the accelerometer, orientation sensor, and touch screen input measurements to collect data. These obtained data construct a big data set, but they are not proper to directly feed into the system model. Due to the difference of data formats, we use different methods to pre-process them before the data is mined through a system model. Figure 5 shows two data preprocessing methods we use.

For the accelerometer and orientation sensors, we attempt to identify handholding patterns based on the collected data. Hence, we use Hidden Markov Model to train and learn these data. HMM algorithm requires an observation sequence. However, the sensor data of accelerometer and orientation are continuous and is hard to be used as observations of HMM model directly. So, the k-means algorithm is employed first to cluster these sensor data to identify a limited number of clusters' indices that are then used actually as observation sequences for the HMM. We start with

a random number of clusters. After several times iterations, the training set is complete and we obtain the centroids of each cluster and each cluster which contains different objects.

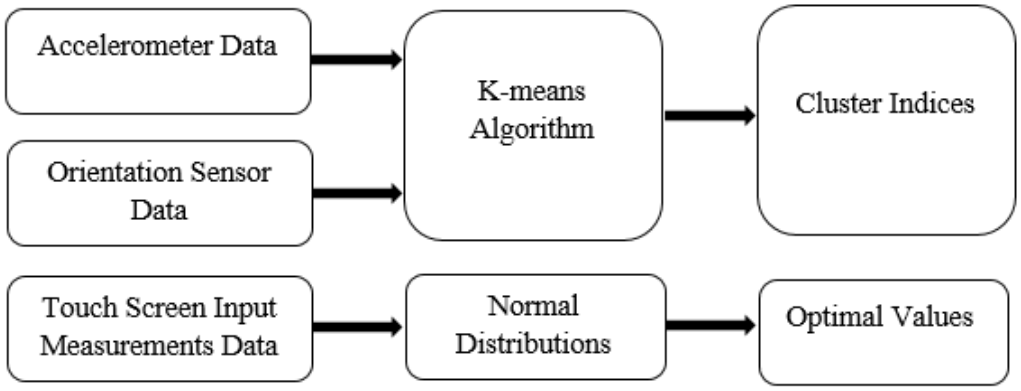


Figure 5: Data preprocessing methods

For the touch screen measurements, There are only one value for the touch size and two values which are x-axis and y-axis for touch position on the screen. So, the k-mean algorithm does not work for the data. We rather choose another method to analyze them. We use Matlab, which is a multi-paradigm numerical computing environment, to plot the data based on the means and standard deviations of the two data sets we collected and find that the shapes are bell-shaped, which are very alike to the curves of the normal distribution. Therefore, we formalize the touch sensor data with normal distributions.

In probability theory, the normal (or Gaussian) distribution is very common continuous probability distribution. The probability density of the normal distribution is:

$$f(x|u,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.1}$$

Here, μ is the mean of the distribution. The parameter σ is its standard deviation with the variance σ^2. About 68% of values drawn from a normal distribution are within one standard

deviation σ away from the mean; while $\mu \pm 2\sigma$ accounts for 95.45% of values; and $\mu \pm 3\sigma$ accounts for 99.73% of values. Because we want to narrow down the data set, we choose the $\mu \pm \sigma$ as our range.

With the normal distribution normalization, the optimal values for each touch sensor data set are identified.

3.3 Hidden Markov Model for Behavior Recognition

The primary algorithm used in this thesis is a Hidden Markov Model. HMM consists of four key components: hidden states, observations, the state transition matrix, and the confusion matrix. So, in our model, it is defined as shown in Figure 6 [9].

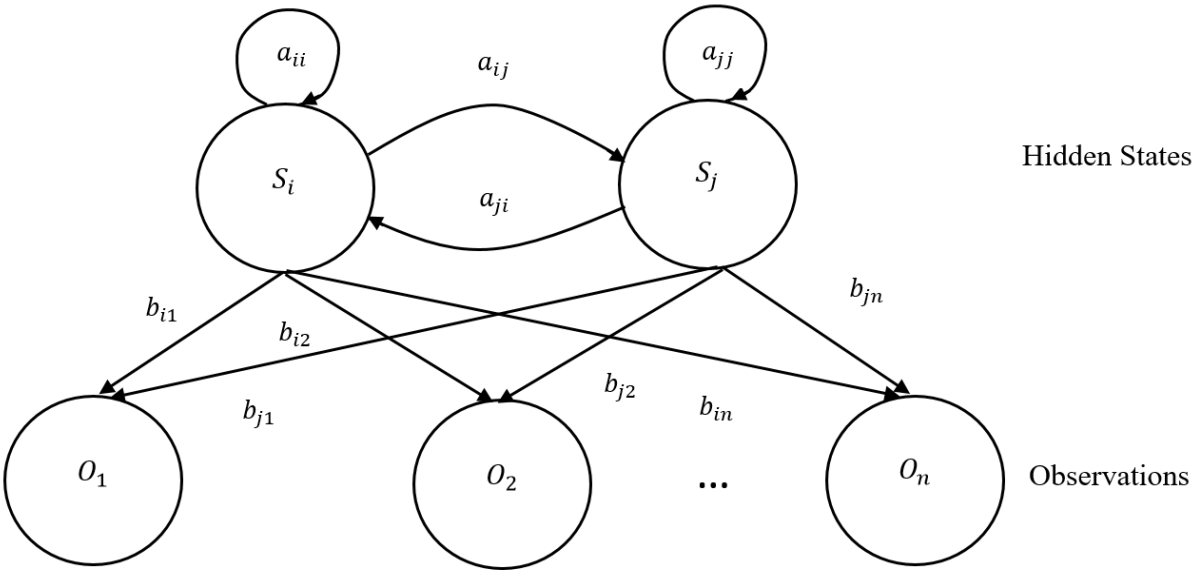


Figure 6: HMM architecture of hidden states, transitions, and observations.

In our model, the hidden states S_i and S_j represent the possible gesture patterns, the possible observations $O_1, O_2, \ldots, O_n$ are the pre-processed data of accelerometer and orientation sensors with their message values. In the data pre-processing as discussed above, we have already mined

the data collected by those two sensors and formulated six clusters using k-means algorithm. So, the resulting HMM has six possible observations in temporal sequence. The transition matrix consists of a_{ii}, a_{ij}, a_{ji}, and a_{jj}. The transition probabilities represent the probability of taking the transition from state S_i to state S_j. Also, it can occur from a state back to itself. The emission probabilities which are $b_{i1}, b_{i2}, b_{in}, b_{j1}, b_{j2},$ and b_{jn} represent the likelihood that a given cluster happens at state S_i or S_j. These probability parameters are obtained through a training process with the Baum-Welch algorithm on a training set of data.

Each type of sensor data is modeled with an individual HMM. Therefore, we combine these two HMMs into a combinational model to as a gesture pattern recognizer as in Figure 7. In order to do gesture recognition, we must perform the following steps:

- For each sensor, we must build an HMM λ^s. As we describe previously, we should get the model parameters (A, B, π) after training.
- For each unknown user which is to be recognized, the processing of a recognized should follow Figure 7. We have the observation sequence $O = \{O_1, O_2\}$, which represent the accelerometer and the orientation sensor respectively using k-means algorithm. Then, followed by calculation of model likelihoods for both two sensors, $P(O|\lambda^s)$.

The probability computation is generally performed by the Baum-Welch algorithm, which we describe before.

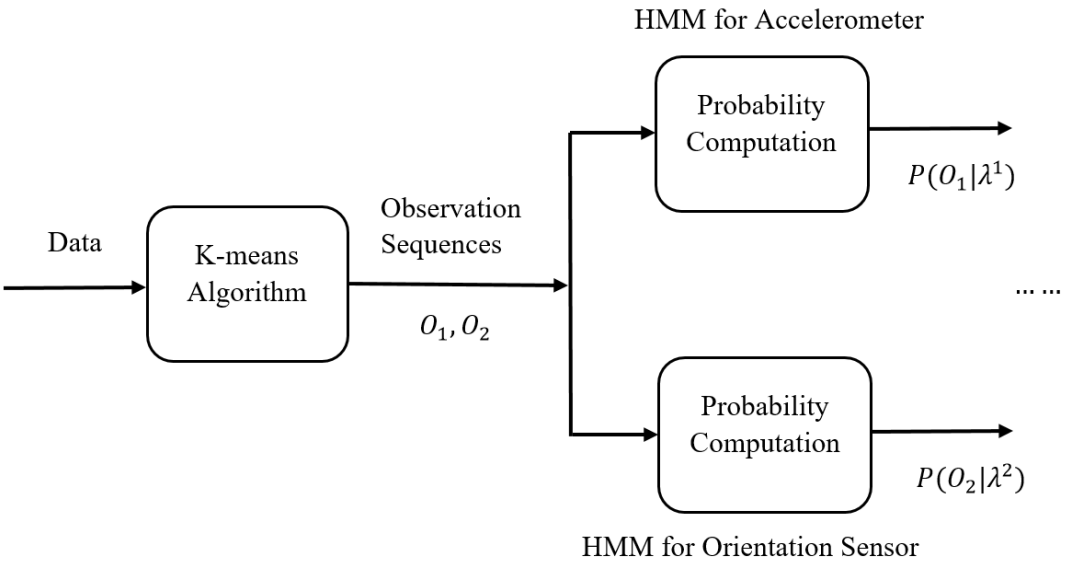


Figure 7: Block diagram of a gesture pattern HMM recognizer

3.4 Least Mean Squares for Authentication

With multiple sensors, the user authentication problem can be formulated as linear regression problem. We use the Least Mean Squares update rule (LMS) [17, 18] for evaluating the parameters of a linear regression function that predicts the result to achieve user authentication. Figure 8 shows the steps in this learning process.

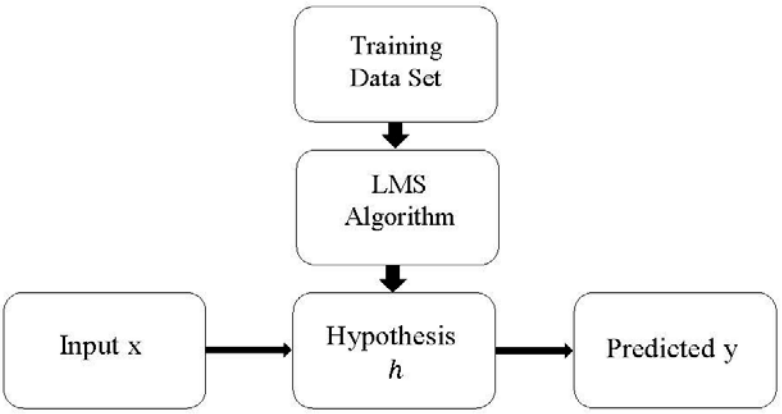


Figure 8: The process of Least Mean Square update rule

As mentioned in previous sections, we have four attributes or features: touch size, touch position, pickup speed (accelerometer) and device orientation. Hence, the x's are four-dimensional vectors in $\mathbb{R}^4$, with $x_1^{(i)}$ being the probability of the i-th data in the accelerometer's training set, $x_2^{(i)}$ being the probability of the i-th data in the orientation sensor's training set, $x_3^{(i)}$ being the i-th user's touch size on the screen of device, and $x_4^{(i)}$ being the i-th user's touch position on the device's screen.

We denote h as a hypothesis linear function of x:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 \tag{3.2}$$

Here, the θ_i's are the parameters (also called weights) parameterizing the space of linear functions mapping from the input $\mathcal{X}$ to the output $\mathcal{Y}$. To simplify our notation, we will introduce the convention of $x_0 = 1$, so that

$$h_\theta(x) = \theta^T x = \sum_{i=0}^{n} \theta_i x_i \tag{3.3}$$

where n is the number of input variables (not counting x_0).

In order to make the hypothesis function $h(x)$ close to y, the cost function in the multivariate case can be written in the following form:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_0\left(x^{(i)}\right) - y^{(i)} \right)^2 \tag{3.4}$$

The regression goal is to minimize cost $J(\theta)$. One way to do it is to use the batch gradient descent algorithm. In batch gradient descent update rule, each iteration performs the update:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_0\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)} \left(\text{simultaneously update } \theta_j \text{ for all } j\right) \quad (3.5)$$

Here, α is the learning rate. With each step of gradient descent, the parameters θ comes close to optimal values that will achieve the lowest cost $J(\theta)$. The gradient descent will run until reaching the convergence to find the final values of θ.

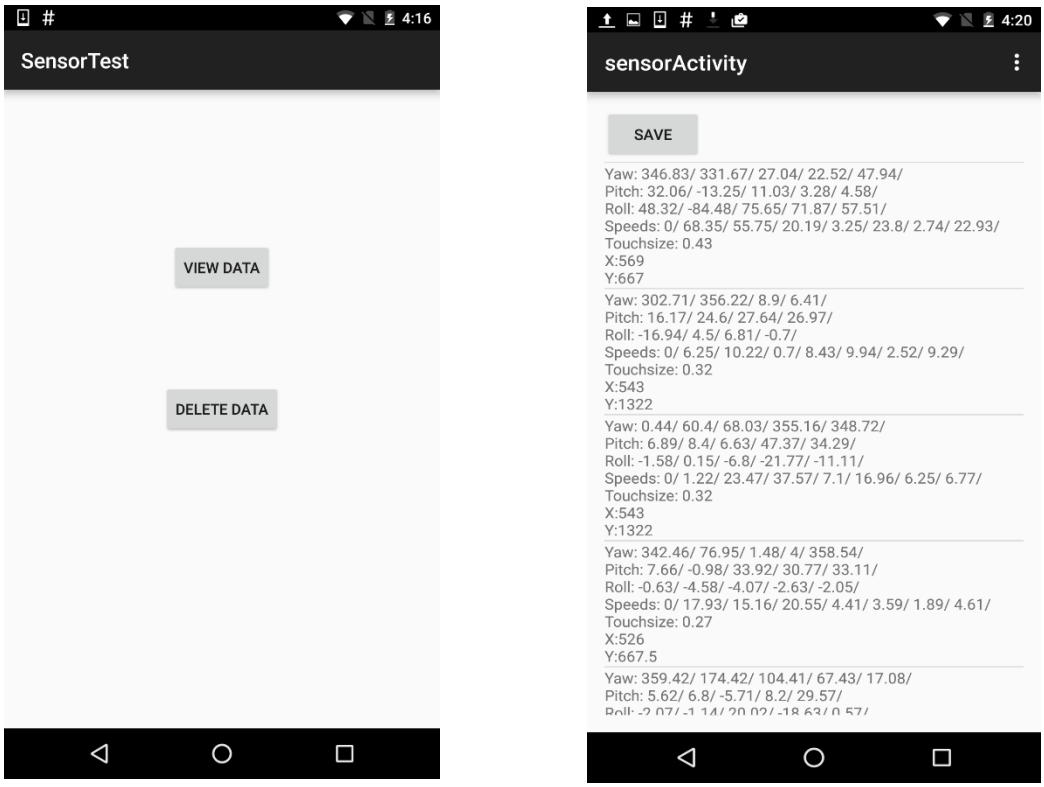
4 EXPERIMENTS

We implement the system and evaluate it with data from real life experiments. We use Google Nexus 5X with Android 6 as our experimental device. Two data sets are collected from the device owner and a non-owner. The sensor data are taken from accelerometer, orientation sensor, and touch screen input measurements. The k-means algorithm pre-processes accelerometer and orientation sensors' data. We use HMM to train and learn the data to obtain a user's portfolio of handholding patterns and finger gestures. Then LMS is employed to combine all the outputs from the HMM models to achieve the final result of authentication. In our experiments, we use the 90% of the data for training and the rest 10% data for testing.

4.1 Data Collection and Preprocessing

We have two data sets which were collected by ourselves. Both data sets are collected from two graduate students working on the project in 2016. The smart device we used is an Android device Google Nexus 5X. Each data set includes the data from the accelerometer, orientation sensor, touch size, and touch position. The activity we performed is picking up the device from the table and unlocking it. Each student performed this activity for hundreds of times. Each data collection takes 2 seconds.

We implement an Android App to collect users' data. Every time, user just need to pick up the device and unlock it by sliding the screen. The sensors will record data during the motion of the device. Figure 9 (a) shows the user interface of the APP. There are two buttons: "VIEW DATA" button and "DELETE DATA" button. When click "VIEW DATA" button, user can enter to the second page which shows the collected data in Figure 9 (b). Also, we use SQLite database which is embedded into every Android device to store users' data.



(a) (b)

Figure 9 (a) (b): Our Android App

The accelerometer data has eight attributes. That means there are 8 sets of accelerometer data collected every 2 seconds. It always starts from 0. The orientation sensor measures degree of rotation that includes three physical axes: yaw, roll, and pitch. The data from the orientation sensor consists of 5 values, which means there are 5 sets of orientation data collected every 2 seconds. Table 2 shows the data we collected by an Android APP.

Table 2: Raw data we collected

Yaw	346.83/331.67/27.04/22.52/47.94
Roll	48.32/-84.48/75.65/71.87/57.51
Pitch	32.06/-13.25/11.03/3.28/4.58
Accelerometer	0/68.35/55.75/20.19/3.25/23.8/2.74/22.93
Touch Size	0.43
Position X	569
Position Y	667

We create a plot using the data of orientation sensor in 3D as shown in Figure 10. The touch screen input measurements include two parts that are touch size and touch position. When a user touches the screen and the sensor gives one value of the size and two values of the position which are X-axis and Y-axis on the touch screen.

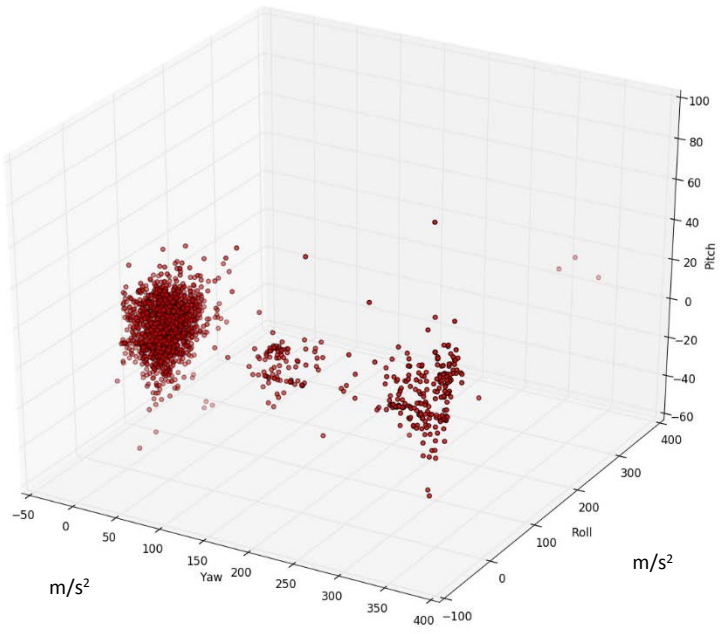


Figure 10: The plot using orientation sensor in 3D

Based on the requirements from Hidden Markov Model, we need to obtain an observation sequence. Therefore, we use the k-means algorithm provided by Weka [24] to find clusters' indices that are used as the observation sequence for the HMM modeling the prediction of user gestures in our solution. For the accelerometer, we have 8 values every 2 seconds. The detail training data is collected into a large file. The k-means algorithm was performed on this data set with the number of clusters set as 6. After 72 iterations, the training is completed. We obtained the centroids of each cluster and each cluster contains different objects. Figure 11 shows the result of clustering using accelerometer data. There are six different colors represent six different clusters.

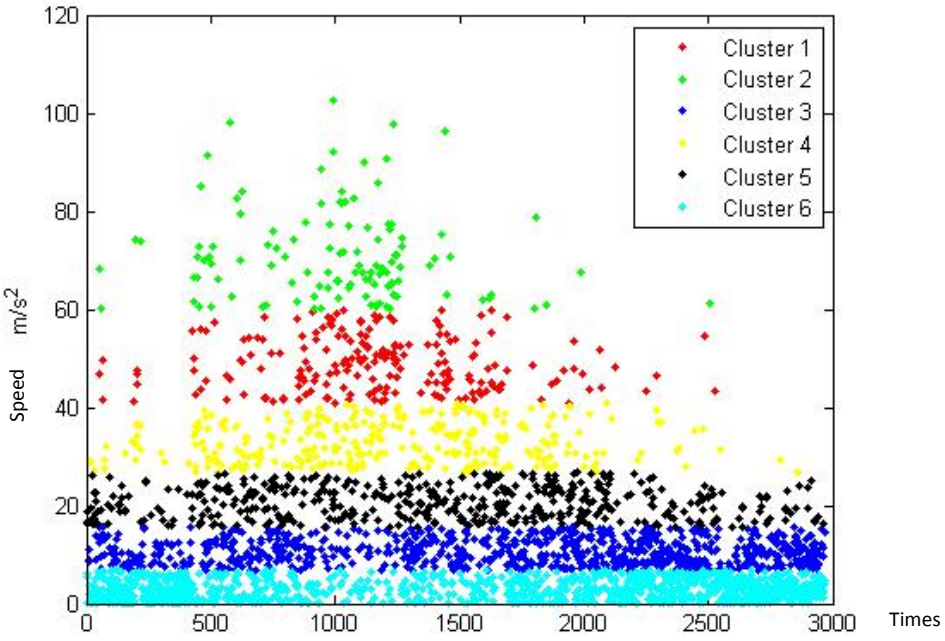


Figure 11: The result of clustering using accelerometer data

The training process of the orientation sensor is the same as the accelerometer. Because the orientation sensor includes three different attributes: yaw, roll, and pitch. First, we perform clustering to identify their own cluster centroids for the attributes as shown in Figure 12 (a) (b) (c).

Each physical axes has 6, 4, and 4 clusters respectively. Then, we compile all clusters' indices into one file. Hence, we generate a new matrix which has three indices number from three attributes. The clustering result is shown in Figure 12(d) below.

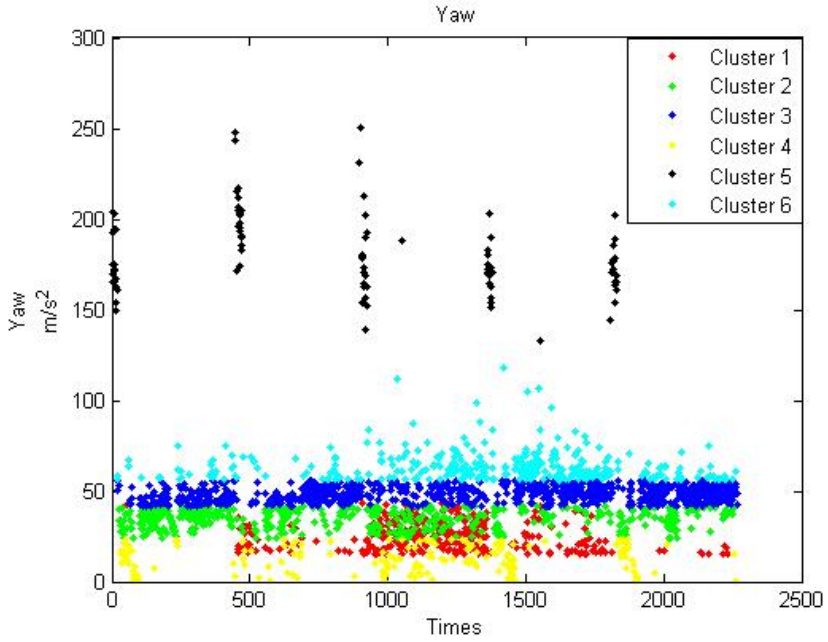


Figure 12 (a): The clustering result of yaw data

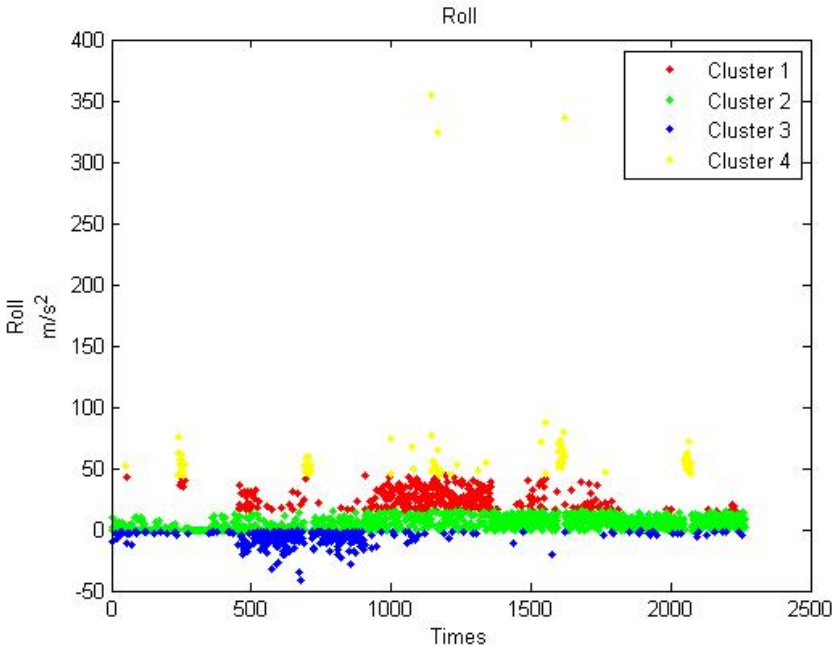


Figure 12 (b): The clustering result of roll data

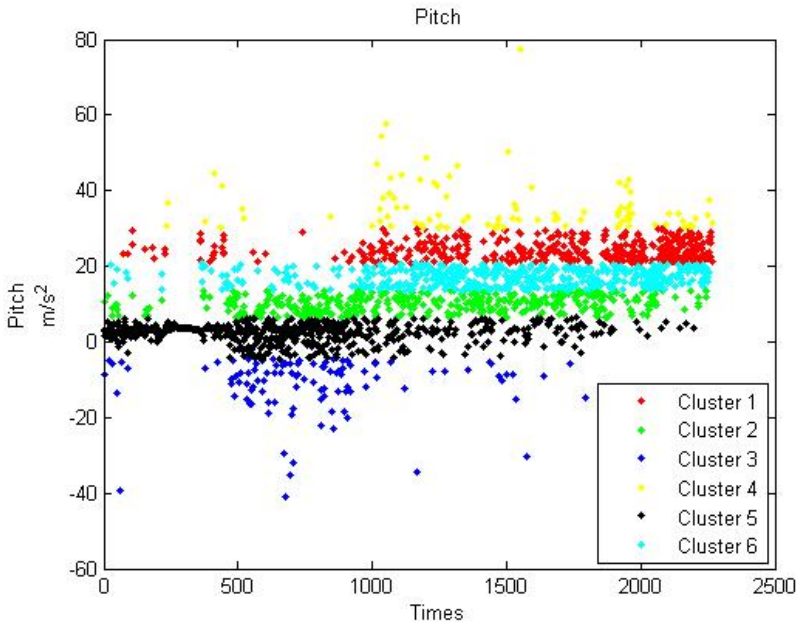


Figure 12 (c): The clustering result of pitch data

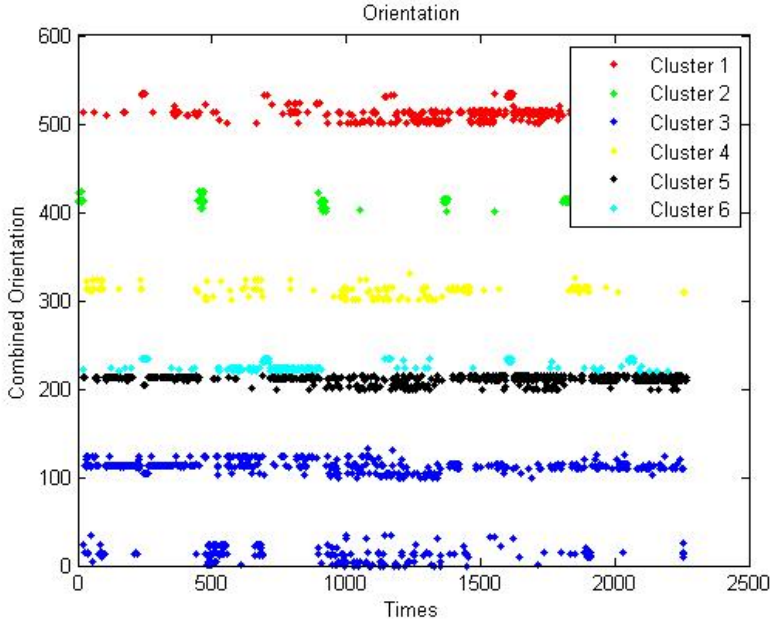


Figure 12 (d): The clustering result of combined orientation data

Table 3 shows an example of the training results of one data set. It shows that the data type, the number of clusters, the number of iterations, the sum of squared error (E), and the training time for the accelerometer sensor, orientation sensor including three physical axes.

Table 3: The training results for each sensors

Sensors	Data Type	Number of Clusters	Number of Iterations	Sum of Squared Error (E)	Training Time (Seconds)
Accelerometer	Numeric	6	62	3.29	0.09
Yaw	Numeric	6	57	1.72	0.05
Roll	Numeric	4	22	1.96	0.03
Pitch	Numeric	6	18	1.47	0.03
Combined Orientation	Numeric	6	4	5.63	0.01

For the touch screen size and the touch screen position, the optimal values are identified through the following procedure. Figures 13 and 14 (a) (b) respectively show the combinations of the histogram of frequency (distribution) of the owner's touch size data and touch screen position and the corresponding normal distribution curve that describes these data. Then, Figure 15 shows two different users' normal distribution curves that describe the touch measurement data. Because 68% of values drawn from a normal distribution are within one standard deviation σ away from the mean, we choose this deviation $\mu \pm \sigma$ to find the false alarm and fallacy. False Alarm is defined as that a value is categorized as true, but actually not from the owner. Fallacy defines that a value is determined as false, but actually true from the non-owner user. The linear regression line of false alarm and fallacy are shown in Figure 16. We can easily find that these two lines have a common point. This common point represents the optimal value 0.242 cm^2 for touch size. Following the same analysis, we can identify the optimal values for both attributes.

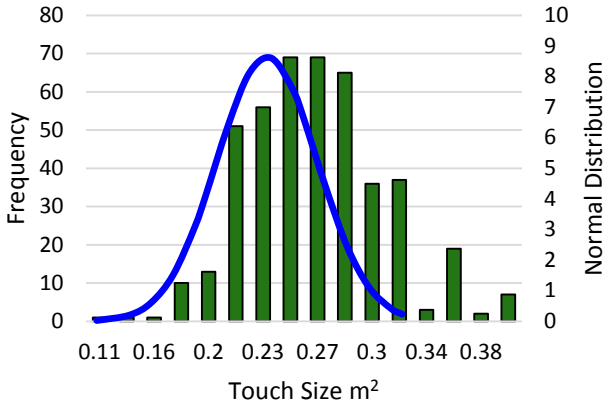


Figure 13: The histogram and normal distribution curve of touch screen size

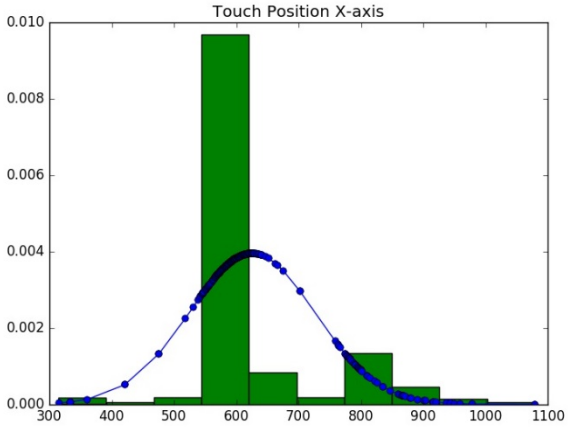


Figure 14 (a): The normal distribution of touch position x-axis

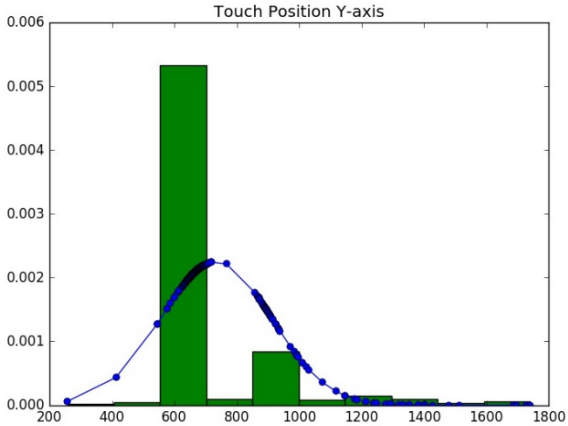


Figure 14 (b): The normal distribution of touch position y-axis

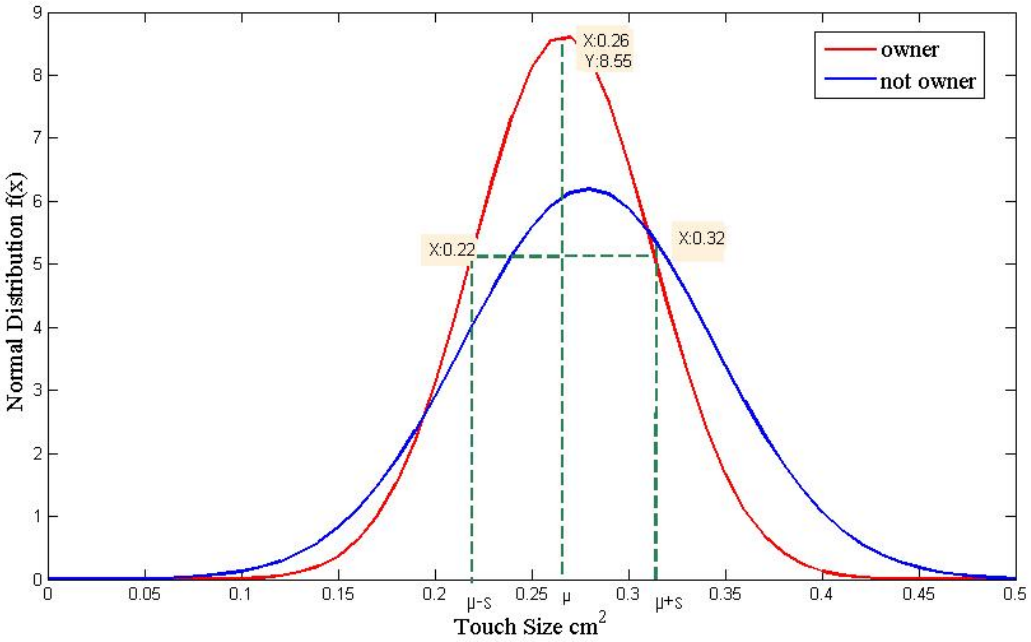


Figure 15: The normal distribution curves of owner and not owner's touch size

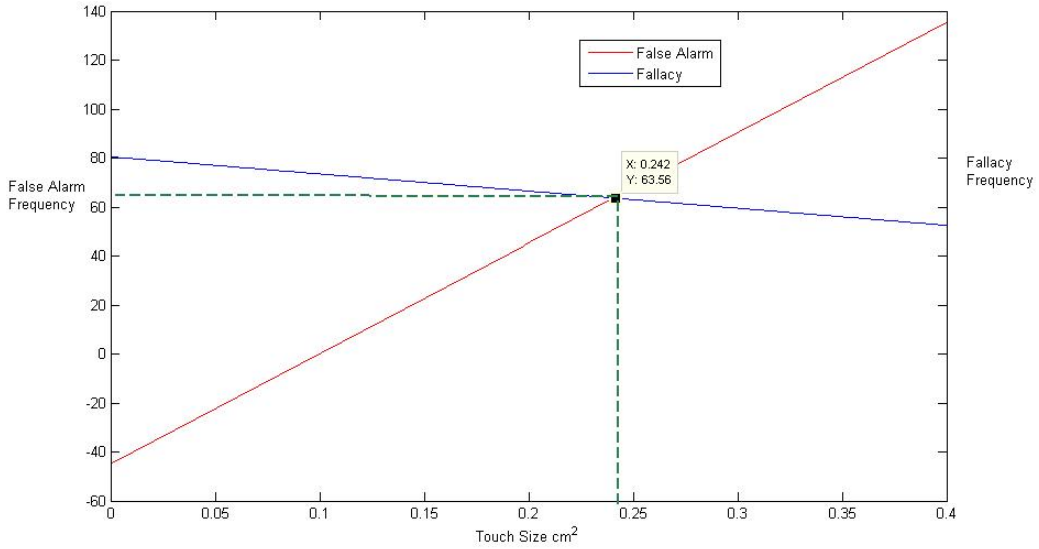


Figure 16: The optimal value for touch size

4.2 Training and Learning

HMM model is used for user gesture recognition and classification. We implement the algorithm in Java. The accelerometer and orientation sensor data are clustered into six classes using k-means algorithm. We use the indices of clusters of each sensor data to construct the observation sequences to train the HMM models to compute the probability matrices. Table 4 shows the initial probabilities for each state which are computed based on the training data. Table 5 shows the transition probabilities for the HMM of accelerometer. Table 6 shows the output probability matrix for the accelerometer HMM. Table 7 shows that the most common patterns of the accelerometer. There are 11 patterns describing the speeds of device motion. Among them, there are 5 patterns whose percentage are more than 10 percent. Table 8 shows the common patterns of orientation sensor. Like that in the accelerometer case, there are also 11 patterns which represent the speeds of device motion, of which, 5 patterns have the percentages over 10 percent. For the orientation sensor, there are 15 patterns which present the device orientation during moving. Only 4 patterns have the percentages over 10 percent.

Table 4: The initial probabilities for each state

	S_0	S_1	S_2	S_3	S_4	S_5	S_6
π_i	0.08498	0.09152	0.08498	0.04576	0.04576	0.60125	0.04575

Table 5: The transition probability matrix for the HMM of accelerometer

A_{ij}	S_0	S_1	S_2	S_3	S_4	S_5	S_6
S_0	0.199	0.207	0.199	0.103	0.103	0.084	0.103
S_1	0.218	0.193	0.218	0.097	0.097	0.081	0.097
S_2	0.199	0.207	0.199	0.103	0.103	0.084	0.103
S_3	0.218	0.193	0.218	0.097	0.097	0.081	0.097
S_4	0.218	0.193	0.218	0.097	0.097	0.081	0.970
S_5	0.211	0.180	0.211	0.090	0.090	0.129	0.090
S_6	0.218	0.193	0.218	0.097	0.097	0.081	0.097

Table 6: The output probability matrix for the HMM

Observations / States	O_0	O_1	O_2	O_3	O_4	O_5
S_0	0.088	0.017	0.277	0.163	0.172	0.283
S_1	0.072	0.072	0.291	0.081	0.176	0.308
S_2	0.088	0.017	0.277	0.163	0.172	0.283
S_3	0.072	0.072	0.291	0.081	0.176	0.308
S_4	0.072	0.072	0.291	0.081	0.176	0.308
S_5	0.007	0.001	0.112	0.016	0.059	0.806
S_6	0.072	0.072	0.291	0.081	0.176	0.308

Table 7: The percentages of common patterns for the accelerometer

Patterns	Percentage
Pattern 1	25%
Pattern 2	11%
Pattern 3	11%
Pattern 4	10%
Pattern 5	17%

Table 8: The percentages of common patterns for the orientation sensor

Patterns	Percentage
Pattern 1	10%
Pattern 2	13%
Pattern 3	15%
Pattern 4	23%

With extensive training in HMM models, Table 9 shows the results obtained, which construct an intermediate dataset, which is then used for the final authentication purpose.

Table 9: The new data set after training

Accelerometer	Orientation Sensor	Touch Size	Touch Position	Authentication Results
0.0016194006	0.0011564414	0.24	799	1
0.0095690141	0.0000396009	0.28	867	1
0.0045214809	0.0000610466	0.30	853	1
0.0001651595	0.0012976529	0.34	1090	0
…				

4.3 Prediction and Authentication

With the intermediate dataset in Table 9, a liner regression is applied to achieve the prediction and authentication of a user, as discussed in Section 3.4. The input vector X of the linear regression consists of *accelerometer*, *orientation*, *touch size* and *touch position* in Table 6. Gradient descent algorithm is performed to learn $\theta's$ and we obtain all weights $\theta_0 = 0.31457939,\ \theta_1 = 0.09017899,\ \theta_2 = 0.2481515,\ \theta_3 = 0.108766,\ \theta_4 = -0.1517374$. When we input the testing data into the function, we can get the final authentication result.

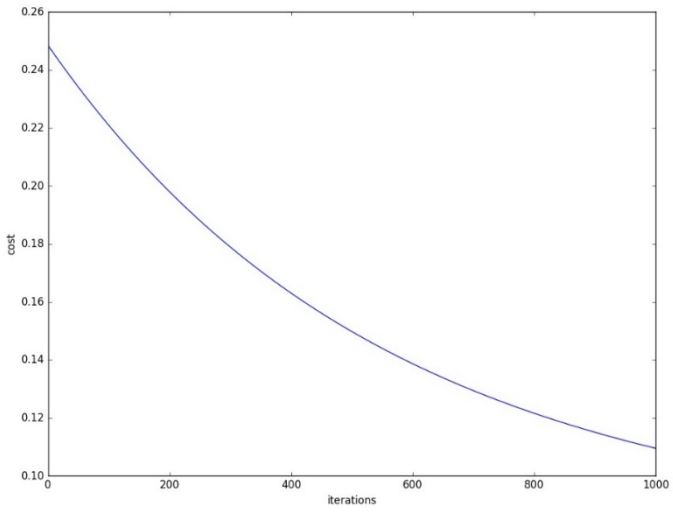


Figure 17: The convergence decreases as the number of iterations grow

The convergence figure of the gradient descent is plotted in Figure 17, which shows that the convergence decreases as the number of iterations grows.

4.4 Results

We repeated above training experiments for 3 times. By using the trained data, we can achieve a reasonable accuracy of 89% in average. Table 10 shows the accuracy after testing two different data sets.

Table 10: Accuracies after testing two different data sets

Data Sets	Accuracy
Set One	0.92
Set Two	0.86

5 CONCLUSION

This thesis proposes an innovative mobile device user authentication based on modeling, learning and recognizing user device handholding patterns and finger gestures using machine learning algorithms of Hidden Markov Model, K-mean clustering and Least Mean Square. By developing an Android App on Google Nexus 5X, we collect two data sets from users to evaluate the system performance. The training process, which preprocesses the raw sensor data with the k-mean clustering algorithm, takes a short time to generate excellent performance for clustering. An HMM algorithm is implemented in Java to learn and recognize a user's gestures based on the preprocessed sensor data. With training with HMM, a common set of patterns are learned for each user. The trained HMM models are combined by Least Mean Square linear regression for the final authentication. The experimental results show that the system can achieve 89% accuracy. The accuracy can be improved with more sensor data taken for learning and training while a device is used for longer and longer.

Considering our future work, we expect higher accuracy by using our mode. To improve the accuracy, we can increase the input dataset size and it is necessary to use more sensor data to learn user gestures. We will invite more users to participate in our experiments. It is necessary to explore more activities during collecting data, not just picking up a device. Through different kinds of activities, we can learn the users' gesture better. Also, based on our research we know that sensors can represent a user's characteristic behavior and physical environment. So, how to decrease the risks and security concerns is an interesting and potential research area.

6 ACKNOWLEDGEMENT

I would like to pay special thankfulness, warmth, and appreciation to my advisor Dr. Shaoen Wu who made my research successful and assisted me at every point to achieve my goal. Also, I am grateful to Rania Alkhazaali, Cory Dewitt, and other classmates for their patience and help.

REFERENCES

[1] "Smartphone Users Worldwide 2014-2019", *Statista*, 2015. [Online]. Available: http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed: 05- Apr- 2016].

[2] "Sensors Overview | Android Developers", Developer.android.com, 2016. [Online]. Available: http://developer.android.com/guide/topics/sensors/sensors_overview.html. [Accessed: 09- Apr- 2016].

[3] M. Baldauf and S. Dustdar. "A survey on context-aware systems", *International Journal of A-d Hoc Ubiquitous Computing*, p.2004, 2004.

[4] K. Wrona and L. Gomez. "Context-aware security and secure context-awareness in ubiquitous computing environments", 2005.

[5] W. Lee and R. Lee, "Multi-sensor authentication to improve smartphone security", in *International Conference on Information Systems Security and Privacy*, Feb. 2015.

[6] H. Kayacik, M. Just, L. Baillie, D. Aspinall and N. Micallef, "Data Driven Authentication: On the Effectiveness of User Behavior Modelling with Mobile Device Sensors", in *Proceeding of the Third Workshop on Mobile Security Technologies (MoST)*, 2014.

[7] J. Zhu, P. Wu, X. Wang and J. Zhang, "SenSec: Mobile security through passive sensing", in *International Conference on Computing, Networking and Communications (ICNC)*, 2013, pp. 1128-1133.

[8] J. Han, J. Pei and M. Kamber, *Data mining*. Amsterdam: Elsevier, 2006.

[9] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, 1989.

[10] J. Yang and Y. Xu, "Hidden Markov Model for Gesture Recognition", Carmegie Mellon University, Pittsburgh, 1994.

[11] K. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012, pp. 606-622.

[12] L. E. Baum and J. A. Egon, "An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology," *Bull. Amer. Meteorol. Soc*., vol. 73, pp. 360-363, 1967.

[13] L. E. Baum and G. R. Sell, "Growth functions for transformations on manifolds," *Pac. J. Math*., vol. 27, no. 2, pp. 211-227, 1968.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stac. Soc.*, vol. 39, no. 1, pp.1-38, 1977.

[15] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, Mar. 1973.

[16] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," IEEE Trans. *Informat. Theory*, vol. IT-13, pp. 260-269, Apr. 1967.

[17] A. Ng, "Machine Learning", Stanford University, 2016.

[18] M. Caraciolo, "Machine Learning with Python - Linear Regression", *Artificial Intelligence in Motion*, 2011.

[19] N. Ravi, N. Dandekar, P. Mysore and M. Littman, "Activity Recognition from Accelerometer Data", *IAAI-05*, pp. 1541-1546, 2005.

[20] A. Crandall and D. Cook, "Using a Hidden Markov Model for Resident Identification", in *the 2010 Sixth International Conference on Intelligent Environments*, 2010, pp. 74-79.

[21] Y. Lee and S. Cho, "Activity recognition using hierarchical hidden markov models on a smartphone with 3D accelerometer", in *Proceedings of the 6th International Conference on Hybrid Artificial Intelligent Systems*, 2011, pp. 460-467.

[22] S. Buthpitiya, Y. Zhang, A. Dey and M. Griss, "n-gram geo-trace modeling", in *Proceeding of the 9th international conference on Pervasive computing*, 2011, pp. 97-114.

[23] L. Song and S. Wu, “Cross-layer Wireless Information Security”, *The 23rd International Conference on Computer Communications and Networks (ICCCN)*, Shanghai, China, Aug. 2014.

[24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. Witten, *Waikato Environment for Knowledge Analysis (Weka)*. University of Waikato, 2009.