11-29-2016 12:00 AM

Leveraging Smartphone Sensor Data for Human Activity Recognition

Xizhe Yin, *The University of Western Ontario*

Supervisor: Weiming Shen, *The University of Western Ontario*
Joint Supervisor: Xianbin Wang, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering
© Xizhe Yin 2016

Recommended Citation

Yin, Xizhe, "Leveraging Smartphone Sensor Data for Human Activity Recognition" (2016). *Electronic Thesis and Dissertation Repository*. 4292.
https://ir.lib.uwo.ca/etd/4292

Abstract

Using smartphones for human activity recognition (HAR) has a wide range of applications including healthcare, daily fitness recording, and anomalous situations alerting. This study focuses on human activity recognition based on smartphone embedded sensors. The proposed human activity recognition system recognizes activities including walking, running, sitting, going upstairs, and going downstairs. Embedded sensors (a tri-axial accelerometer and a gyroscope sensor) are employed for motion data collection. Both time-domain and frequency-domain features are extracted and analyzed. Our experiment results show that time-domain features are good enough to recognize basic human activities. The system is implemented in an Android smartphone platform.

While the focus has been on human activity recognition systems based on a supervised learning approach, an incremental clustering algorithm is investigated. The proposed unsupervised (clustering) activity detection scheme works in an incremental manner, which contains two stages. In the first stage, streamed sensor data will be processed. A single-pass clustering algorithm is used to generate pre-clustered results for the next stage. In the second stage, pre-clustered results will be refined to form the final clusters, which means the clusters are built incrementally by adding one cluster at a time. Experiments on smartphone sensor data of five basic human activities show that the proposed scheme can get comparable results with traditional clustering algorithms but working in a streaming and incremental manner.

In order to develop more accurate activity recognition systems independent of smartphone models, effects of sensor differences across various smartphone models are investigated. The impairments of different smartphone embedded sensor models on HAR applications are presented. Outlier removal, interpolation, and filtering in pre-processing stage are proposed as mitigating techniques. Based on datasets collected from four distinct smartphones, the proposed mitigating techniques show positive effects on 10-fold cross validation, device-to-device validation, and leave-one-out validation. Improved performance for smartphone based human activity recognition is observed.

With the efforts of developing human activity recognition systems based on supervised learning approach, investigating a clustering based incremental activity recognition system with its potential applications, and applying techniques for alleviating sensor difference effects, a robust human activity recognition system can be trained in either supervised or unsupervised way and can be adapted to multiple devices with being less dependent on different sensor specifications.

Keywords: Activity recognition, supervised learning, sensors, incremental clustering, filter, interpolation, outliers

Acknowledgements

I would like to express my foremost gratitude to my advisor, Dr. Weiming Shen, for his invaluable guidance and support throughout my master's studies. I want to thank him for giving me freedom to pursue research topics that I am interested in, and being helpful and supportive whenever I need advice. His insights motivated me through my studies.

I own my sincere thanks to my co-advisor, Dr. Xianbin Wang. Besides his valuable suggestions and continuous help on the research topic, he shared advices on my future career and encouraged me to look around for today's industrial demands.

I would like to thank examiners Dr. Abdallah Shami, Dr. Jagath Samarabandu, and Dr. Evgueni Bordatchev for taking the time to review my thesis and provide valuable suggestions.

I would like to thank peer students in Dr. Wang's group, and other professors and students in the ECE department at University of Western Ontario.

Finally I thank my family for giving me unconditional love, support, and encouragement all the time.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
API	Application Program Interface
BMR	Basal Metabolic Rate
BSN	Body Sensor Network
CRF	Conditional Random Field
DT	Decision Tree
ECG	Electrocardiogram
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
GPS	Global Positioning System
HAR	Human Activity Recognition
HMM	Hidden Markov Model
ID3	Iterative Dichotomiser 3
IMU	Inertial Measurement Unit
KNLR	Kernel Nonlinear Regression
kNN	k-Nearest Neighbor
LPF	Low Pass Filter
MEMS	Microelectromechanical Systems
MLP	Multilayer Perceptron
NB	Naive Bayes

PCA	Principal Components Analysis
PT	Postural Transition
ROC	Receiver Operating Characteristic
SMA	Signal Magnitude Area
SVM	Support Vector Machine
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1

Introduction

1.1 Sensor based Human Activity Recognition

A Human Activity Recognition (HAR) system can automatically recognize physical activities [2], which is a key research issue in mobile and ubiquitous computing. An HAR system performs tasks of recognizing different human daily activities from simple to complex. The sensors involved in an HAR system can be video sensors, inertia sensors, and environment sensors. The GPS receiver can also be used for activity recognition but is limited to outdoor environments.

Based on the installation of sensors, HAR systems can be divided into three categories: wearable devices based sensing systems, smartphone sensing systems, and smart living environments. Although wearable devices and smart living environments can deliver good activity detection results, smartphone based applications are an increasingly prominent solution as smartphones have become an indispensable part of our daily life. Especially with the rapid evolution of hardware, ever-increasing computing and networking capacity, and rich embedded sensors, smartphone based HAR systems can tell us different kinds of human activities in real time using machine learning techniques. In addition, using smartphones for human activity recognition has a wide range of applications including healthcare, daily fitness recording, anomalous situation alerting, personal biometric signature identification, and indoor localization and navigation. All this benefits from the fast development of mobile phone software and hardware.

Smartphones in the market have embedded sensors, and the advanced MEMS (micro-electro-mechnical systems) design has enabled low-power and high-quality sensors for mobile sensing. The best-known MEMS sensors in smartphones are accelerometer and gyroscope, but there are a lot more MEMS sensors in today's mobile device like electronic compass, pressure sensor, light sensor, and microphone.

In the Android environments, the most commonly used and installed sensors can be categorized as follows [3]:

- Motion sensors: the motion sensors are based on inertial force.
- Environmental sensors: these sensors measure environmental parameters, like temperature and pressure, using barometers or thermometers.
- Position sensors: these sensors include orientation sensors and magnetometers, measuring the physical position of the device.

There are already plenty of mobile sensing applications in Google's Play store (updated in April 2016, shown in Table 1.1). Some of the apps incorporate other smart devices like bands and watches.

Index	**Vendor**	**App Name**	**Downloads**
1	FitnessKeeper Inc.	Runkeeper-GPS Track Run Walk	10,000,000
2	Google Inc.	Google Fit	10,000,000
3	ITER S.A	Fade: fall detector	5,000
4	MapMyFitness Inc.	Map My Fitness Workout Trainer	1,000,000
5	Microsoft Corporation	Microsoft Health	100,000
6	Noom Inc.	Noom Walk Pedometer	5,000,000
7	ProtoGeo	Moves	1,000,000
8	Runtastic	Runtastic Running & Fitness	10,000,000
9	Hamideh Kerdegari	Smart Fall Detection	1,000
10	Strava Inc.	Strava Running and Cycling GPS	5,000,000
11	tayutau	Pedometer	10,000,000

Table 1.1: Activity detection applications in the market

The features and functionalities of these mobile apps can be summarized as follows (Table 1.2):

- Activity Tracking
  - Automatic Mode: User's activities can be recognized automatically without user annotation. These applications can only detect some simple activities such as walking, running, and cycling. Most of the applications have to use GPS data or gait analysis.

Features		App Index
Activity Tracking	Automatically	2, 7
	Semi-automatic	1, 4, 5, 8, 10
	Manually	2, 4
Burned Calorie Calculation		1, 4, 5, 8, 10, 11
Chart/Graph Activity Records		1, 2, 4, 5, 7, 8, 10, 11
Fall Detection		3, 9
Food Log		4, 5
Health & Fitness Knowledge		5
Pedometer		2, 6, 7, 11
Routes on Map		1, 4, 5, 7, 8, 10
Social Networking		1, 4, 6, 10

Table 1.2: Product Features (the app index is shown in Table 1.1)

  - Semi-automatic Mode: The tracking tasks are trigged by users. The applications will then recognize activities accordingly. The tracking tasks are terminated by the apps automatically or manually by the users.
  - Manual Mode: In these applications, users will denote the type of activities they are performing during the tasks.

- Burned Calorie Calculation:
  Calculate the burned calories for individual activity events. The formulas of calorie expenditure are obtained from physiology studies. Currently, most calorie calculators will first calculate the Basal Metabolic Rate (BMR) and then obtain the total daily energy expenditure by the activity factor. According to different activity intensities (sedentary, lightly, moderately, very, and extremely active), the activity factor varies.

- Chart/Graph Records: Record and display activities through charts and graphs.

- Fall Detection: Detect falls as an anomaly behavior and send alerts to caregivers.

- Food Log: Users can record and track their daily meals. Such manually inputed food log can be used for future analyses.

- Health & Fitness Knowledge: Provide articles and training courses on related fitness domain knowledge.

- Pedometer: It can tell the user how many steps he/she has walked that day. It is popular in today's smart devices as a daily exercise counter and motivator. Today's step counters integrated in devices are based on MEMS inertial sensors and algorithms to detect steps. With the accurate multi-axis acceleration readings from advanced MEMS sensors, the best pedometer can be accurate to within ±5% error [4].

- Routes on Map: Track the user's activity and display on map. Record the geographic information where she/he goes or stays.

- Social Networking: Users can share their progress and activities on social networks.

1.2 Research Motivations

Although there are already various applications in the market as illustrated in the previous section, most of them do not make full use of the smartphone embedded inertial sensors. The granularity of such applications is also not sufficient. In some applications, only the activities of walking and motionless are recorded. Some applications that only use GPS signals fail to function in indoor environments.

The effectiveness of employing machine learning techniques on smartphone based sensor data is marked to be investigated, with the purpose of recognizing human activities. Different domain features and data processing techniques need to be studied. The unsupervised scheme for activity recognition on smartphones is rarely investigated in the literature. In addition, there have been very few studies on alleviating the impacts of mobile sensing performance differences across multiple devices. Our research is motivated by the demand of fulfilling the lack of studies on those topics in order to develop more accurate HAR algorithms.

1.3 Research Objectives

Based on the motivations described above, the study in HAR is still not sufficient. Three objectives of this thesis are derived to solve the problem of recognizing activities in different schemes by learning and leveraging smartphone sensor data with emphasis on considering the sensor difference effects. The objectives are as follows:

- A full study as well as developing human activity recognition applications under the supervised learning scheme is demanding. Collecting and recording smartphone sensor readings for different human activities in real scenarios is the preliminary of this work.

With regard to this, the effectiveness of various features extracted from raw sensor data will be investigated.

- Besides the supervised learning scheme, developing a method for clustering based activity recognition is proposed, which can help avoid the intensive labelling work. Specifically, the new method is expected to work in an incremental way that can deal with streamed sensor data.

- The third objective of our study is to address the issue of sensor heterogeneities across various smartphone embedded sensor models, as there are few studies in the literature focusing on the applicability of HAR systems for multiple platforms. In this study, techniques to alleviate the sensor difference effects are proposed and examined in order to obtain an improved performance of HAR applications.

1.4 Contributions

In this thesis, the activity recognition problem is studied by learning the advanced smartphone embedded multi-module sensor data. Based on a comprehensive literature survey and machine learning techniques, methods for both supervised and unsupervised learning for HAR are proposed. The main contributions of this thesis work can be summarized as follows:

- The proposed supervised learning based human activity detection system recognizes activities including walking, running, sitting, going upstairs, and going downstairs. Embedded sensors (a tri-axial accelerometer and a gyroscope sensor) are employed for motion data collection. A two-stage data analysis approach is used for prediction model generation: short period statistical analysis (maximum, minimum, mean, and standard deviation) and long period data analysis using machine learning. Both time-domain and frequency-domain features are extracted and analyzed. Our experiment results show that two time-domain features, mean and standard deviation, are good enough to recognize basic human activities. The system is implemented in an Android smartphone platform.

- An unsupervised (clustering) activity recognition scheme that works in an incremental manner is proposed, which contains two stages. In the first stage, streamed sensor data are processed. A single-pass clustering algorithm is used to generate pre-clustered results for the next stage. In the second stage, pre-clustered results are refined to form the final clusters, which means the clusters are built incrementally adding one cluster at a time. Experiment results on smartphone sensor data of five basic human activities show that the proposed scheme can get comparable results with traditional clustering algorithms but working in a streaming and incremental manner, which is promising for automatic annotated data collection activity discovery.

- In achieving a robust HAR system through various smartphone types, sensing heterogeneities are investigated through various smartphone sensors. Sensor heterogeneities are identified, such as sensor biases, sampling rate heterogeneity, and sampling rate instability. Possible mitigating techniques (interpolation, outlier removal, and filters) are investigated to alleviate these differences. Testing results show that the outlier removal and low pass filter have positive effects on alleviating the sensor differences.

1.5 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 provides an overview of basic concepts and algorithms in machine learning and data mining as well as a description of the WEKA framework used in this thesis work. A comprehensive literature review for supervised activity recognition is made, followed by four categories: HAR based on single or multiple accelerometers; HAR based on a combination of sensors; HAR based on smartphones; and other smartphone assisted sensing systems. Existing studies on unsupervised activity recognition are then briefly reviewed.

Based on the background and literature review provided in Chapter 2, Chapter 3, 4, and 5 present studies that fulfill the research issues proposed in section 1.3. Chapter 3 describes the proposed supervised HAR system. Starting with the data collection procedure, feature extraction methods are presented. Both time-domain and frequency domain features are examined. After the classification, there are satisfied classification accuracies for recognizing five basic activities using popular machine learning algorithms (C4.5 Decision Tree, SVM, Naive Bayes, and MLP). The classification results are presented and discussed in the last part of this chapter.

In Chapter 4, the unsupervised learning problem in HAR systems is studied. An incremental clustering method is introduced. The methodology involves the data collection and time-domain feature extraction. Then, the two-stage incremental clustering algorithm is presented followed by the clustering results on our collected dataset. Finally, the effects of two parameters (initial data point number and data window size) in the proposed incremental clustering algorithms are examined and results are discussed at the end of the chapter.

A study on the sensor heterogeneity effects is presented in Chapter 5. Firstly, a series of experiments for HAR on multiple devices are conducted, in which recognition performance degradation is observed. Next, the causes of sensor differences are discussed. To mitigate such differences, techniques (outlier removal, interpolation, and filtering) are proposed and validated. Detailed results on datasets collected from multiple smartphones and results for different classifiers are presented. Discussions based on the results are provided to reveal the techniques that have positive effects and the classifiers that are relatively less affected by the sensor differences.

In the end, Chapter 6 summarizes the contributions of this thesis and suggests some potential future work.

Chapter 2

Background and Literature Review

2.1 Data Mining and Machine Learning

Nowadays, an overwhelming amount of data is generated in the world and it is continuing to increase. Although such an amount of data can always be stored in disks, there is a gap between the generation of data and the understanding of data. To resolve this issue, data mining and machine learning techniques are utilized to discover patterns in data and make future predictions.

To better understand how to adopt a machine learning approach in a specific problem, we need to define a training set $\{\boldsymbol{x_1, ..., x_N}\}$ that contains N instances to train and tune the parameters of a model. The categories (or labels) of each instance in the training set are known in advance. In most cases, the categories are labelled manually during the data collection stage. The category of each instance in the training set can be expressed as the target vector $\boldsymbol{t}$. The output of running a machine learning algorithm can be expressed as computing a function $f(\boldsymbol{x})$ that takes a new instance as input and outputs a vector $\boldsymbol{y}$, where $\boldsymbol{y}$ is in the same form as the target vectors. The form of the function $f(\boldsymbol{x})$ can be determined in the training (or learning) phase based on the training set. Once the model is generated, we can utilize it to make predictions on the test set.

In some application scenarios, it will be easier to solve the machine learning problem if the original input variables are transformed into some new space of variables, which is called pre-processing [5]. Sometimes feature extraction has the same meaning as pre-processing. Note that both the test data and training data must be pre-processed using the same procedure. In addition, as the feature extraction might speed up the computation of the model, some information might be discarded. We have to be careful to check whether such kind of information is important in order to avoid harming the overall accuracy of the machine learning system.

The classification of machine learning problems can be summarized as follows. If the training data is comprised of input vectors $\boldsymbol{x}$ and the corresponding target vectors $\boldsymbol{t}$, such kind of machine learning problems are known as supervised learning problems. The supervised learning problems can be further divided into two categories: classification problems and regression problems. The aim of the classification problem is to assign each input to a discrete category, while the regression model will output one or more continuous variables. Another class of machine learning problems is the unsupervised learning problems, where the training data set only contains the input vectors $\boldsymbol{x}$ without labels. Unsupervised learning can be used for discovering groups with similar properties, which is called clustering, or it can be used to estimate the distribution of input data.

2.1.1 Classification Algorithms

In this subsection, some popular classification algorithms that are related to the scope of this thesis are briefly reviewed: Decision Tree (DT), Naive Bayes (NB), k-Nearest Neighbor (kNN), Support Vector Machines (SVMs), and Artificial Neural Networks (ANNs).

Decision Tree (DT)

Decision Tree (DT) is a non-parametric supervised learning method for classification. There are various decision tree implementations. Among them, ID3 (Iterative Dichotomiser 3) was developed by Ross Quinlan in 1986 [6]. It first calculates the entropy (or information gain) for each feature in the data set $\boldsymbol{S}$. Then it splits the data set $\boldsymbol{S}$ into subsets according to the feature which has the minimum entropy (or information gain). After that, a decision tree node is generated containing that feature. This procedure recurses until there are no remaining features.

C4.5 is an extension of the ID3 algorithm [7]. J48 is its Java implementation integrated in the WEKA data mining tool. Different from ID3, C4.5 can deal with both continuous and discrete features, handle training data with missing values, and support tree pruning after creation. C4.5 is referred as a statistical classifier. Let $\boldsymbol{S}$ be the data set for training. $freq(C_i, \boldsymbol{S})$ is the number of instances in $\boldsymbol{S}$ that belongs to class C_i. The number of instances in data set $\boldsymbol{S}$ is $|\boldsymbol{S}|$. Then the entropy of set $\boldsymbol{S}$ is defined as:

$$Info(\boldsymbol{S}) = -\sum_{i=1}^{k}((\frac{freq(C_i, \boldsymbol{S})}{|\boldsymbol{S}|}) \cdot \log_2(\frac{freq(C_i, \boldsymbol{S})}{|\boldsymbol{S}|})), \tag{2.1}$$

where k is the number of classes.

If set $\boldsymbol{S}$ is partitioned in accordance with n outcomes of one feature X, then the information entropy after using feature X to split S into n partitions can be formulated as:

$$Info_X(\boldsymbol{S}) = \sum_{i=1}^{n}((\frac{|\boldsymbol{S_i}|}{|\boldsymbol{S}|}) \cdot Info(\boldsymbol{S_i})), \tag{2.2}$$

where $\boldsymbol{S_i}$ is the subset of $\boldsymbol{S}$ in which all the instances belong to class C_i.

Using the above formulas, the information gain on feature X is defined as:

$$Gain(X) = Info(\boldsymbol{S}) - info_X(\boldsymbol{S}). \tag{2.3}$$

In the C4.5 algorithm, the splitting criterion is the normalized information gain defined above. For each feature in the data set, the normalized information gain is calculated. Then the feature which has the highest normalized information gain is chosen and a decision node is generated based on the selected feature. The recursion procedure continues until there are no remaining features to split.

The advantages of decision trees are that they are simple to understand and visualized easily. The trees can be easily expressed by Boolean logic. So decision trees use a white box model compared with a black box model (e.g. an artificial neural networks), which makes it easy to interpret. In order to generate the tree, little data preparation is required. However decision trees have some disadvantages. If the data are not well-generalized, over-complex trees may be formalized. Techniques like tree pruning and setting the maximum depth of the tree are necessary to avoid the overfitting problem. What's worse, there are some concepts that are hard to learn through a decision tree.

Naive Bayes

One of the supervised classifiers is the Naive Bayes classifier which is based on the Bayesian theorem. To explicitly explain it, we first get the basic Bayes theorem:

$$p(c_j|d) = \frac{p(d|c_j) \cdot p(c_j)}{p(d)}, \tag{2.4}$$

where $p(c_j|d)$ is the probability of instance d being in class c_j, which is also what we are going to compute. $p(d|c_j)$ is the conditional probability of instance d given the class c_j. This conditional probability also means that being in the class c_j makes d have some probability. $p(c_j)$ is the probability of the occurrence of class c_j (the frequent of class c_j occurred in the data set). $p(d)$ is the probability of instance d occurring.

The above shows the Bayes classification process when there is only one feature for an input instance. When there are multiple features in the dataset, the conditional probability

$p(d|c_j)$ can be estimated by:

$$p(d|c_j) = p(d_1|c_j) \cdot p(d_2|c_j) \cdots p(d_n|c_j). \tag{2.5}$$

Here we assume that features have independent distributions. $p(d_i|c_j)$ is the conditional probability of instance d given the class c_j for the *ith* feature.

Then the equation 2.4 can be rewritten as:

$$p(c_j|d) = \frac{p(c_j) \cdot \prod_{i=1}^{n} p(d_i|c_j)}{p(d)}. \tag{2.6}$$

Since $p(d)$ is a constant given by the dataset, the classification can be expressed as:

$$p(c_j|d) \propto p(c_j) \cdot \prod_{i=1}^{n} p(d_i|c_j). \tag{2.7}$$

The estimated class $\hat{c_j}$ can be derived as:

$$\hat{c_j} = \arg\max_{c_j} p(c_j) \cdot \prod_{i=1}^{n} p(d_i|c_j). \tag{2.8}$$

The advantages of the Naive Bayes classifier are that it is fast to train and make classifications, it is not sensitive to irrelevant features, and it can handle real and discrete data. However, the Naive Bayes classifier assumes features are independent, which is one of its drawbacks.

k-Nearest Neighbors (*k*NN)

The *k*-Nearest Neighbors algorithm is a non-parametric technique for classification and regression. It is also among one of the simplest machine learning algorithms. The kNNs is a kind of instance-based classifier, which means it does not abstract any information from the training data during the training stage. For an unknown instance, it is labelled as the same class as that of the majority of its k nearest neighbors. The nearest neighbors are measured by a distance function. The most commonly used distance functions are the Euclidean distance, the Manhattan distance, and the Minkowski distance, which are defined as follows. The Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)} = ||x_i - y_i||_2. \tag{2.9}$$

The Manhattan distance:

$$d(x, y) = \sum_{i=1}^{k} |x_i - y_i|. \tag{2.10}$$

The Minkowski distance:

$$d(x, y) = (\sum_{i=1}^{k} (|x_i - y_i|)^q)^{1/q}. \tag{2.11}$$

Another issue of k-Nearest Neigobor algorithm is the computation of the k nearest neighbors. The naive neighbor search method is the brute-force computation of distance between the pairs of points in the dataset. However this approach is only feasible for small datasets. To efficiently search the nearest neighbors, a variety of tree-based structures have been introduced, such as K-D tree and ball tree.

The advantage of kNN classifier is that it is a simple method and works well on basic machine learning problems. However, it does not learn anything from the training data. And if the dataset is large with high dimensions, the searching of k neighbors can be slow and inefficient. What's worse, kNN classifier can be easily affected by noisy data.

Support Vector Machines (SVMs)

The support vector machine method is considered as an optimization algorithm. The basic ideas of SVMs are that it creates optimal hyperplane for linearly separable patterns, and for the patterns that are not linear separable, kernel functions can be used to transform the original data into new space. SVMs and kernel methods have a theoretical model that guarantees the performance. In addition, SVMs are not affected by local minima and the curse of dimensionality. For simplicity's sake, we first consider the two-class classification problem with the linear model:

$$y(x) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b, \tag{2.12}$$

where $\phi(\boldsymbol{x})$ represents a feature-space transformation. The training dataset has N input vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$, and the corresponding targets are $t_1, ..., t_N$ where $t_n \in \{-1, +1\}$. When a new data instance $\boldsymbol{x}$ comes, it will be classified according to the sign of $y(\boldsymbol{x})$. If the training dataset is linearly separable and the function 2.12 satisfies $y(\boldsymbol{x}) > 0$, the data instance has $t_n = +1$; If $y(\boldsymbol{x}) < 0$, the data instance has $t_n = -1$. So that there is a constraint that $t_n y(\boldsymbol{x}_n) > 0$ for all training data instances.

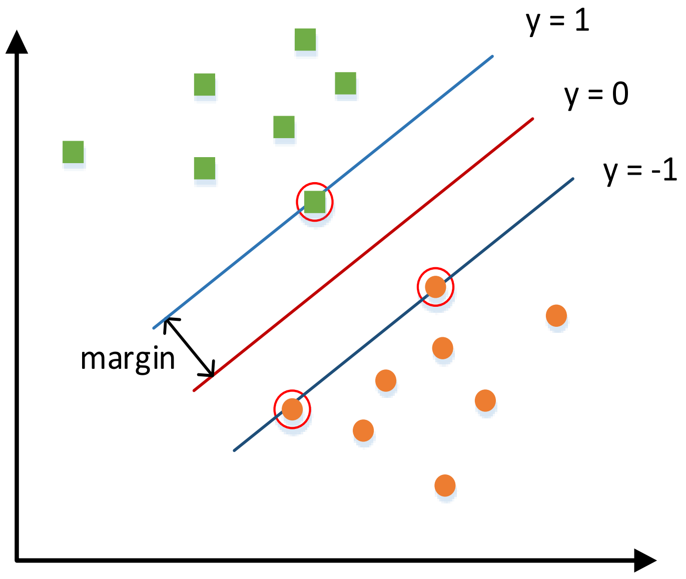


Figure 2.1: The maximum margin, where the margin is defined as the perpendicular distance between the decision boundary

Then the concept of the margin is introduced. It is defined as the smallest distance between the decision boundary, as shown in Figure 2.1. The support vectors are defined as the data instances that lie closest to the decision boundary, which are the most difficult to classify. The SVMs maximize the margin around the decision hyperplane. This is formulated as a quadratic programming problem. As formula 2.12 indicates, the perpendicular distance of a instance $\boldsymbol{x}$ from the decision hyperplane is $|y(\boldsymbol{x})|/||\boldsymbol{w}||$. As denoted above, if all the instances are correctly classified, there is $t_n y(\boldsymbol{x}_n) > 0$. The distance of an instance $\boldsymbol{x}_n$ to the decision boundary is:

$$\frac{t_n y(\boldsymbol{x}_n)}{||\boldsymbol{w}||} = \frac{t_n(\mathbf{w}^T \phi(\boldsymbol{x}_n) + b)}{||w||}. \tag{2.13}$$

In order to maximize the margin, the parameters $\boldsymbol{w}$ and b are optimized. The maximum margin can be derived by solving:

$$\arg\max_{w,b} \left\{ \frac{1}{||\boldsymbol{w}||} \min_{\boldsymbol{n}} \left[t_n(\mathbf{w}^T \phi(\boldsymbol{x}_n) + b) \right] \right\}. \tag{2.14}$$

To solve this optimization problem, it can be transformed [5] into:

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2. \tag{2.15}$$

Then Lagrange multipliers $a_n \geq 0$ are introduced with the Lagrangian function:

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{n=1}^{N} a_n\{t_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) + b) - 1\}, \tag{2.16}$$

where $\boldsymbol{a} = (a_1, ..., a_N)^T$. The dual problem of optimizing the maximum margin can be expressed as:

$$Maximize : \tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) \tag{2.17}$$

Subject to:

$$a_n \geq 0, n = 1, ..., N \tag{2.18}$$

$$\sum_{n=1}^{N} a_n t_n = 0. \tag{2.19}$$

When the model is trained, the class of a new instance can be evaluated by function 2.12. The above descriptions are the two-class SVM classification scenario. In practice, there may be problems involing $K > 2$ classes. One approach to deal with multiclass problem is to call the one-versus-the-rest approach. It divides the dataset into two classes, C_k and the remaining $K - 1$ classes. Another class is called one-versus-one approach. It trains $K(K - 1)/2$ two-class SVMs for all possible pairs of classes. The instance to be classified is determined by a voting according to which class has the highest votes.

Artificial Neural Networks (ANNs)

ANNs are models that are composed of many nonlinear computational nodes in parallel and perform like biological neural networks [8]. The best known model of ANNs is the feed-forward neural network, which is also known as the multilayer perceptron. The nodes or neurons in the neural network are connected via weights and updated during the learning process.

For a neuron the structure is represented in Figure 2.2. Figure 2.3 shows the example of a two-layer neural network. The single neuron is based on the linear combinations of nonlinear functions $\phi_j(x)$, represented as:

$$y(x, w) = f(\sum_{j=1}^{M} w_j \phi_j(x)), \tag{2.20}$$

where the $f(\cdot)$ is the nonlinear activation function, $\phi_j(x)$ is the basis function. The whole network function using sigmoidal activation functions can be written as:

$$y_k(x, w) = \sigma(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}) + w_{k0}^{(2)}). \tag{2.21}$$

The sigmoid function $\sigma(a) = \frac{1}{1+exp(-a)}$ is used as the activation function. The nonlinear functions $h(\cdot)$ are the hidden unit activation functions and are chosen to be sigmoidal functions or the 'tanh' function. $w_{ji}^{(1)}$ refers to the weight from the node i to the node j in the first layer of the neural network.

The training of the neural network involves minimizing the error function:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} ||\boldsymbol{y}(\boldsymbol{x}_n, \boldsymbol{w}) - \boldsymbol{t}_n||^2. \tag{2.22}$$

The parameter $\boldsymbol{w}$ can be found by minimizing $E(\boldsymbol{w})$ through the maximum likelihood solution [5]. However, in practice the error function $E(\boldsymbol{w})$ can be nonconvex, the local maxima may be found.

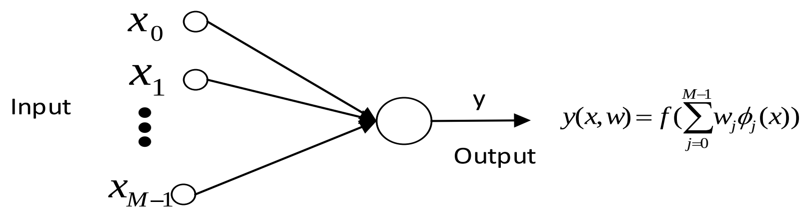


Figure 2.2: A model of a neuron or node that forms a weighted sum of M inputs and output the result through a nonlinearity

The advantages of artificial neural networks are that they are nonlinear models and are easy to train and use compared with statistical methods. Furthermore neural network models can implicitly reflect the nonlinear and complex relationships of the inputs [9]. However, neural network models are a black box that is limited to explicitly describe the structure of approximated functions by the network. What's worse, training neural network models needs greater computational efforts. In the meanwhile, such models are prone to overfitting.

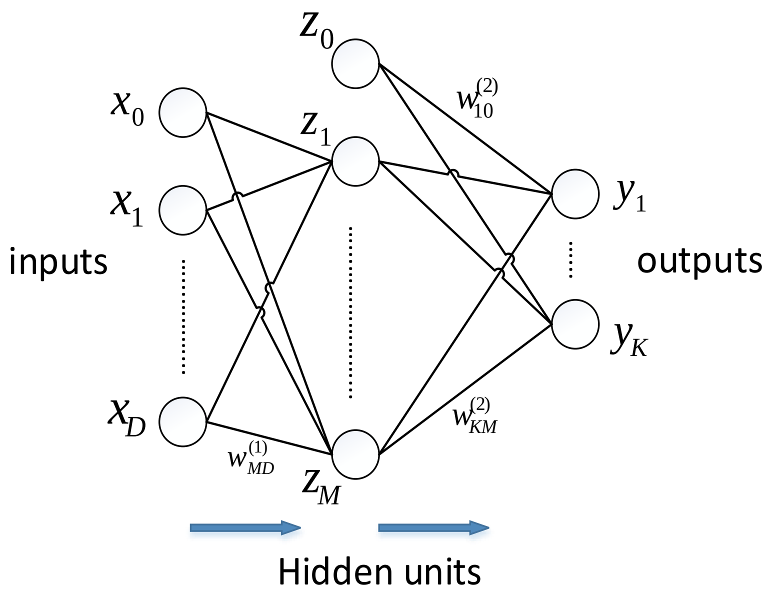


Figure 2.3: Example for the two-layer neural network

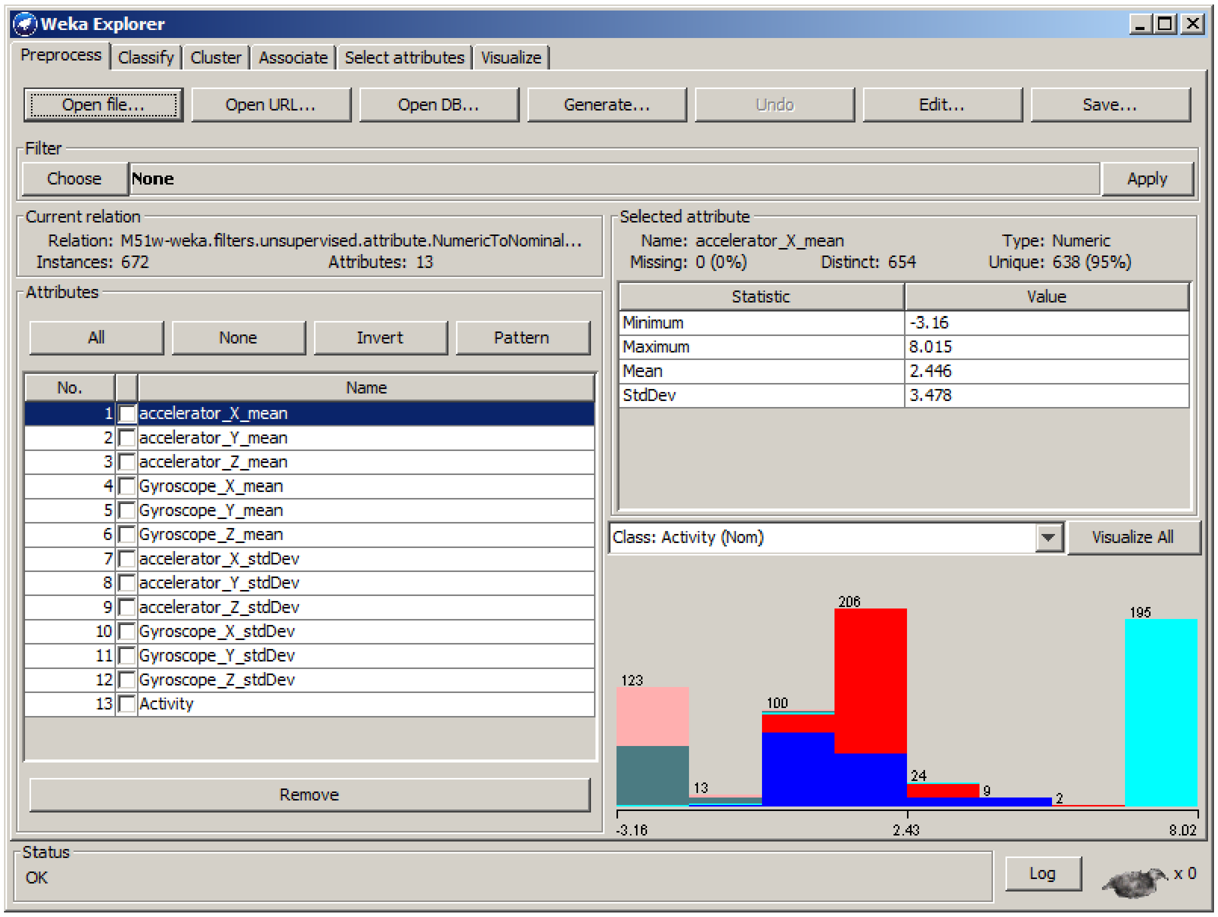
2.1.2 WEKA Machine Learning Framework

WEKA stands for Waikato Environment for Knowledge Analysis, which started in 1992 [10] and was funded by the New Zealand government. The purpose of WEKA project was to create a collection of learning algorithms as well as a framework inside which new algorithms can be implemented. Now this machine learning algorithms toolkit is an open-source Java-based machine learning software framework.

The WEKA Workbench

The WEKA project provides functions that researchers can use to try out and compare different machine learning methods on new datasets. In addition, APIs are provided for customized solutions and explorations. Classification, regression, clustering, and attribute selection are integrated in the workbench.

The main graphical user interface of WEKA is call the "Explorer" (shown in Figure 2.4), which has six components: Preprocess, Classify, Cluster, Associate, Select attributes, and Visualize. The “Preprocess” panel is used for data preprocessing. Different preprocessing tools are called “filters” in WEKA. The second panel called “Classify” includes various classification and regression algorithms. Validation methods as well as evaluation visualization like ROC curves can be drawn in this panel. In addition, WEKA also supports unsupervised learning algorithms. Evaluation and visualization of the clustering algorithms are provided in the third panel. The fourth panel enables the methods for association rule mining, which is used for discovering interesting relations between items in datasets. Attributes selection is another important task in practical machine learning and data mining. The “Select attributes” panel gives access to some attribute selection algorithms, such as principal components analysis (PCA), information gain evaluation, and gain ratio attribute evaluation. The last panel is the data visualization, which provides the color scatter plot of the data matrix that can be visualized.



Filter
Choose None Apply

Current relation
Relation: M51w-weka.filters.unsupervised.attribute.NumericToNominal...
Instances: 672 Attributes: 13

Attributes

No.		Name
1	☐	accelerator_X_mean
2	☐	accelerator_Y_mean
3	☐	accelerator_Z_mean
4	☐	Gyroscope_X_mean
5	☐	Gyroscope_Y_mean
6	☐	Gyroscope_Z_mean
7	☐	accelerator_X_stdDev
8	☐	accelerator_Y_stdDev
9	☐	accelerator_Z_stdDev
10	☐	Gyroscope_X_stdDev
11	☐	Gyroscope_Y_stdDev
12	☐	Gyroscope_Z_stdDev
13	☐	Activity

Selected attribute
Name: accelerator_X_mean Type: Numeric
Missing: 0 (0%) Distinct: 654 Unique: 638 (95%)

Statistic	Value
Minimum	-3.16
Maximum	8.015
Mean	2.446
StdDev	3.478

Class: Activity (Nom)

Status
OK

Figure 2.4: The WEKA Explorer GUI

Core of WEKA

New learning algorithms, pre-processing filters, and usability improvements and supports have been added to WEKA since version 3.4. There are more than 690 Java class files.

Some of the classification algorithms in WEKA 3.6 include

- Bayesian logistic regression [11]: the Bayesian logistic regression approach for text categorization with a Laplace prior to avoid overfitting.
- Bayes net: Bayes network learning using diverse searching algorithms and quality measures, such as hill climber, genetic search, and simulated annealing.
- Naive bayes [12]: a Naive Bayes classifier using estimator classes.
- LibLINEAR [13]: WEKA provides a wrapper class for the libnear tools, which support logistic regression and linear support vector machines.
- LibSVM [14]: a wrapper class is provided for the libsvm tools, which support solving SVM optimization problems, multiclass classifications, probability estimation, and parameter selection.
- Multilayer perceptron: a classifier that can use backpropagation to classify instances. The built network can be monitored and modified during the training time. The nodes used in the network are all sigmoid.
- RBF network: a normalized Gaussian radial basis function network.
- IB1: a nearest-neighbour classifier implementation that uses normalized Euclidean distance.
- IBK: a K-nearest neighbours classifier. The value of K can be selected by cross-validation.
- J48: a C4.5 decision tree implementation [7].
- Random forest: an implementation for constructing a forest of random trees, which is also a combination of tree predictor [15].

Some of the clustering algorithms in WEKA 3.6 include

- Cobweb: an implementation for the incremental system of hierarchical conceptual clustering.
- DBSCAN [16]: a basic implementation of density-based spatial clustering of applications with noise.

- EM: a simple implementation of expectation maximization clustering algorithm, which will find the MLE of the marginal likelihood by applying the expectation step (E step) and the maximization step (M step).
- Hierarchical clusterer: implementations of a number of agglomorative clustering algorithms. The agglomorative clustering is a ”bottom up” method, which means each instance starts in its own cluster.

WEKA has also integrated a number of preprocessing tools, which are called ”Filters” in WEKA’s Preprocess panel. Some of the filters include:

- Supervised filters: filters that take class distributions into account.
  - Add classification: adding the classification, the class distribution, and the error flag to the dataset.
  - Attribute selection: an attribute filter that is flexible and support various search and evaluation methods, such as Cfs subset evaluation, Chi square attribute evaluation, Gain ratio attribute evaluation, Best first search, Genetic search, and Greedy stepwise search.
  - Resample: a filter that can produce a random subsample of a dataset using either replacement or without replacement.
- Unsupervised filters: filters that work without taking any class distributions into account.
  - Add cluster: a filter that can add a new nominal attribute, which represents the cluster assigned to each instance. Different clustering algorithms can be selected in the filter.
  - Add values: a filter that can add the labels from a given list to a denoted attribute.
  - Add: a filter that can add a new attribute to the dataset.
  - Numeric to nominal: a filter that can convert numeric attributes to nominal ones.
  - Remove: a filter that can remove a range of attributes from the dataset.
  - Replace missing values: a filter that can replace all missing values for nominal and numeric attributes in a dataset.
  - Normalize: a filter that can normalize numeric values in the dataset. The resulting values can be [0,1] or in the range [-1,+1].
  - Standardize: a filter that can standardize numeric attributes in the dataset, which makes the dataset have zero mean and unit variance.

  - Remove with values: an instance filter that can filter the instances according to the value of an attribute.
  - Remove range: an instance filter that can remove a given range of instances in a dataset.
  - Interquartile range: a filter that can detect outliers and extreme values based on interquartile range.
  - Wavelet: a filter that performs wavelet transformation.

WEKA API

Although WEKA provides graphical user interface, it also defines API (application programming interfaces) that make it possible to embed WEKA's functions into other projects. The most common components in WEKA are

- weka.core.Instances: class for handling instances (data to be processed, trained, and tested).
- weka.filters.Filter: class for instance filters. Filters take instances as inputs, perform some transformations, and output the results.
- weka.classifiers/weka.clusterers: interface classes. When implemented, predictions or clusterings can be made.
- weka.classifiers.Evaluation: class for evaluating machine learning models with options of different evaluation mode.
- weka.attributeSelection: class for removing or modifying attributes in the dataset.

The above classes are the most common used components when using WEKA in our own codes. The basic steps for solving a classification problem are: building feature vector, training a classifier, testing a classifier, and using a classifier. More details about WEKA framework can be found in the WEKA developer's manual [17].

2.2 Human Activity Recognition based on Supervised Learning

Human activity recognition (HAR) has been widely studied in the literature. Research in machine learning developed a number of inference methods. Probabilistic methods are widely used for human activity systems, such as Hidden Markov Models (HMMs), Conditional Random Fields (CRFs), and Bayesian networks. Discriminative approaches, such as Support Vector Machines (SVMs), C4.5 decision trees, and neural networks, are also successfully applied in the area of HAR.

The task of activity detection can be performed by employing variety of sensors, including video cameras, environmental sensors (light, relative humidity, temperature, and pressure sensors), and wearable sensors. The sensors used can be on-board smartphone sensors or sensors installed in wearable devices.

Although camera sensors can provide a rich and unique set of information that cannot be obtained from other types of sensors, camera-based methods require continuously monitoring a person's activities, which requires vast storage space and computation resources. In addition, people may feel uncomfortable being watched by cameras continuously [18]. An example of such a camera-based indoor human activity monitoring system can be found in [19], which provides continuous video monitoring and intelligent video processing. Another purpose for utilizing camera sensors is providing the ground truth for human activity recognition systems. For instance, in [20], the proposed activity recognition system employs varieties of sensors, where the cameras are used for documentation and visual annotation. In the daily routines recognition system [21], snapshots taken by the mobile phone's built-in camera are used for annotations.

Environmental sensors can track and record the user's interaction with the environment. For example, in the experimental environment of [20], wireless Bluetooth acceleration and gyroscope sensors are attached on objects used in the test scenario, recording the usage of objects. Moreover wired microphone arrays are used at room side in order to sense ambient sound. In addition, reed switches are instrumented on doors, drawers, and shelves to sense usage as well as provide ground truth. However, the limitations of the environmental sensors are that they are restricted to certain circumstances and building layouts, which makes the HAR system not universal. One well designed and trained HAR system may not be simply immigrated to another ambient environment. Also, the deployment cost of such sensors are relatively high.

Wearable sensors that are worn on human body can determine physical states and characters of the subject's daily activities. Such sensors include inertial sensors (accelerometers and gyroscopes), GPS, and even magnetic field sensors are widely used in activity recognition applications.

2.2.1 Human Activity Recognition Based on Single or Multiple Accelerometer Sensors

In some studies, one or more accelerometers attached to different parts of human body are used for activity recognition. Dong and Biawas [22] presented a wearable sensor network that was designed for human activity detection. In their system, each of three sensors, which were worn on ankle, thigh, and wrist, was equipped with an ADXL202 accelerometer. The mean and entropy of acceleration were computed over time windows and were used for activity detection. Ten dynamic activities (bicep curls, riding a bike briskly/slowly, jogging, jumping, walking briskly/slowly, sweeping, squatting, and stair climbing) can be detected with good accuracy. Similarly, Curone et al. [23] used one wearable tri-axial accelerometer for activity detection. Signal magnitude area (SMA) [24] was introduced in their method, which was an effective measurement for identifying activity intensity. In their system, human postures (upright standing, move trunk and arms, picking up objects, moving lying down, etc.) can be detected in real time. Al-Ani et al. [25] combined wavelet and hidden Markov models for on-line human activity detection. A bi-axial accelerometer ADXL202E was attached on the belt of the person to identify activities (walking slowly, walking quickly, sitting down-getting up, fall during walking, fall from a position upright). In Bao and Intille's study [26], acceleration data were collected from 20 individuals and fed into decision tree classifiers to detect 20 daily activities. With five bi-axial accelerometers attached on different locations of human body, the HAR system can reach a reasonable accuracy. They also reported that with just two accelerometers (thigh and wrist), the recognition performance dropped slightly. Mannini and Sabatini [27] applied Hidden Markov Models on the dataset provided by Bao and Intille [26] and got satisfied classification accuracy. Banos et al. [28] employed a group of accelerometers attached to different parts of the body (hip, wrist, arm, ankle, and thigh) for tracking four activities: walking, sitting, standing, and running. Zhang et al. [29] tested the number of accelerometers and settings that may affect the accuracy to identify eight activities, standing, walking, running, jumping, lying, sitting, tooth brushing, and eating. SVM was used for their data classification.

There have been some datasets created and published by researchers and used by others around the world. WISDM (Wireless Sensor Data Mining) is a dataset published by Fordham University, which contains data collected in controlled and laboratory conditions [30]. Only the cell phone-embedded accelerometer is used in the data collection. The sampling rate is 20Hz and the total number of instances is 1,098,207. The number of attributes is six (user index, activity type, timestamp, x-acceleration, y-acceleration, and z-acceleration). 29 subjects are involved in the collection task. Six activities are labelled (walking (38.6%), jogging

(31.2%), upstairs (11.2%), downstairs (9.1%), sitting (5.5%), standing (4.4%)). From the raw time series data, time-domain features are extracted in the publisher's study, such as the average value, standard deviation, average absolute difference, average resultant acceleration, time between peaks, and binned distribution.

The Skoda mini checkpoint dataset is another dataset published by the Wearable Computing Group in ETH [31]. It describes one subject performing ten different manipulative gestures in a car maintenance scenario. The ten activities include write on notepad, open/close hood, check gaps on the front door, open left front door, check trunk gaps, check steering wheel, etc. In total, there are 20 3D acceleration sensors (10 on the left arm and 10 on the right hand used for data collection).

2.2.2 Human Activity Recognition Based on the Combination Accelerometers and Other Types of Sensors

Some other studies combine multiple sensors, such as gyroscopes, microphones, and magnetic field sensors. In the abnormal activity detection system proposed by Yin et al. [32], three MTS310CA sensor boards were attached to different parts of a human body with seven features selected as inputs (light, temperature, microphone, bi-axial accelerometer, and bi-axial magnetometer). In their approach, a one-class support vector machine (SVM) and a kernel nonlinear regression (KNLR) were employed for the two-phase training and activity detection. Their work makes it possible for abnormal activity models to be automatically derived without the need to explicitly label the abnormal training data. Lee and Masc [33] proposed a system using a bi-axial accelerometer, a digital compass sensor, and a gyroscope that can detect sitting, standing, walking, going up a stairway, and going down a stairway. In Najafi et al. [34], the authors attached one kinematic sensor, which was composed of one gyroscope and two accelerometers, to the subject's chest. The physical activity monitoring system can not only detect body postures (sitting, standing, and lying), but also different postural transitions (PTs) (i.e., lying-to-sitting, sitting-to-lying, and turning the body in bed). Subramanya et al. [35] introduced a customized wearable sensor system, which is consisted of a tri-axial accelerometer, two microphones, phototransistors, temperature and barometric pressure sensors, and GPS to identify activities including walking, running, going up/down stairs, driving a vehicle, and going indoors using graph models. Parkka et al. [36] also built a system using many different sensors (three accelerometers with two on chest and one on wrist, one microphone attached to wrist, and one compass attached to chest) to recognize activities such as lying, sitting/standing, walking, Nordic walking, running, rowing, and cycling. Decision trees and artificial neural networks were tested in their system. In Tapia et al. [37], five tri-axial accelerometers and a wireless heart rate monitor were used to identify activities with intensities. Banos et al. [38] examined the effect of signal segmentation and reported that the interval of 1-2 seconds would provide the best trade-off between recognition speed and accuracy. Since body sensor networks (BSNs) based activity detection systems usually have difficulties in real-world applications due to the programming complexity and the lack of high-level software abstractions, Fortino et al. [39] developed a programming framework called signal processing in node environment (SPINE) to support rapid development of BSN applications.

Among these studies, Chavarriaga et al. [40] made their dataset public, which is called the Opportunity Dataset. This dataset is brought into the public in order to deal with the benchmarking requirements for activity recognition. Previously, each research group reports the results of their proposed methods based on their own dataset with specific experimental setups. It is hard to compare and assess different methods on various experimental environments. This dataset is proposed by the Wearable Computing Group in ETH. The dataset aims at providing a platform that enables the comparison of different methods in the same condition.

In this dataset, the researchers provide 18 labelled sessions from four subjects. In addition, four different tasks are identified: multimodal activity recognition (modes of locomotion), automatic segmentation task, multimodal activity recognition of gestures, and recognition of gestures with noise. The sensors deployed in the environment are described as follows. Five IMUs are mounted in a motion jacket, 12 bluetooth accelerometers and two inertial sensors are placed on the feed. In each task, only a subset of the provided sensors are used.

For the modes of locomotion task, four sub-activities are further identified: standing, walking, sitting, and lying. For the automatic segmentation task, classes are null class and activity class. For the task of gesture recognition, right-arm gestures performed in a daily activities scenario are recognized. Classes include open/close dishwasher, open/close doors, clean table, move cups. etc. In the last task, rotational and additive noise is added to the testing dataset for recognizing gestures.

2.2.3 Human Activity Recognition Based on Smartphone Sensors

As mentioned above, with ever increasing computing power, convenient Internet connections, and numerous mobile applications, smartphones have been an indispensable role in our daily life. What's more, even low-end smartphones have a set of sensors (accelerometer, GPS, and gyroscope, etc.), which make it possible to use a smartphone to detect human activities. In [41], a mobile phone based fall detection system was proposed and implemented in the Android platform. In their experiments, the mobile phones were still attached to different locations of human body. In [42], a hierarchical SVM classifier was used for recognizing walking, going-upstairs, going-downstairs, running, and motionless. The features selected from accelerometer data were standard deviation of Y axis, correlation of Y, Z axis, autoregressive fitting of Y axis, and the signal magnitude area (SMA). Also, the mean, standard deviation, and skewness of pitch were selected for classification. Boyle et al. [43] proposed a gait-based-recognition system to identify walking activity. The wavelet transform was employed to extract features from raw data and the k Nearest Neighbors (kNN) algorithm was used to perform the classification. Miluzzo et al. [44] developed the CenceMe application, which can distinguish sitting, standing, walking, and running from acceleration data collected by a Nokia N95 smartphone. In Brezmes et al. [45], the authors implemented a real-time classification system for some basic human activities from accelerometer data, including walking, climbing-down stairs, climbing-up stairs, sitting down, standing up, and falling. Their monitoring system was decentralized, which meant no server processing data are involved. Kwapisz et al. [30] built another HAR system which can identify six human activities. The data were collected from smartphone embedded accelerometers placed in the subjects' front pants leg pockets. Chiang et al. [46] proposed a portable activity pattern recognition system to identify physical activities. Accelerometer and GPS data were collected and four classifiers were tested. Anjum et al. [47] proposed a smartphone application using accelerometer and gyroscope sensors as well as GPS signals to detect seven activities. C4.5 decision tree classifier was reported to yield the best performance. All the systems reviewed above are based on the supervised learning approach for activity detection. Kwon et al. [48] examined the unsupervised learning method for human activity recognition based on smartphone sensors. A comprehensive survey on smartphone sensor based activity recognition can be found in [49].

The Heterogeneity Human Activity Recognition (HHAR) dataset is a public dataset based on smartphone sensors for HAR studies collected from a variety of different device models and use-scenarios, which reflects the sensing heterogeneities when HAR applications are deployed in real-world scenarios [50]. This dataset contains the sensor readings of nine users with six different activities: biking, sitting, standing, walking, stair up, and stair down. Smartphone embedded accelerometer and gyroscope sensors are employed in the study, which work at the highest sampling rate the device allows. Different from other public datasets, the HHAR dataset incorporates four smartwatches (two models) and eight smartphones (four models).

2.2.4 Other Smartphone Based Sensing Systems

Besides detecting human activities, smartphone based sensing systems can perform other tasks. Such tasks include monitoring electrocardiogram [51], in which a smartphone is integrated with a customized phone-case-type ECG sensor to support the pervasive healthcare apps.

Medrano et al. [52] proposed a smartphone based fall detector, where detection algorithms, like nearest neighbor, and one-class support vector machine, are tested for fall detection. Casilari et al. [53] systematically reviewed the existing efforts of Android device-based solutions for fall detection with built-in sensors. A wearable and context-aware ECG monitoring system is proposed in [54], which is integrated with built-in smartphone sensors (accelerometer, gyroscope, and orientation sensors) with a self-designed ECG sensor. The system can provide improved diagnosis accuracy for abnormal ECG pattern in different activities.

2.3 Human Activity Recognition based on Unsupervised Learning

In the literature, activity detection is considered as a supervised learning problem. For the supervised scheme, a training dataset is needed with the ground truth labels of different activities. However, the labeling work may be tedious and labor-intensive. In this respect, unsupervised approaches are investigated by some researchers [21] [55] [56]. Kwon et al. [48] proposed unsupervised learning methods for human activity recognition using smartphone sensors and studied the performances of different clustering methods even when the number of activities is unknown. Huynh et al. [21] explored an unsupervised learning method based on the concept of multiple eigenspaces to detect individual activities in accelerometer data. A sensor network based smart home environment was proposed in [55], where an unsupervised approach was claimed useful for detecting activities. The authors in [56] employed the unsupervised method for activity recognition with three accelerometers attached to human body. Their results reported were comparable to supervised learning methods.

To the best of our knowledge, the most similar work to incremental activity clustering is Ong et al. [57], which contains incremental clustering concepts that can automatically discover human activities from unlabeled RGB-D sensors. However, Ong's method is based on video sensors and the number of clusters is not specified. In [58], another incremental learning framework is proposed using video sensors. The probabilistic neural networks and adjustable fuzzy clustering based incremental learning framework [59] is a supervised method rather than working in an unsupervised way. In [60], the authors introduce a tree-based clustering method that can deal with overlapping clusters, which allows overlapping clusters to be obtained when data increase. It has to be noticed that in the literature, when we talk about incremental clustering, it has two meanings [61]: (1) Single pass incremental algorithms where data points are processed at each iteration and cluster centers are refined [62] [63] [64]; (2) Algorithms where clusters are built incrementally [65] [66]. An HAR system that incorporates both the unsupervised learning and the two kinds of incremental features is rarely studied.

2.4 Summary

In this chapter, we first review some basic concepts that are fundamental for the rest of this thesis. Data mining and machine learning methods and algorithms are presented from the mathematics point of view. Then a comprehensive literature review is given based on the different categories of HAR scenarios. Under the supervised learning scheme, HAR systems are divided into four classes according to the deployment methods of sensors and sensor types. The background and literature review provide a good foundation for us to develop our own HAR applications based on both supervised and unsupervised learning as well as explore the HAR applicability across multiple devices in the next three chapters.

Chapter 3

Human Activity Recognition Based on Supervised Learning

3.1 Introduction

Most smartphones are embedded with tri-axial accelerometers and gyroscopes as well as other motion and environment sensors. Using smartphones for human activity recognition has more flexibility than wearable devices, along with a wider range of applications including healthcare, daily fitness recording, and anomalous situations alerting.

In this chapter, a smartphone based human activity recognition system is proposed to detect human activities including walking, running, sitting, going upstairs, and going downstairs. As standing still or putting the phone on a table is similar to the situation of sitting, sitting is used in this paper to represent the motionless state. Also, it is assumed that people carry a smartphone in pockets with a fixed position. The smartphone based applications perform the tasks of collecting sensor data and making predictions. Both time and frequency domain features are extracted from the data collected by smartphone embedded sensors. Several machine learning algorithms (J48, SVM, Naive Bayes, and Multilayer Perceptron) are employed to make classifications. The system is implemented in an Android platform.

Figure 3.1 shows a basic process of smartphone based human activity recognition system based on supervised learning. The data collection and feature extraction are performed using a smartphone while the model generation is performed in a computer. Then the model is implemented in another app for activity prediction. In the training phase, the label information serves as the ground truth for the purpose of training and evaluating. In the prediction phase, the same features are extracted and fed into the generated classifier model in order to get the corresponding predicted activity label.

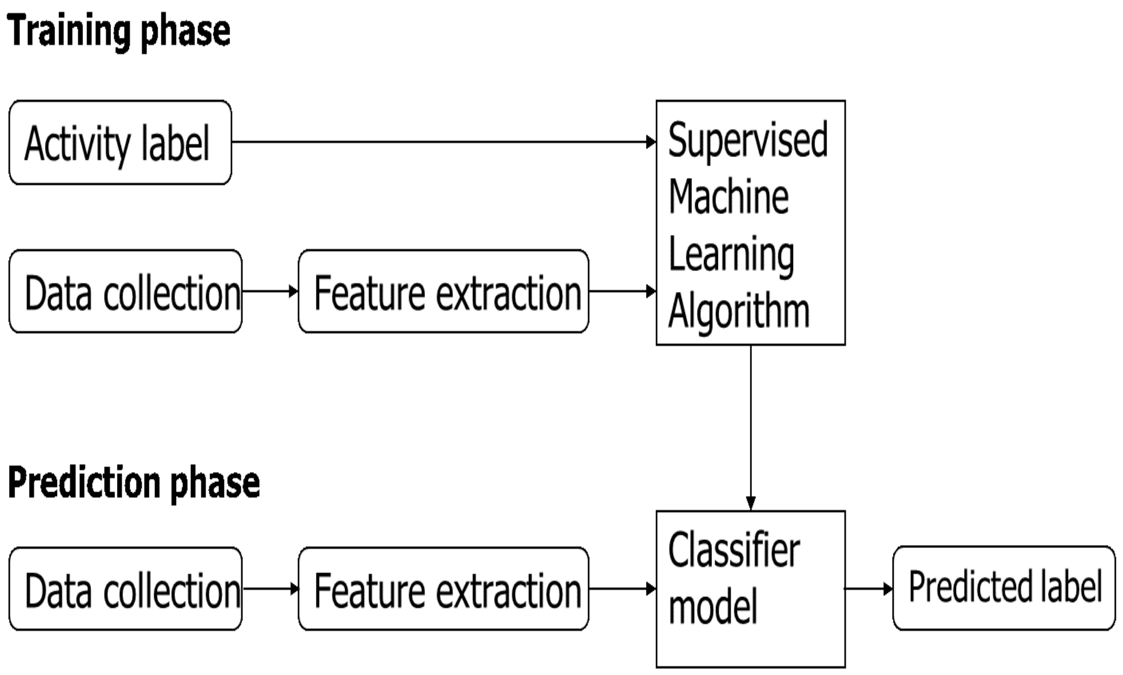


Figure 3.1: Activity recognition process

3.2 Data Collection

In this study, the signals detected from the embedded accelerometer and gyroscope in smartphones are used to infer different human activities. To collect the sensor data, one subject puts a smartphone in his/her pocket with a fixed position and performs some activities. Preliminary data collection and analyses show little difference when the smartphone is placed in different orientations or in different pockets, which would make the smartphone based detection system simpler and more practical. However, in order to make the data easy to analyze, the phone's position of placement is fixed when performing the data collection tasks. An Android application is running during the experiments. Sensor data are saved as CSV files. In order to increase the classification accuracy, data recorded at the beginning and the end of each activity is trimmed from the data files. Due to the Android APIs [3], the sensor data are recorded with a sampling rate of 50000 microsecond (20 Hz) [26].

The incoming data are the raw accelerometer and gyroscope readings in three axes. The definition of the coordinate-system is shown in Figure 3.2. Figure 3.3 shows the description of the sensor data collection app and the recognition app based on decision tree. When it starts, raw sensor data are recorded accompanied with some personal information (height, weight, age, and gender). In addition, the placement of the smartphone is stored. The activity label is pre-set before the collection starts.

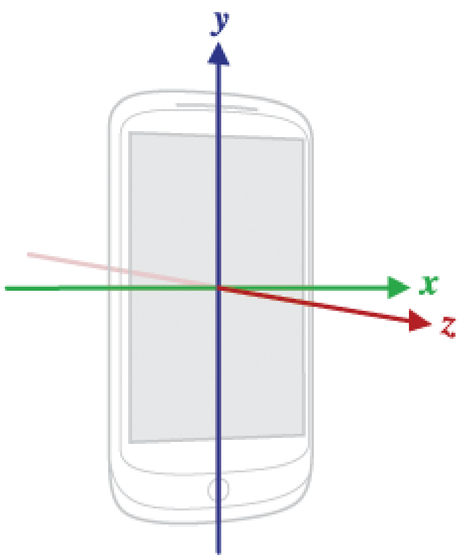
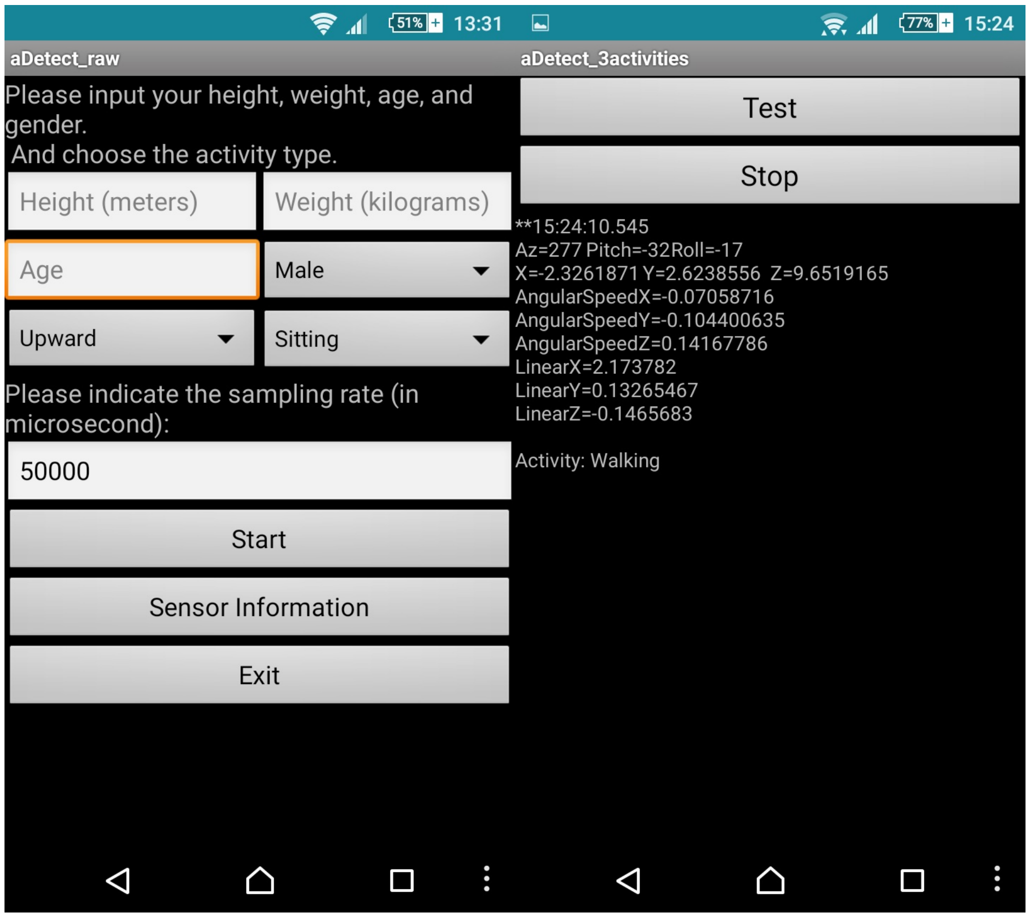


Figure 3.2: The definition of the coordinate-system in Android [1]



Please input your height, weight, age, and gender.
And choose the activity type.

Please indicate the sampling rate (in microsecond):

50000

```
**15:24:10.545
Az=277 Pitch=-32Roll=-17
X=-2.3261871 Y=2.6238556  Z=9.6519165
AngularSpeedX=-0.07058716
AngularSpeedY=-0.104400635
AngularSpeedZ=0.14167786
LinearX=2.173782
LinearY=0.13265467
LinearZ=-0.1465683

Activity: Walking
```

Figure 3.3: The sensor data collection app (left) and the recognition app (right)

3.3 Feature Extraction

The changes of accelerometer and gyroscope values along each axis of the smartphone are monitored. A fixed window length of 64 data points for time-domain (256 data points for frequency-domain data) with a 50% overlap is used in the feature extraction stage. In the literature, features that can be extracted are summarized in [67].

3.3.1 Time-domain Features

For each window, some basic statistical characters can be calculated from raw sensor data.

- Mean. The mean value of each window along each axis.
- Max, Min. The maximum and minimum values of each window along each axis.
- Range. The difference between maximum and minimum values.
- Standard deviation, Variance. The standard deviation and variance of each window along each axis.
- Correlation coefficient. The correlation coefficient is calculated along each pair of axes of the sensor data.
- Signal Magnitude Area (SMA). SMA is calculated as the sum of the magnitude of each of the three-axis sensor signal (accelerometer or gyroscope) [68]:

$$SMA = \frac{1}{t}\left(\int_0^t |x(t)|dt + \int_0^t |y(t)|dt + \int_0^t |z(t)|dt\right) \tag{3.1}$$

- Signal Vector Magnitude (SVM). SVM can provide a measure of movement intensity, which is useful for fall detections [69]. It is defined as:

$$SVM = \frac{1}{n}\sum_{i=1}^{n} \sqrt{x_i^2 + y_i^2 + z_i^2} \tag{3.2}$$

3.3.2 Frequency-domain Features

Frequency-domain features have been widely used to capture the periodical nature of signals. A commonly used algorithm to extract frequency domain features is the Fast Fourier Transform (FFT). Typical frequency-domain features include:

- Energy. The spectral energy is calculated as the squared sum of discrete FFT component magnitudes.

- Entropy. The information entropy is calculated using the normalized information entropy of the discrete FFT component and it can help differentiate signals that have similar energy features [26].

- Time between peaks. Time between peaks in the sinusoidal waves [30].

- Binned distribution. It is computed as follows. First determine the range of values (maximum minimum) for each axis. Then divide this range into 10 equal size bins. Finally calculate the fraction of the values in each of the bins [30].

3.3.3 Feature Selection

In our study, the mean and standard deviation values in time-domain and energy and entropy in frequency-domain are extracted, which show satisfactory results. Some sensitivity analysis has been done and these features are deemed to be more sensitive and useful for this application: According to decision tree algorithm, these features selected above seem to be irrelevant without much loss of information, while the redundant features are removed during the decision tree generation. In addition, such features are relatively easy and efficient to be computed and extracted.

For each sensor, its three axes are regarded as three individual features. As two sensors are monitored and there are 2 time-domain features and 2 frequency-domain features for each axis as well as three axes for each sensor, a total of 24 features are extracted (shown in Table 3.1). The activity types are labeled manually in each file. The fixed sliding window size with 50% overlap has demonstrated success in [26].

Features from accelerometer and gyroscope sensors
∘ The mean value of X, Y, and Z-axis (window size 64, 50% overlap)
∘ Standard deviation of X, Y, and Z-axis (window size 64, 50% overlap)
∘ Energy of X, Y, and Z-axis (window size 256, 50% overlap)
∘ Entropy of X, Y, and Z-axis (window size 256, 50% overlap)

Table 3.1: Features extracted

3.4 Classification

A number of classifiers (Decision Tree, SVM, Naive Bayes, and Multilayer Perceptron) that have been reviewed in Section 2.1 are studied on the collected smartphone sensor data. The features extracted from the raw data can be either time-domain or frequency-domain features. To evaluate the effectiveness of different classifiers, cross validation is used as the evaluation method. More specifically, the 10-fold cross validation is performed in our tests. After feature extraction, the smartphone sensor dataset is divided into 10 subsets. One of the 10 subsets is used as the test set and the remaining nine subsets are put together to form a training set. This procedure repeats 10 times. Then the average performance across 10 trials is calculated. The performance metrics selected to evaluate the classification methods are precision and recall. Precision (P) is defined as the number of true positives (T_p) over the number of true positives and the number of false positives (F_p):

$$P = \frac{T_p}{T_p + F_p} \tag{3.3}$$

Recall (R) is defined as the number of true positives (T_p) over the number of true positives and the number of false negatives (F_n):

$$R = \frac{T_p}{T_p + F_n} \tag{3.4}$$

Accuracy is also used for comparing the classification results, which is defined as:

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{3.5}$$

3.5 Experimental Results and Analyses

In this section, four different classifiers (Decision Tree, Naive Bayes, SVM, and MLP) are tested to recognize activities. The performances and some observations are discussed below.

3.5.1 Experiment Setup

Using the apps described in Figure 3.3, tri-axial acceleration and gyroscope data (angular velocity in 3 axes) are collected and used for predicting activities like walking, running, sitting, going upstairs, and going downstairs. A smartphone is loosely placed in a person's pocket with fixed orientations. The user can also denote their height, weight, age, and gender for future reference. Then the smartphone's placement position (upward or downward) and the activity labels are recorded. This procedure repeats several times for each activity. The default sampling rate is 50000 microsecond (20 Hz). Figure 3.4 and Figure 3.5 show the raw accelerometer and gyroscope data of five activities concatenated together.

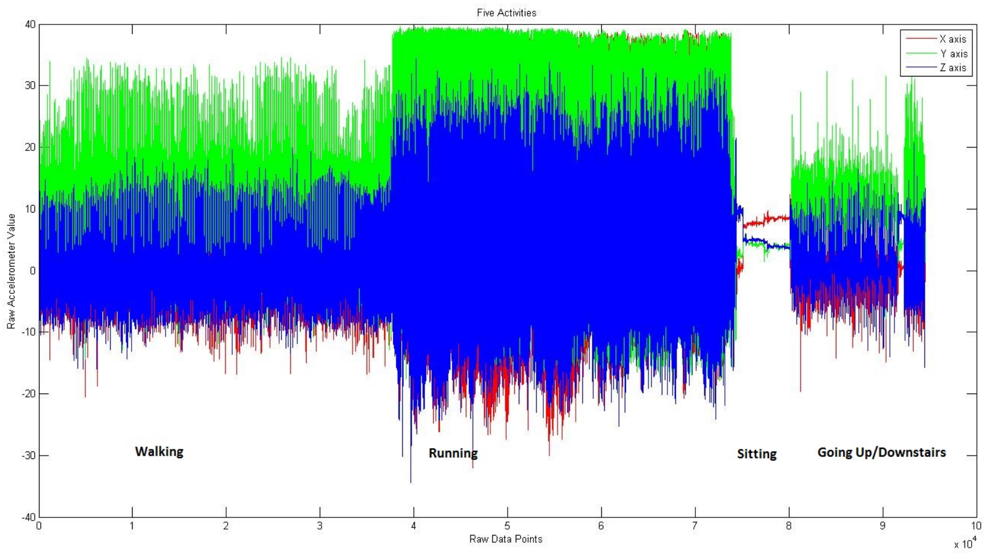


Figure 3.4: Raw acceleration data

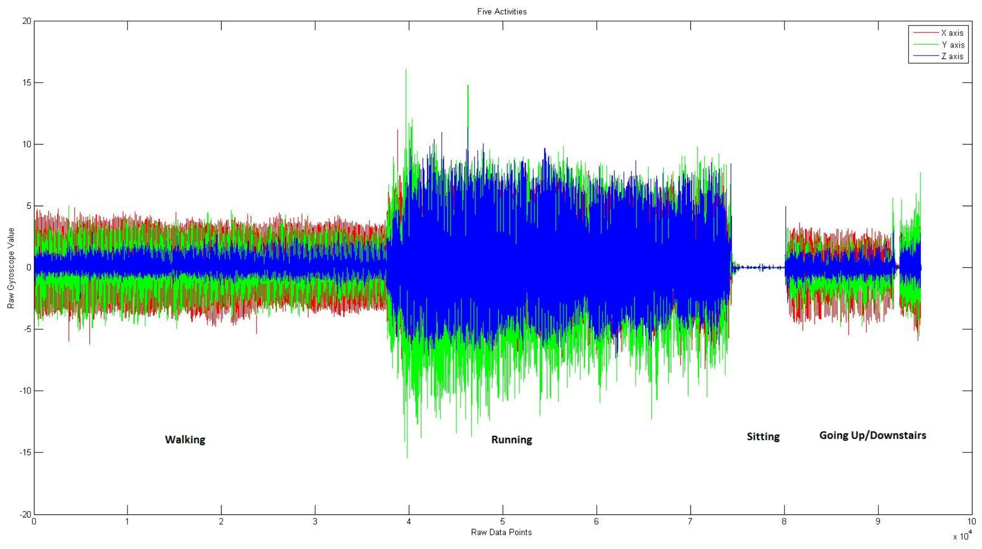


Figure 3.5: Raw gyroscope data

Time-domain and frequency-domain features are calculated individually. For the time-domain features, a window size of 64 with a 50% overlap is used, while the frequency domain features are derived with a window size of 256 samples. The training set is described in Table 3.2. A 10-fold cross-validation method is employed for testing and WEKA framework is used in our study.

Activity Type	**Number of Instances (in time domain)**	**Number of Instances (in frequency domain)**
Walking	1386	346
Running	1365	341
Sitting	1472	368
Going upstairs	784	196
Goding downstairs	487	122
Total	5494	1373

Table 3.2: The Training Set

3.5.2 Result Analyses

Table 3.3 reveals that all classifiers can achieve a recognition performance (in precision and recall) of 96%~100% for the dataset that contains three activities, walking, running, and sitting. However, the multilayer perceptron classifier takes the longest time to build a model, which is a disadvantage if we want to perform the training step in a smartphone. For the SVM classifier, parameter settings will significantly affect the system performance. When the sigmoid function is used as the kernel type in SVM, the accuracy is only 63.25%. The optimal SVM parameters for one problem may vary from one to another.

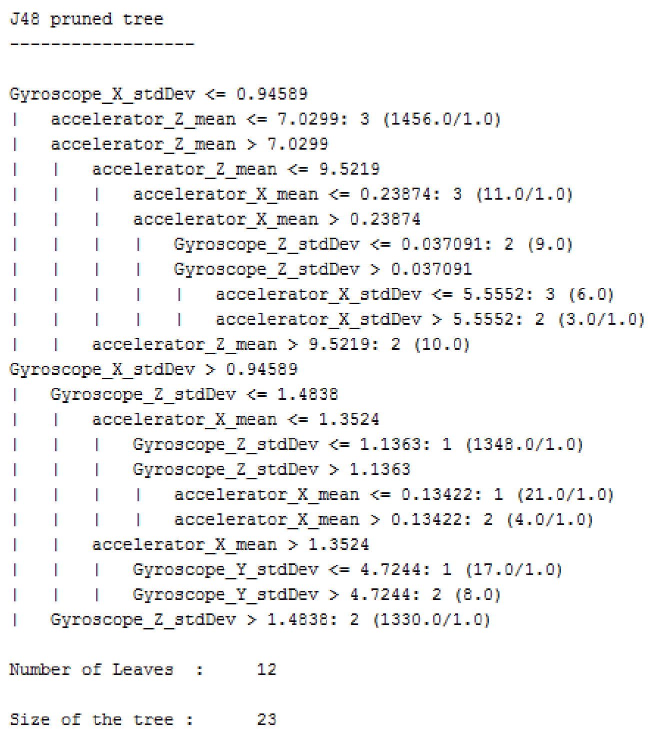
Decision Tree and Naive Bayes are both simple and efficient classifiers. In addition, J48 can generate a decision tree for identifying activities. From the decision tree, it is observed that not all 12 features are necessary for identifying activities. Figure 3.6 shows a generated J48 pruned tree. In this case, only 6 features (mean value and standard deviation of accelerometer in X axis, mean value of accelerometer in Z axis, and standard deviation of gyroscope in X, Y, and Z axis) are enough for detecting the three activities. What's more, for the gyroscope sensor, only the standard deviation values are needed according to the generated decision tree. This indicates that the standard deviation of a period of sensor data is more sensitive than the raw sensor readings when the smartphone is placed in pockets as it can tell us the intensity of different activities. The visualized data for different activities are shown in Figure 3.7 to Figure

		Walking	Running	Sitting	Weighted Average
Precision	Decision Tree	0.994	0.993	0.991	0.992
	SVM	1	0.897	1	0.967
	Naive Bayes	0.987	0.998	0.987	0.991
	Multilayer Perceptron	1	0.998	0.983	0.993
Recall	Decision Tree	0.995	0.987	0.995	0.992
	SVM	0.903	1	0.985	0.963
	Naive Bayes	0.999	0.982	0.99	0.991
	Multilayer Perceptron	0.998	0.983	0.998	0.993

Table 3.3: Classification Results Using 12 Features

3.16, which demonstrate the standard deviations in a period of time are the ideal features to identify activities. Figure 3.7 and Figure 3.8 plot the mean values of acceleration in all axes for walking and running. Y-axis values are around the same level (around $10m/s^2$ because of the gravity) while X and Z-axis values vary significantly that help distinguish the two activities. The direction of how the smartphone is placed in pocket and the coordination system defined in smartphone make the Y-axis irrelevant to different activities. Figure 3.11 plots the mean acceleration in three axes for sitting. It shows sitting has the minimum standard deviation around three axes. Going upstairs and downstairs (Figure 3.9 and Figure 3.10) are activities similar to walking but with larger variances.

Figure 3.12 to Figure 3.16 plot the standard deviation of angular velocity for all three axes of five activities. For walking and running, the standard deviation of x, y, and z vary differently. The standard deviations of sitting are almost zero as sitting is inactive activity that the smartphone changes very little. From the plots we can observe that standard deviations are more informative and sensitive than the real sensor readings and other statistical values.



```
J48 pruned tree
------------------

Gyroscope_X_stdDev <= 0.94589
|   accelerator_Z_mean <= 7.0299: 3 (1456.0/1.0)
|   accelerator_Z_mean > 7.0299
|   |   accelerator_Z_mean <= 9.5219
|   |   |   accelerator_X_mean <= 0.23874: 3 (11.0/1.0)
|   |   |   accelerator_X_mean > 0.23874
|   |   |   |   Gyroscope_Z_stdDev <= 0.037091: 2 (9.0)
|   |   |   |   Gyroscope_Z_stdDev > 0.037091
|   |   |   |   |   accelerator_X_stdDev <= 5.5552: 3 (6.0)
|   |   |   |   |   accelerator_X_stdDev > 5.5552: 2 (3.0/1.0)
|   |   accelerator_Z_mean > 9.5219: 2 (10.0)
Gyroscope_X_stdDev > 0.94589
|   Gyroscope_Z_stdDev <= 1.4838
|   |   accelerator_X_mean <= 1.3524
|   |   |   Gyroscope_Z_stdDev <= 1.1363: 1 (1348.0/1.0)
|   |   |   Gyroscope_Z_stdDev > 1.1363
|   |   |   |   accelerator_X_mean <= 0.13422: 1 (21.0/1.0)
|   |   |   |   accelerator_X_mean > 0.13422: 2 (4.0/1.0)
|   |   accelerator_X_mean > 1.3524
|   |   |   Gyroscope_Y_stdDev <= 4.7244: 1 (17.0/1.0)
|   |   |   Gyroscope_Y_stdDev > 4.7244: 2 (8.0)
|   Gyroscope_Z_stdDev > 1.4838: 2 (1330.0/1.0)

Number of Leaves  :     12

Size of the tree :      23
```

Figure 3.6: J48 decision tree (1 for walking, 2 for running, and 3 for sitting)

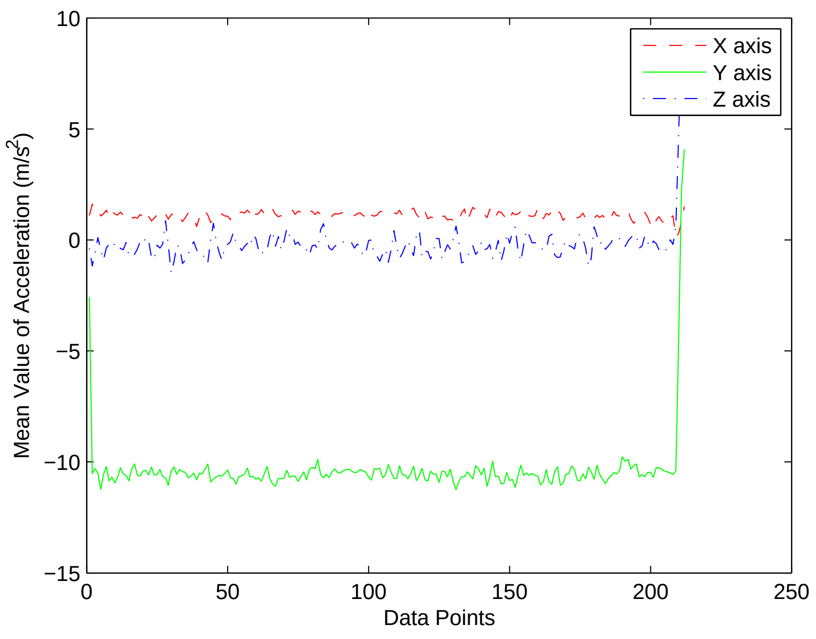


Figure 3.7: The mean values of accelerator data for Walking

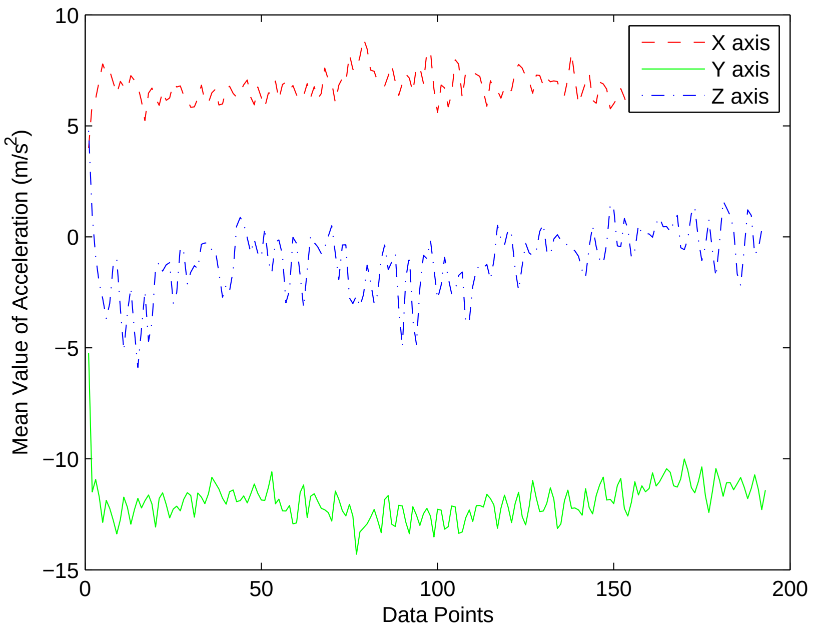


Figure 3.8: The mean values of accelerator data for Running

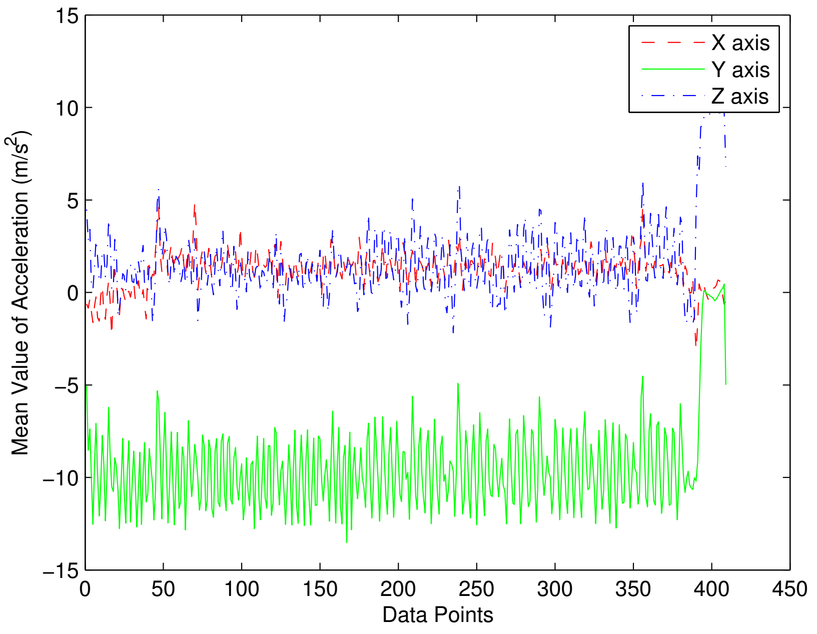


Figure 3.9: The mean values of accelerator data for going upstairs

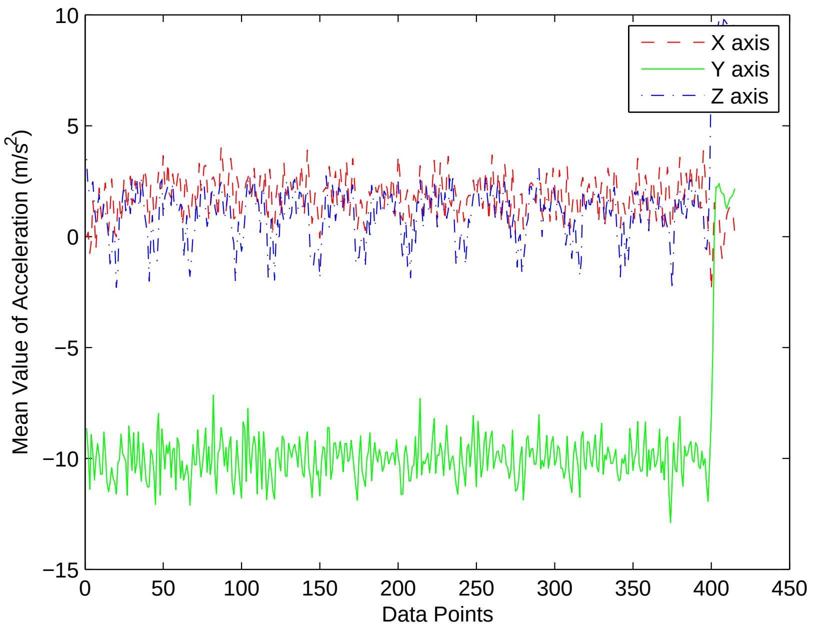


Figure 3.10: The mean values of accelerator data for going downstairs

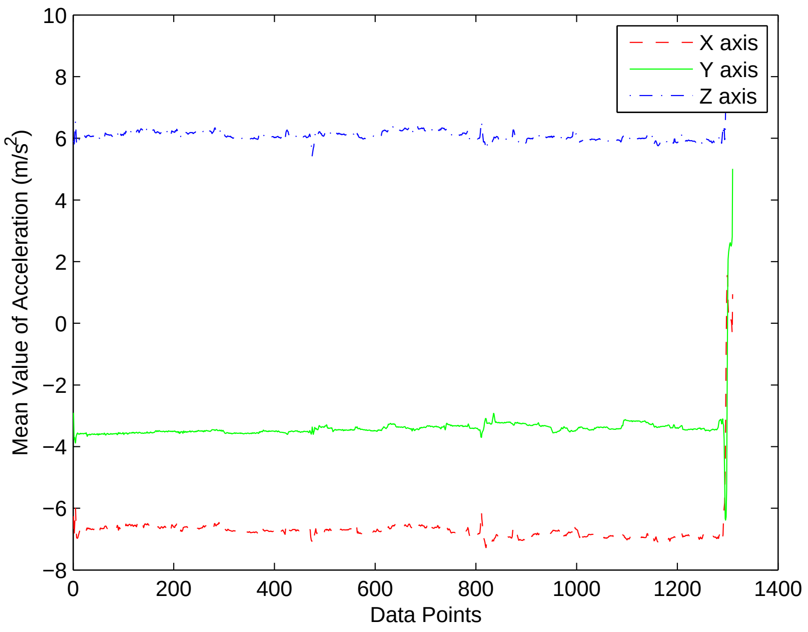


Figure 3.11: The mean values of accelerator data for Sitting

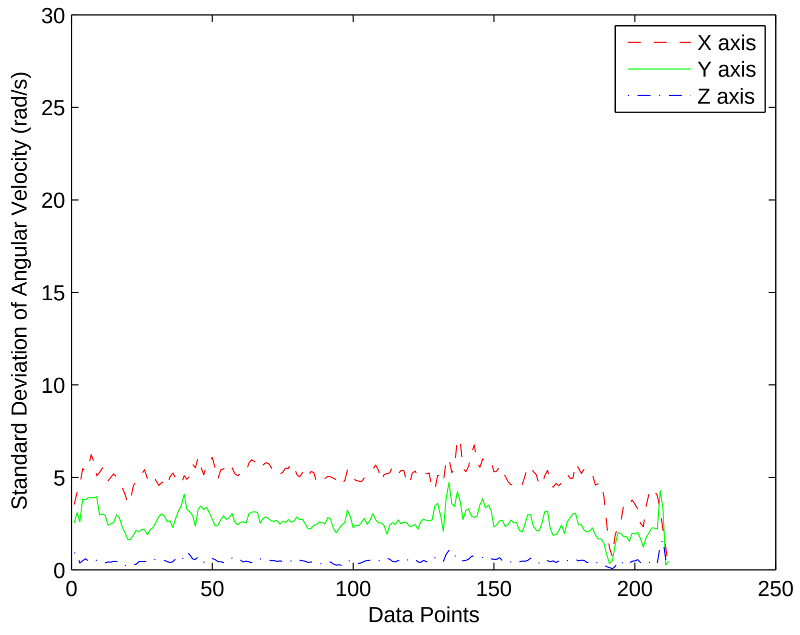


Figure 3.12: The standard deviation of angular velocity for Walking

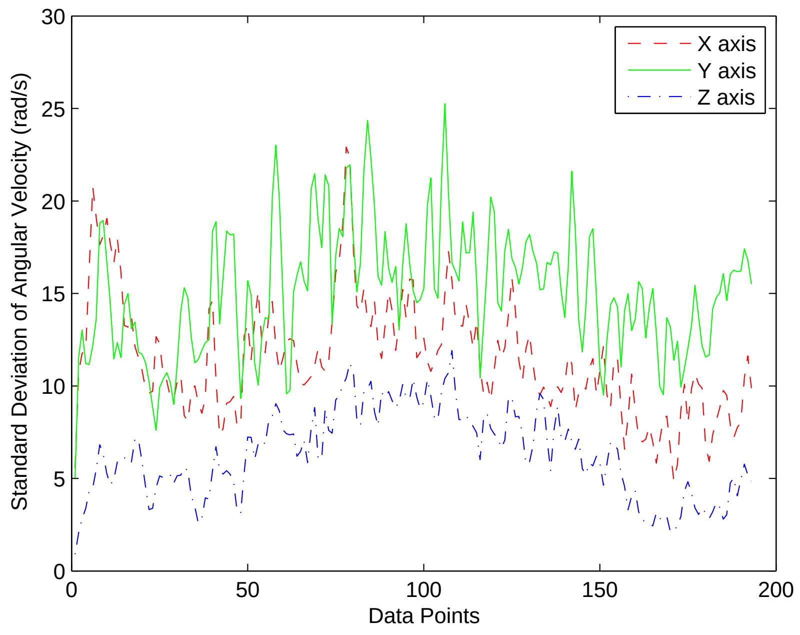


Figure 3.13: The standard deviation of angular velocity for Running

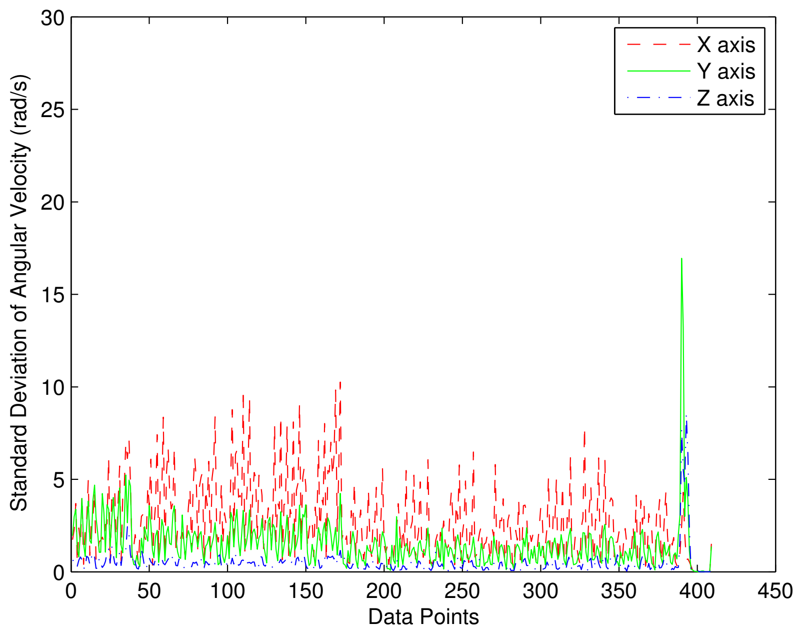


Figure 3.14: The standard deviation of angular velocity for going upstairs

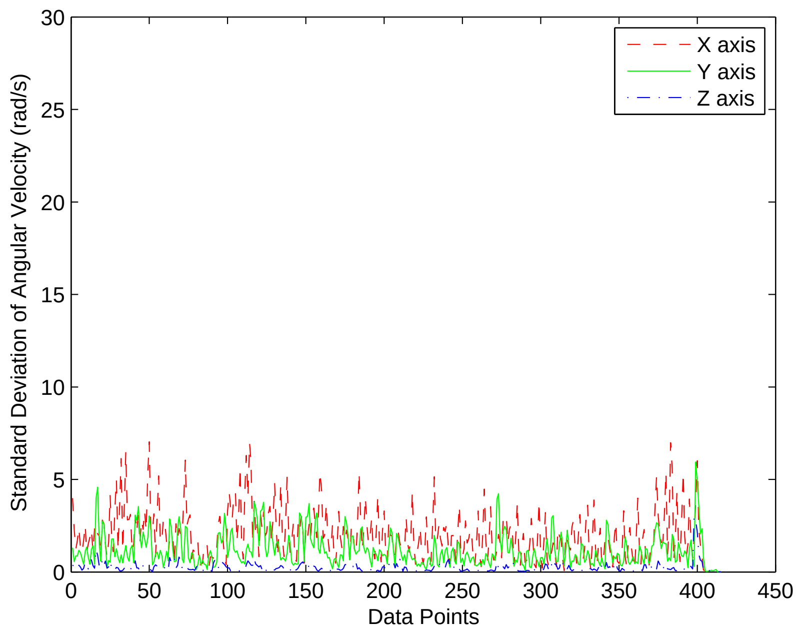


Figure 3.15: The standard deviation of angular velocity for going downstairs

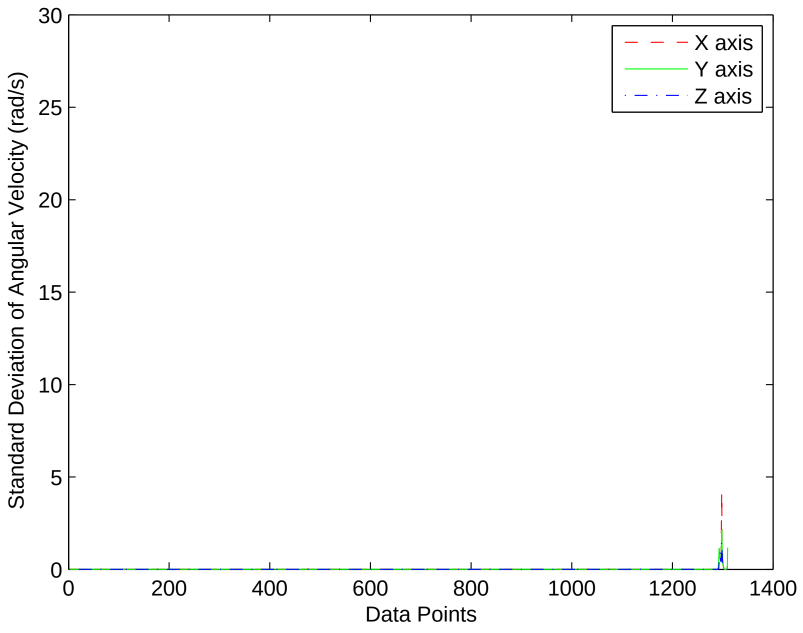


Figure 3.16: The standard deviation of angular velocity for Sitting

3.5.3 Extension

As three basic activities (walking, running, and sitting) can be well-recognized, the whole dataset is tested with two additional activities, going upstairs and downstairs. Using 12 time-domain features selected in Table 3.1, the performance of different machine learning algorithms for identifying the five activities is analyzed. Results are shown in Table 3.4.

Although they all have precision and recall scores over 89%, SVM classifier outperforms all other classifiers in identifying going upstairs and downstairs. The precisions for the Naive Bayes classifier to identify going upstairs and downstairs are only 75.8% and 62.2%, while SVM reaches 91.8% and 90.0%. It is believed that the performance of SVM can be further improved if its parameters are tuned.

Even if the C4.5 Decision Tree is an efficient classifier with low complexity in implementation, it has difficulties to model a large number of complex activities. What's worse, it can be easily confused while identifying walking, going upstairs and downstairs. As can be seen in Table 3.5, the confusion matrix shows that there is a large probability going downstairs is misclassified as walking or going upstairs.

Finally, the effectiveness of frequency-domain features is also examined. The testing results indicate that frequency-domain features (energy and entropy) can also be used for activity recognition and get satisfied results (as shown in Table 3.6). According the results in [26], a window size of 512 samples is suitable for fast computation of FFT components. If the sampling frequency is 50 Hz, each window represents 10.24s. The energy and entropy are extracted from the sliding windows for activity recognition. However, 10.24s may be too large for an activity recognition system to capture different activities. It has been proven that the interval of 1-2 seconds is the best trade-off between recognition accuracy and recognition speed [38].

Once the model has been trained, it can be used to detect different activities. However, the input of the detection model is only one data instance, and it is defective. In our experiments, although the precision and recall for identifying 3 activities reaches as high as 96%, the output can change quickly from one activity to another. To improve the robustness when using the trained model for real-time detection, one promising way is to consider the context information to determining the activity undertaken by a person.

		Walking	Running	Sitting	Going Upstairs	Going Downstairs	Weighted Average
Precision	Decision Tree	0.967	0.993	0.997	0.913	0.868	0.965
	SVM	0.977	0.771	0.998	0.918	0.9	0.916
	Naive Bayes	0.883	0.994	0.965	0.758	0.622	0.892
	Multilayer Perceptron	0.973	0.988	0.988	0.893	0.891	0.962
Recall	Decision Tree	0.967	0.997	0.993	0.897	0.891	0.965
	SVM	0.908	1	0.991	0.742	0.628	0.905
	Naive Bayes	0.903	0.979	0.99	0.583	0.791	0.89
	Multilayer Perceptron	0.978	0.984	0.98	0.935	0.84	0.962

Table 3.4: Classification Results Using 12 Features for 5 Activities

Classified As –>	Walking	Running	Sitting	Going Upstairs	Going Downstairs
Walking	1340	5	1	25	15
Running	0	1361	0	4	0
Sitting	0	2	1462	3	5
Going Upstairs	33	1	1	703	46
Going Downstairs	13	2	3	35	434

Table 3.5: Confusion Matrix of J48 Classifier

		Walking	Running	Sitting	Going Upstairs	Going Downstairs	Weighted Average
Precision	Decision Tree	0.935	0.991	0.992	0.867	0.836	0.949
	Naive Bayes	0.913	0.991	0.992	0.911	0.84	0.947
	Multilayer Perceptron	0.968	0.994	0.995	0.888	0.933	0.967
Recall	Decision Tree	0.954	0.991	0.986	0.867	0.836	0.949
	Naive Bayes	0.968	0.982	0.981	0.832	0.861	0.946
	Multilayer Perceptron	0.971	0.979	0.984	0.934	0.918	0.966

Table 3.6: Classification Results Using Frequency Domain Features

3.6 Summary

With the evolution of both software and hardware of smartphone devices, smartphone based human activity recognition systems take advantage of these advanced technologies. When compared with wearable devices, smartphones are pervasive with sensing functionalities integrated as well as power management, all of which make smartphones ideal for phone-based activity recognition to detect and predict our daily activities.

This chapter examine the data analysis and supervised learning approach for activity recognition model. As exhibited by the experiment results, the time-domain features (standard deviation and mean values) are efficient enough to recognize some basic human activities, which are also simple and practical to be extracted from smartphone sensor readings.

Experimental results also reveal that the proposed system can recognize simple human activities (walking, running, sitting, going upstairs, and going downstairs) with a satisfied accuracy (over 89%). Popular machine learning algorithms (C4.5 Decision Tree, SVM, Naive Bayes, and MLP) are tested, and among these options, Decision Tree is believed to be a better solution to be integrated into smartphone applications. As a decision tree model, it is easy to compute when implemented as a set of IF-THEN rules. Compared with time-domain features, although frequency-domain features can get satisfied classification results as well, extracting frequency-domain features are not practical in real applications due to the longer sampling intervals.

Chapter 4

Incremental Clustering for Human Activity Recognition

4.1 Introduction

It is necessary to introduce the incremental clustering for the current unsupervised activity recognition approach as a new application scenario. The reasons are as follows. First, in order to take advantage of unsupervised learning (we do not need to label data manually and will still be able to get comparable results to supervised methods.), the incremental clustering method is one step further than traditional clustering methods. By dealing with streamed sensor data collected by smartphone sensors, we do not need to get the entire dataset and can start clustering at any time. Also, as incremental clustering can be a pre-processing procedure for auto-annotated data collection, it would be a great help if we can process data at mobile devices instead of sending data to a remote server. Streaming and incremental techniques can meet the requirements such as memory and computing constraints of smartphones.

In our work, we incorporate both the streaming algorithm [64] and incrementally cluster-building procedure [66] to achieve the human activity detection purpose. The proposed system contains two stages. The first stage will process the streamed phone sensor data one by one and output the pre-clustered results. Then in the second stage, another incremental clustering procedure will be performed. With a user denoted number of human activities, the pre-cluster results will be refined and the final clusters formed. The advantages of our approach are that it can start and stop at any time with limited memory space and most importantly the second-stage incremental clustering (adding one cluster at one time) can be a compensation for the performance loss in the streaming stage.

4.2 Methodology

4.2.1 Data Collection

Experiments were conducted using smartphone embedded sensors (accelerometers and gyroscopes) to collect data describing five common activities (walking, running, sitting, going upstairs, and going downstairs). Each of the activities was performed for two to five minutes. The embedded accelerometers and gyroscopes would provide tri-axial acceleration and angular velocity at the sampling frequency of 20Hz. The sampling frequency is sufficient enough to capture daily physical activities. In our data collection app (as shown in Fig 3.3), personal information (without name and ID to protect privacy) is also recorded and a user chooses what kind of activities he/she is performing, which means each data point was labeled with the corresponding activity. Even though we were going to cluster the data later, such labelling information can help us evaluate the quality of clustering results. In addition, we assume the smartphone's orientation is fixed when in the subject's pocket.

4.2.2 Feature Extraction

The features were computed on a fixed window size of 64 consecutive data points with a 50% overlap. For each window, statistical features, including mean and standard deviation, were calculated. In addition, we also computed frequency domain features based on the Fast Fourier Transform (both the energy and entropy). However, due to our previous study on supervised learning methods, time domain features are good enough to get satisfied classification results. In this respect, we just omit the frequency domain features and use only the time domain features for clustering. In other words, for clustering purpose, our dataset consists of 12 features from the time domain (the mean and standard deviation of X, Y, and Z axis from the accelerometer and gyroscope).

4.2.3 Activities

As we choose five activities: walking, running, sitting, going upstairs, and going downstairs, it is possible to see how the patterns of acceleration and angular velocity differentiate for those activities.

Figure 4.1 shows the data visualization of the accelerometer's mean value. The X, Y, and Z axis in the coordinate system are the same as the three axes of the accelerometer. It is obvious from the figure that three activities, walking, running, and sitting, are easily separable. However, it is difficult to distinguish going upstairs and downstairs from walking as these three

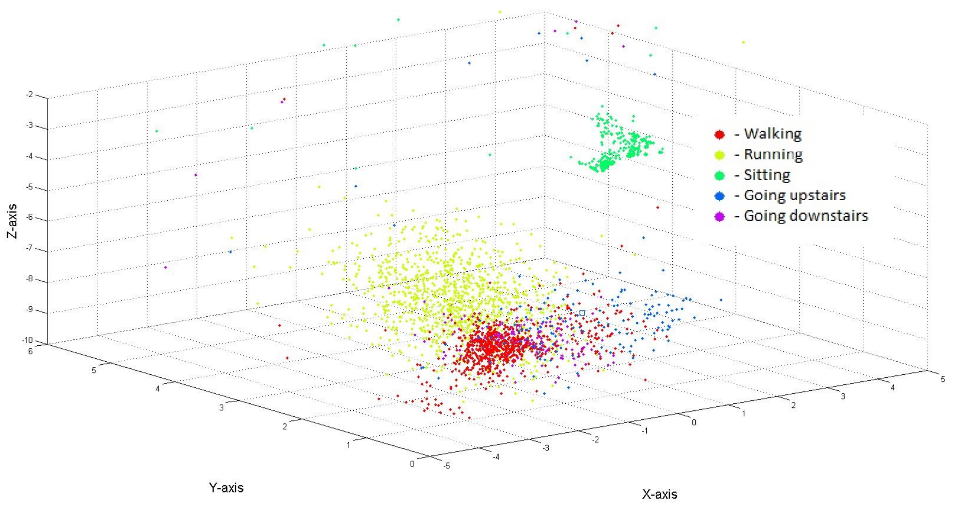


Figure 4.1: Data visualization for the mean value of accelerometer

activities overlap together in the coordinate space. The confusion matrix in Table 3.5 also shows the similarity of going upstairs and downstairs. The same observation can be obtained for the standard deviation of the accelerometer as shown in Figure 4.2.

4.2.4 The Proposed Two-stage Clustering Algorithm

In this section, we will give the description of the two-stage incremental clustering framework. All the clustering procedures are assumed to be done within the smartphone (as shown in Figure 4.3). The necessities of performing incremental clustering in mobile devices are as follows. If we want to cluster sensor data at the server side, we have to either transmit the whole dataset or the streamed sensor data to the server continuously, which would be a burden for both the server and mobile devices. What's more, in order to cluster data within the processing ability and memory constraints of mobile devices, a streaming manner is needed, which means we do not need to manipulate the whole dataset, on the contrary single data point can be processed from smartphone sensors using only limited memory space.

The two-stage algorithm is based on the traditional K-Means algorithm. It has three steps: 1) choose K points arbitrarily for the K initial cluster centers; 2) Assign every point in the dataset to the closet cluster and update the cluster centers accordingly; 3) If the cluster centers do not move, the algorithm terminates; otherwise, go to step 2. However, the traditional K-Means algorithm can only work on the entire dataset and will converge to a local minimum [66].

The first stage of our algorithm is a single pass streaming clustering algorithm modified

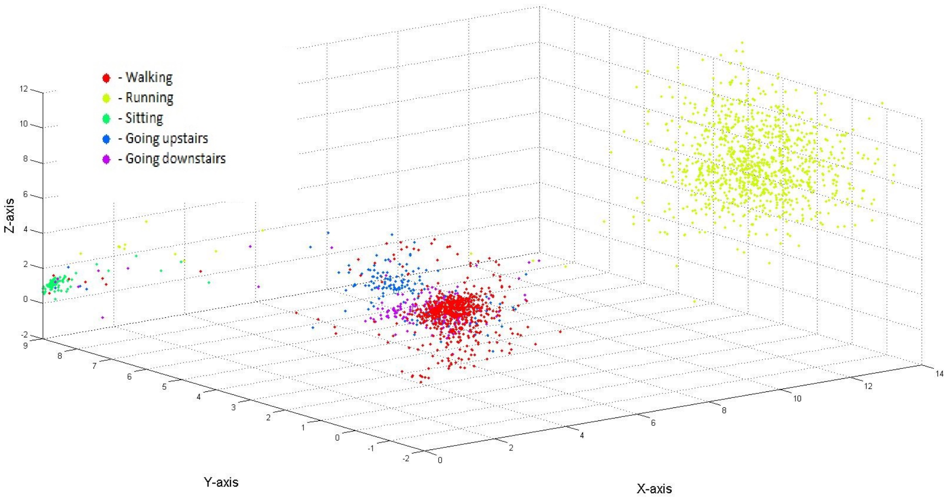


Figure 4.2: Data visualization for the standard deviation of accelerometer

from [64]. The flow chart of this algorithm is shown in Figure 4.4. First parameters are selected: the number of initial points m and the window size n for streaming. The algorithm then starts the initialization: choosing the first m points in the stream to be the initial candidate centroids. Assign weight $w_i = 1$ to each centroid. After the initialization, the incremental clustering is performed. For each sequent data point p in the stream, a counter is set up to track if we reach the window size. Then p is assigned to the nearest candidate centroid c_i, and the centroid is updated according to w_i and p. After updating, w_i is incremented by one. This procedure keeps going until the counter meets the preset window size. If the counter reaches n, all the candidate centroids in the list will be updated as follows: first a survival probability is calculated, which will be compared with a random number drawn uniformly from [0, 1]. Based on the comparison result, centroid c_i is either retained for the next n points or killed. After resetting all the weights to 1, the algorithm returns to the incremental procedure. The first stage continues running until there are no data points in the stream. Usually the initial number of clusters m is larger than the real number of activities. Thus the streaming stage will produce a pre-clustered result, which will be refined in the next stage.

The second stage is the incremental procedure, where the number of clusters K starts from one and is increased by one until a user defined value is reached. The first initial cluster center is randomly selected from pre-clustered data obtained in Stage 1. In each iteration, the data point with the greatest distortion I_i will be assigned to the closet cluster center and the current cluster centers in the list will be updated accordingly. After each iteration, if the cluster centers do not move and K have not reached the specified value, the number of clusters K will increase

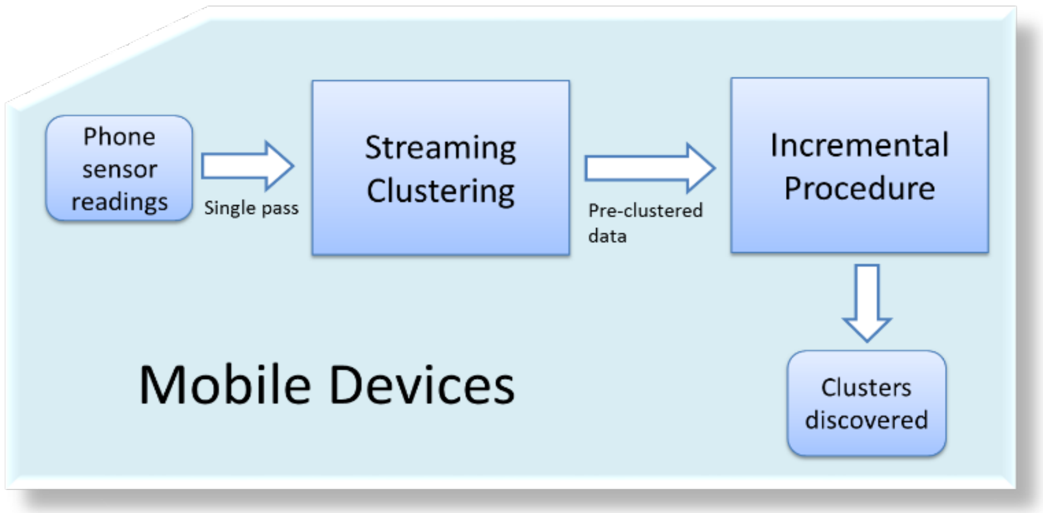


Figure 4.3: Proposed framework

by one and a new iteration starts. According to Pham et al. [66], adding cluster centers one by one will lead to a better clustering result and reduce the overall cluster distortion. The Euclidean distance is employed to measure distances between data points.

The distortion error of cluster z is defined as:

$$I_z = S_z - N[d(w_z, w_0)]^2 \tag{4.1}$$

Where N is the cluster's capacity (number of data instances belonging to the cluster), S_z is the sum of the squared distances between data instances in cluster z and the center of the Euclidean space, and w_z is the center of cluster z. A cluster C_i is characterized as a triple (w_i, N_i, S_i).

In order to illustrate why adding cluster centers one by one can lead to a better result, two operations are defined: removing cluster center C_i and moving a cluster to a new position. When removing a cluster center C_i is taken out. The worst case is that all instances belonging to C_i will be re-assigned to the second nearest cluster C_j without affecting any other clusters. This operation can be formulated as follows [66]:

$$N_k = N_i + N_j \tag{4.2}$$

$$w_k = \frac{1}{N_k}\sum_{i=1}^{N_k} x_i^k \tag{4.3}$$

$$S_k = S_i + S_j \tag{4.4}$$

Then the increase of the distortion ΔI can be derived as:

$$\begin{aligned}\Delta I &= I_k - I_i - I_j \\ &= \frac{N_i N_j}{N_i + N_j}[d(w_i, w_k)]^2\end{aligned} \tag{4.5}$$

When a cluster center is moved, the sum of cluster distortion errors decrease. If the cluster C_z is assumed to be a hyper-cube with uniform distribution density p, the following results can be obtained:

$$\begin{aligned}I_z &= \int_{-\frac{d}{2}}^{\frac{d}{2}}\int_{-\frac{d}{2}}^{\frac{d}{2}}...\int_{-\frac{d}{2}}^{\frac{d}{2}}[d(x, x_0)]^2 p dx^{(1)} dx^{(2)}...dx^{(N_d)} \\ &= \frac{N_z N_d d^2}{12}\end{aligned} \tag{4.6}$$

Then the decrease of the distortion errors is:

$$\begin{aligned}\Delta D &= \frac{N_{z1} N_{z2}}{N_z}[d(w_{z1}, w_{z2})]^2 \\ &= \frac{3 I_z}{4 N_d}\end{aligned} \tag{4.7}$$

If we first remove a cluster center and than insert a new cluster center, the sum of the distortion errors will be changed by a value $\Delta M = \Delta I - \Delta D$. It is obvious that $\Delta M < 0$ will lead to a better clustering result. For the extreme case, let $\Delta I = 0$, which means there is no removal operation. It equals to the fact that the number of initial clusters is set to 1 and increased by 1 in each iteration. This process repeats until a specific number of clusters is reached. The pseudocodes of the two stages are described in Procedure 1 and Procedure 2. The flow charts are shown in Figure 4.4 and Figure 4.5.

Procedure 1 First Stage Clustering

```
procedure FIRSTSTAGE($m, n$)    ▷ The number of initial points $m$ and the window size $n$
    Select $m$ points as $c_1, ..., c_m$    ▷ Initialize the candidate centers
    Assign weights $w_i$ to each of these centers.
    $Count \leftarrow 0$    ▷ Set up a counter
    for each sequent data point $p$ in the stream do
        $Count \leftarrow Count + 1$
        Find the nearest candidate center $c_i$ to the point $p$
        $c_i \leftarrow \frac{w_i \cdot c_i + p}{w_i + 1}$    ▷ Update the selected center
        $w_i \leftarrow w_i + 1$    ▷ Update the weight
        if $Count \mod n == 0$ then
            for each center $c_i$ in the list do
                $p_i \leftarrow \frac{w_i}{\sum_{i=1}^{m} w_i}$    ▷ Calculate survival probability
                Draw $\delta \sim U(0, 1)$
                if $p_i \geqslant \delta$ then
                    Retain the center $c_i$    ▷ Use it in the next $n$ points
                else
                    Kill the center $c_i$
                    Select a random point from the current window as $c_i$
                end if
                $w_i \leftarrow 1$    ▷ Reset the weight to 1
            end for
        end if
    end for
    return pre-clustered $m$ centers
end procedure
```

Procedure 2 Second Stage Clustering

```
 1: procedure SECONDSTAGE(pre-clustered data)
 2:     $K \leftarrow 1$                         ▷ The initial number of final clusters K is set to 1
 3:     checkPoint1:
 4:     if $K == 1$ then
 5:         Select a random point as the cluster center from the pre-clustered data
 6:     else
 7:         Select the point with the greatest distortion $I_i$ as the new center
 8:     end if
 9:     checkPoint2:
10:     Assign and update                        ▷ Similar with the traditional K-Means
11:     if the cluster center moves then
12:         goto checkPoint2
13:     else
14:         while $K$ hasn't reached a specified value do
15:             $K \leftarrow K + 1$
16:             goto checkPoint1
17:         end while
18:     end if
19:     return final $K$ cluster centers
20: end procedure
```

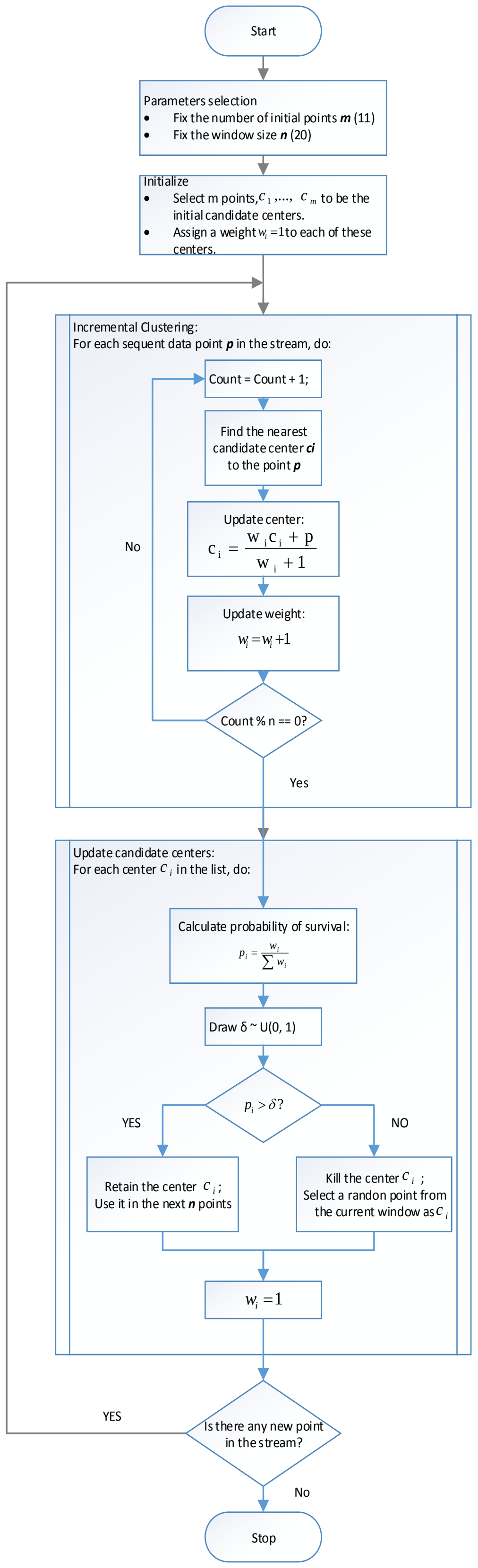

Start
Parameters selection
Fix the number of initial points m (11)
Fix the window size n (20)
Initialize
Select m points, $c_1, \dots, c_m$ to be the initial candidate centers.
Assign a weight $w_i = 1$ to each of these centers.
Incremental Clustering:
For each sequent data point p in the stream, do:
Count = Count + 1;
Find the nearest candidate center ci to the point p
Update center: $c_i = \frac{w_i c_i + p}{w_i + 1}$
Update weight: $w_i = w_i + 1$
Count % n == 0?
No
Yes
Update candidate centers:
For each center c_i in the list, do:
Calculate probability of survival: $p_i = \frac{w_i}{\sum w_i}$
Draw δ ~ U(0, 1)
$p_i > \delta$?
YES
NO
Retain the center c_i; Use it in the next n points
Kill the center c_i; Select a randon point from the current window as c_i
$w_i = 1$
Is there any new point in the stream?
YES
No
Stop

Figure 4.4: The flowchart of the first stage clustering

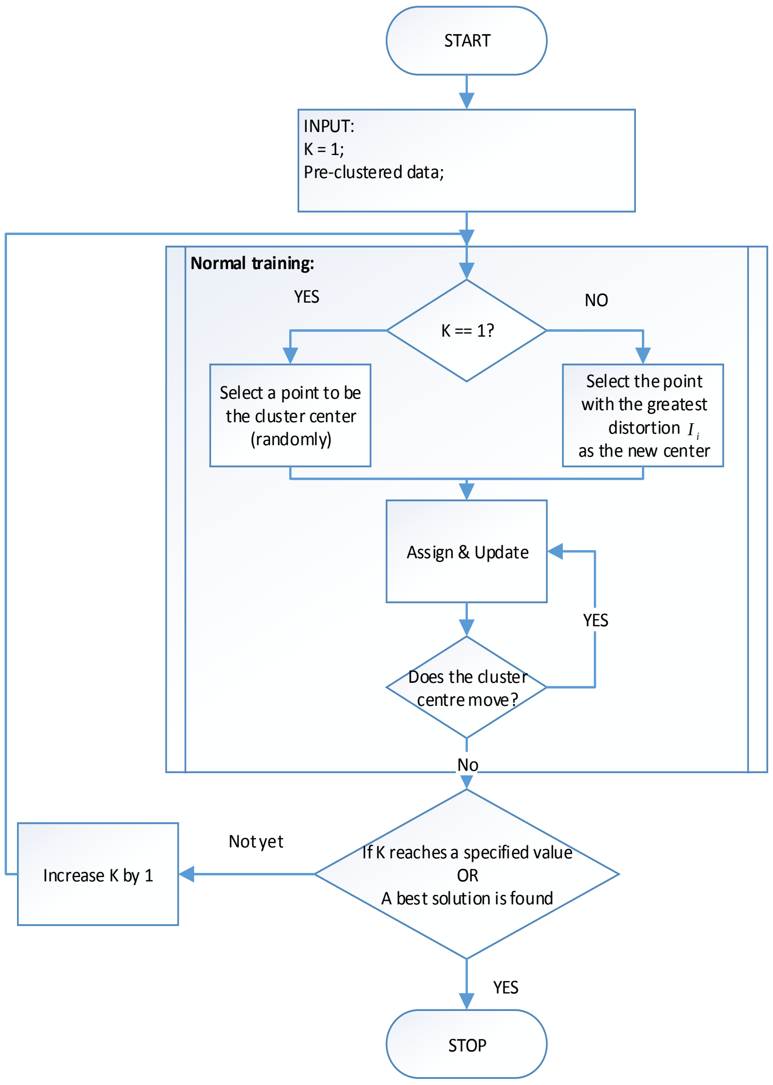

START
INPUT:
K = 1;
Pre-clustered data;
Normal training:
YES
K == 1?
NO
Select a point to be the cluster center (randomly)
Select the point with the greatest distortion I_i as the new center
Assign & Update
YES
Does the cluster centre move?
No
Not yet
Increase K by 1
If K reaches a specified value OR A best solution is found
YES
STOP

Figure 4.5: The flowchart of the second stage clustering

4.3 Experimental Results and Analyses

4.3.1 Incremental Clustering Results

The two-stage algorithm is implemented in the WEKA data mining framework [10] and tested on the dataset containing 2840 data points which are computed from raw sensor readings. The overview of the dataset can be seen in Table 4.1. In order to evaluate the clustering results, label information in the dataset is used and the classes to clusters evaluation is employed for the testing. A series of experiments were conducted with our collected sensor data. Details are discussed as follows.

First we got the baseline performance of traditional K-Means clustering, where K, the number of clusters, is set the same as the real number of activities (five daily activities: walking, running, sitting, going upstairs, and going downstairs). Then we evaluated the performance of the two-stage incremental algorithm with different parameter *m*, which is the number of initial number of points in the streaming procedure. In addition, the algorithm is compared with a variant produced without the second-stage incremental clustering (i.e., only using streaming procedure to produce the five activity clusters). The two-stage incremental algorithm is also compared with another simple sequential k-Means algorithm which is described in [70]. Based on a repetition of 10 times for each algorithm, the results are shown in Table 4.2.

Activity Type	**Number of Instances**
Walking	722
Running	1012
Sitting	807
Going upstairs	147
Going downstairs	152
Total	2840

Table 4.1: Dataset Overview

As it can be seen in the table, our two-stage incremental clustering algorithm can get comparable results with the traditional K-Means and reaches an accuracy over 72% on the testing dataset. The proposed two-stage incremental clustering algorithm also outperforms the sequential K-Means in [70]. Although K-means is simple, it outperforms GMM on human activity sensor data. What's more, if the second stage of our algorithm is removed, a significant performance loss is observed, which in return shows the effective of the second stage in compensating the performance drop in streaming stage. Such losses come from processing the streamed data

Methods	**Incorrectly Clustered Instances (%)**
K-Means	24.4542% ±7.4061%
Sequential K-Means	44.2807% ±10.6600%
Proposed Incremental Clustering (n=20, m=8)	27.7475% ±9.1320%
Streaming stage only	33.1843% ±7.5452%
Gaussian Mixture Model (GMM)	31.9648% ±4.6392%

Table 4.2: Performance

one by one and make it impossible to access the entire dataset.

4.3.2 Different numbers of initial points

The incorrectly clustered rate was examined when choosing different numbers of initial points m. As stated before, m is the number of clusters that obtained in the first stage and will be fed in to the second stage. The result is shown in Figure 4.6. The experiments reveal that the best number of initial points seems to be $m = 8$. When m is reduced to 5, the algorithm degrades into the streaming only situation. Larger values of m will not lead to a better clustering result.

4.3.3 Different window sizes

The effects of choosing different window sizes were compared. The comparison is shown in Figure 4.7, where m is fixed to 8 ($m = 8$). The best window size is around 20. Neither a larger nor a smaller window size would lead to a better performance.

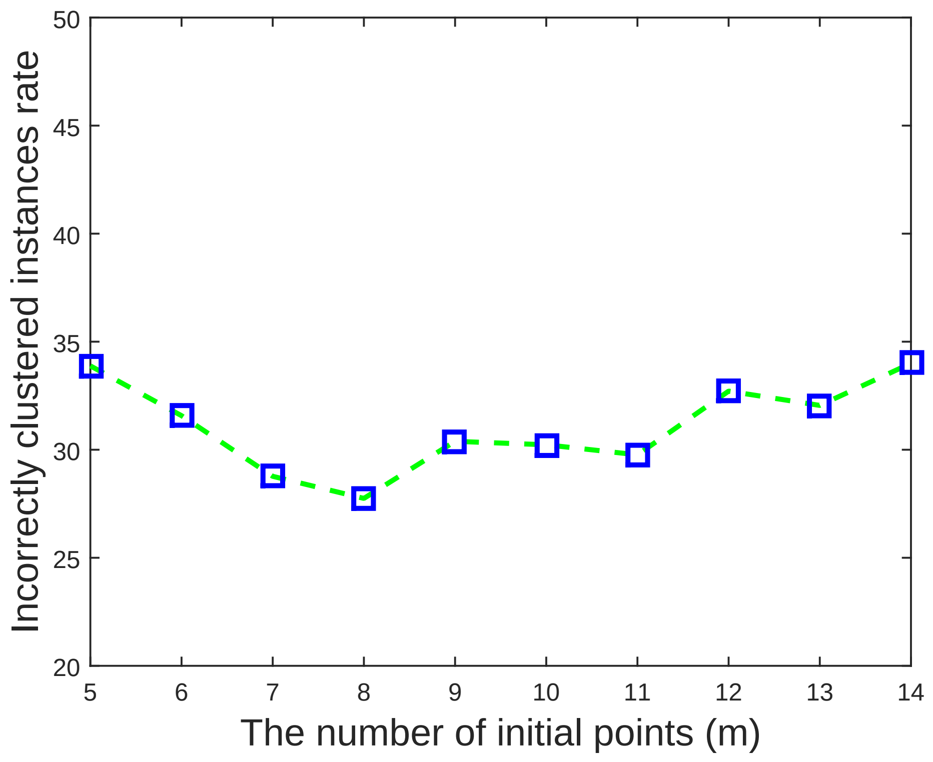


Figure 4.6: The effect of different m, when $n = 20$

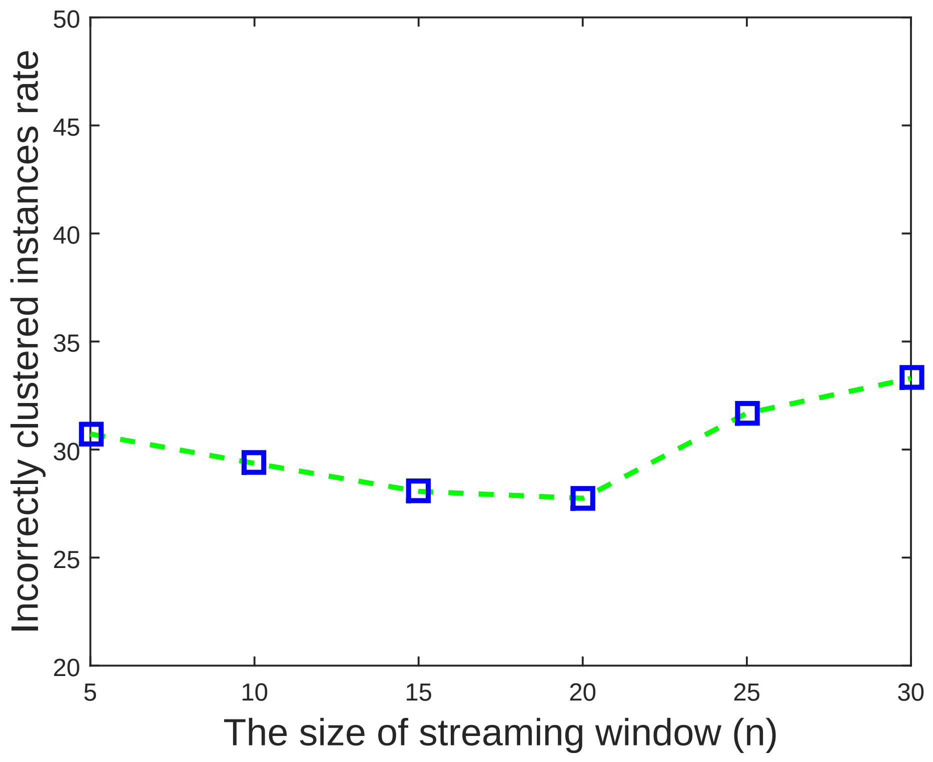


Figure 4.7: The effect of different window size, when $m = 8$

4.4 Summary

This chapter proposes a two-stage incremental clustering approach applied on the smartphone sensor based human activity recognition (HAR) system. The proposed method can deal with streamed phone sensor data and carry out the clustering within mobile devices. The first stage of the algorithm is a single pass clustering procedure, while the second stage will increase the final number of clusters one by one to reduce the overall distortion error. We also show the necessity in introducing such an incremental clustering method into human activity detection as a new scenario. First, the unsupervised learning approach is needed for activity detection as it can be used for auto-annotated data collection. Also, in order to process data on mobile devices within limited memory space and computation power, streaming technology will be a great help. The second stage incremental clustering can help alleviate the performance loss in the streaming stage. Our experiments show that the proposed algorithm can reach comparable results with traditional clustering algorithms but working in a streaming and incremental manner. And the effects of different number of initial points and window size are explored.

Chapter 5

Sensor Heterogeneity Effects

5.1 Introduction

Generally, an HAR system has two components: software and hardware. The hardware provides the foundation of the HAR system, especially the inertial sensors, which are indispensable. The software is incorporated with data processing techniques and recognition algorithms. Only with the robust hardware and reliable software, can an HAR system work to determine human activities. The hardware involves different kinds of devices that vary from customized wearable devices, smart watches, smart bands, and smartphones. In addition, inside these devices, sensors are made by different manufacturers with different specifications. Data collected by sensors will be sent to the software level through API calling and be used for activity recognition. The software treats the sensor readings as input and outputs activity types that the sensor data imply. The most important parts of the HAR software are how it deals with sensor data and how its algorithms are used for activity recognition. Recently, with the rapid development of machine learning techniques, the recognition methods have shifted from threshold methods to statistical learning approaches. Advanced machine learning algorithms are employed and tested in HAR systems, such as SVM, decision trees, neural networks. Even the most advanced deep learning algorithms have shown positive effects on sensor based human activity recognition [71].

However, the ultimate goal of HAR software is to generate a universal model that can work regardless of the hardware differentiation. Different mobile device vendors have their own HAR software and work on specific hardware platforms. For example, Samsung's fitness app S Health [72] is different from Sony's fitness APP Lifelog [73] in the UI design and inner function implementation. The same situation happens with Apple's Health app [74]. Such fitness apps can only work on specific smartphone modules, while a third party HAR software cannot meet user's demand of cross-platform activity recognition recording. Thus it is necessary and important to investigate whether the HAR software is flexible and irrespective with different hardware sensor performance indexes.

A successful HAR system can not only recognize different subjects' activities but also perform well on various hardware platforms. Here we identify two HAR system application scenarios:

- HAR system on multiple users: In the first scenario, an HAR system is capable of recognizing multiple users' activities. It may have an integrated model for multiple subjects. Due to the behavior differences of people with diverse physical characters, the activity features may vary slightly, which leads to a deviation of recognition results on multiple users. One possible solution to alleviate the multi-subject issue is to employ the users' physical characteristics such as weight and height to match the model prepared in advance. Such method proposed in [75] would not require the end user to collect and label his/her sensor data for training.

- HAR system across multiple devices: The second scenario is that an HAR system can recognize activities across diverse hardware platforms. This scenario is needed in HAR applications as a user can have multiple devices that record and recognize his/her activities. However, usually there is only one trained model for this user as he/she would not collect labeled sensor data and train activity data for all individual devices. One difficulty is that sensors from different manufacturers are often different in precision and resolution. Even if mobile device sensors come from the same manufacturer, due to the rotation error and misalignment relative to the circuit [50], data coming from these sensors may also have deviations. We will show and discuss in the following sections that such deviations harm HAR system performances.

Most researchers focus on the evaluation method that is often based on only one device. However it may not be applicable on other devices even though they have the same type of inertial sensors. This issue is not that serious when HAR applications only identify simple human activities, say running and walking. This is because with the help of embedded pedometer and GPS signals, it is easy to identify if a user is moving or still, and this is also how most existing applications work. When we go further and try to recognize more complex activities, including jogging, walking, going upstairs/downstairs, and even lifting, the sensor differences matter as multiple inertial sensors (accelerometer, gyroscope, etc.) are involved and subtle distinctions on sensor readings will influence what activity it is. We discovered that a dataset collected from one mobile device is not applicable on another mobile device. If we apply the trained model on other devices, a significant performance loss is observed. Further, it is impractical to collect sensor data for each device as it is time consuming and the labeling work is labor intensive.

In this chapter, the issue of sensor differences through multiple mobile devices are stressed. The causes of sensor differences such as sampling rate instability, the diversity of sensor calibration metrics and resolution range due to various sensor vendors are discussed. Then the effects of sensor differences based on a dataset collected from four smartphones in Scenario 2 identified above are reported. Various feature extraction and representation techniques and popular classifiers in the HAR research literature are evaluated. Some mitigating techniques are proposed and examined. Testing results for outlier removal, interpolation, and low-pass filter are presented. In addition, the performance of popular classifiers on tolerating sensor differences is investigated.

5.2 HAR on Multiple Devices

To conduct experiments on human activity recognition across multiple devices, a series of procedures to collect data from four different smartphone models are designed and performed. Features in time domain and frequency domain are selected based on the current HAR research. Then classifications are performed using some popular supervised machine learning techniques. In this stage, different evaluation methods are tested, comparing the performance of applying one dataset on another device as well as the performances of different classifiers in adapting the sensor differences.

5.2.1 Data Collection on Multiple Devices

To collect sensor data for the case study, one subject performs five simple activities using a sensor data collection app implemented in the Android platform. The five activities are: Walking, Running, Sitting, Going Upstairs, and Going Downstairs. The labels are preset by the user before performing activities. The length for each activity is about five minutes and the sampling rate is set to be 50000 microsecond (20 Hz). To reduce complexity and remove irrespective disturbances, the tested smartphone has two fixed positions when placed in the subject's pocket: "Upward" and "Downward". This procedure is repeated four times as there are four testing smartphones from different manufacturers (OnePlus One, Sony Xperia Z1 Compact, Samsung Galaxy S Infuse, and Motorola Nexus 6) (as shown in Table 5.1).

According to a previous study [26], the sampling rate for inertial sensor based activity recognition studies is sufficient when it is within 20 Hz to 50 Hz. In our experimental setting, the sampling frequency is fixed to 20Hz. The raw sensor readings are collected and stored,

Model	**Device ID**	**OS Version (Android)**	**Sensor Vendor & Model**	
			Accelerometer	**Gyroscope**
OnePlus One	A0001	5.1.1	STMicroelectronics LIS3DH	STMicroelectronics L3GD20
Samsung Galaxy S	SGH-I997R	2.3.3	STMicroelectronics K3DH	STMicroelectronics K3G
Nexus 6	shamu	6.0.1	Invensense	Invensense
Sony Z1C	M51w	4.4.2	BOSCH BMA2X2	BOSCH,BMG160

Table 5.1: Phone Sensors Information

which enables us to investigate various preprocessing and feature calculation techniques.

5.2.2 Feature Extraction

The smartphone embedded accelerometer and gyroscope data along each axis are collected. To extract time-domain features, a fixed window size of 64 data instances with a 50% overlap is set up in our study. Both the mean and standard deviation values along X, Y, and Z axis for each sensor are calculated based on the sliding window. Based on our investigation in Chapter 3, other time-domain features are not sensitive or significant and therefore are not considered. In addition, frequency-domain features (energy and entropy) are extracted (window size of 256 and 50% overlap).

5.2.3 Classification and Evaluation

In the classification stage, five popular supervised machine learning algorithms are tested: C4.5 decision tree, SVM, Random Forest, Nave Bayes, and Multilayer Perceptron. Each algorithm's performance is examined to check which classifier is relatively less affected by the differential sensor data.

In order to measure the performance of the classification results, the weighted average F-measure is employed, which is a better score than precision and recall. The weighted average F-measure considers both the precision and the recall:

$$WeightedAvg.F - Measure = \frac{\sum_{i=1}^{c} F_i - Measure}{\sum_{i=1}^{c} w_i} \tag{5.1}$$

Where the $F_i - Measure$ is the F-Measure of the *ith* class and the w_i is the number of instances of class i present in the test dataset. The $F - Measure$ is defined as:

$$F - Measure = \frac{2 * precision * recall}{precision + recall} \tag{5.2}$$

To evaluate the impacts of different accelerometer and gyroscope sensors on human activity recognition systems, three testing modes are identified and compared to each other:

1. 10-fold cross validation: As described in Section 3.4, for the cross validation, first the dataset is broken into 10 subsets equally. Then training is performed on 9 subsets and testing on the remaining subset. This procedure repeats 10 times to obtain the average score.

2. Device-to-device validation: As we have four distinct smartphones in our study, we train the dataset from one smartphone, and test the model on the other three smartphones. Device-to-device validation straightly assesses the impact of different sensors of smartphones.

3. Leave-one-out validation: In this mode training is performed on the dataset synthesized from all available smartphone models, except for one, which is retained for testing only. This mode considers the real-world HAR system deployment scenario: the training is done based on various different smartphone modes, while the testing is done on one unknown smartphone type.

5.2.4 Results on Original Datasets

The evaluation is performed on the datasets described above using the three evaluation modes. Both time-domain and frequency-domain features are examined. An example of a validation matrix for Random Forest is shown in Table 5.2. The bold diagonal represents the results for 10-fold cross validation. Each smartphone type is trained and tested on itself with a cross validation method. The row 'leave-one' represents the leave-one-out validation. Figure 5.1 and Figure 5.2 show the performances of different validation modes on all classifiers.

	Testing Set			
Training Set	A0001	M51w	SGH-i997R	shamu
A0001	**0.981**	0.672	0.556	0.781
M51w	0.641	**0.972**	0.732	0.653
SGH-i997R	0.798	0.828	**0.979**	0.79
shamu	0.803	0.762	0.873	**0.975**
Leave-one-out	0.828	0.889	0.811	0.906

Table 5.2: Evaluation Matrix on Original Datasets for Random Forest (in F-Measure)

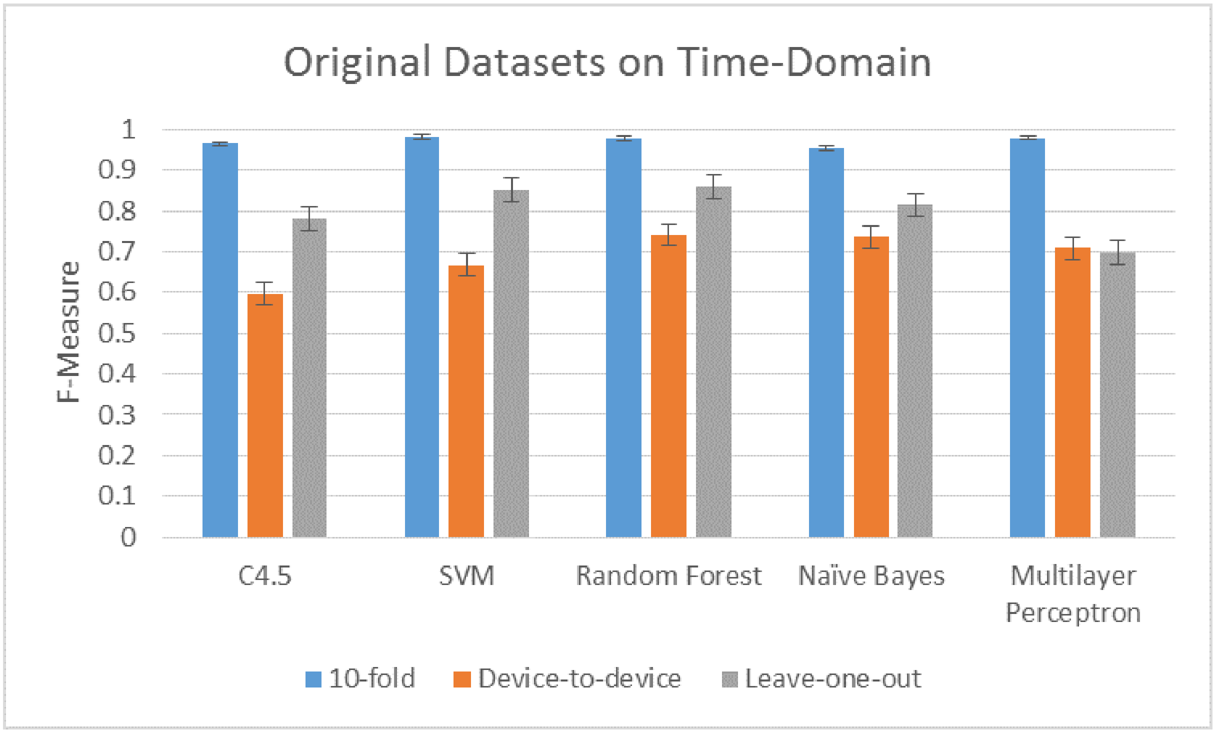


Figure 5.1: The performance evaluation on time-domain features

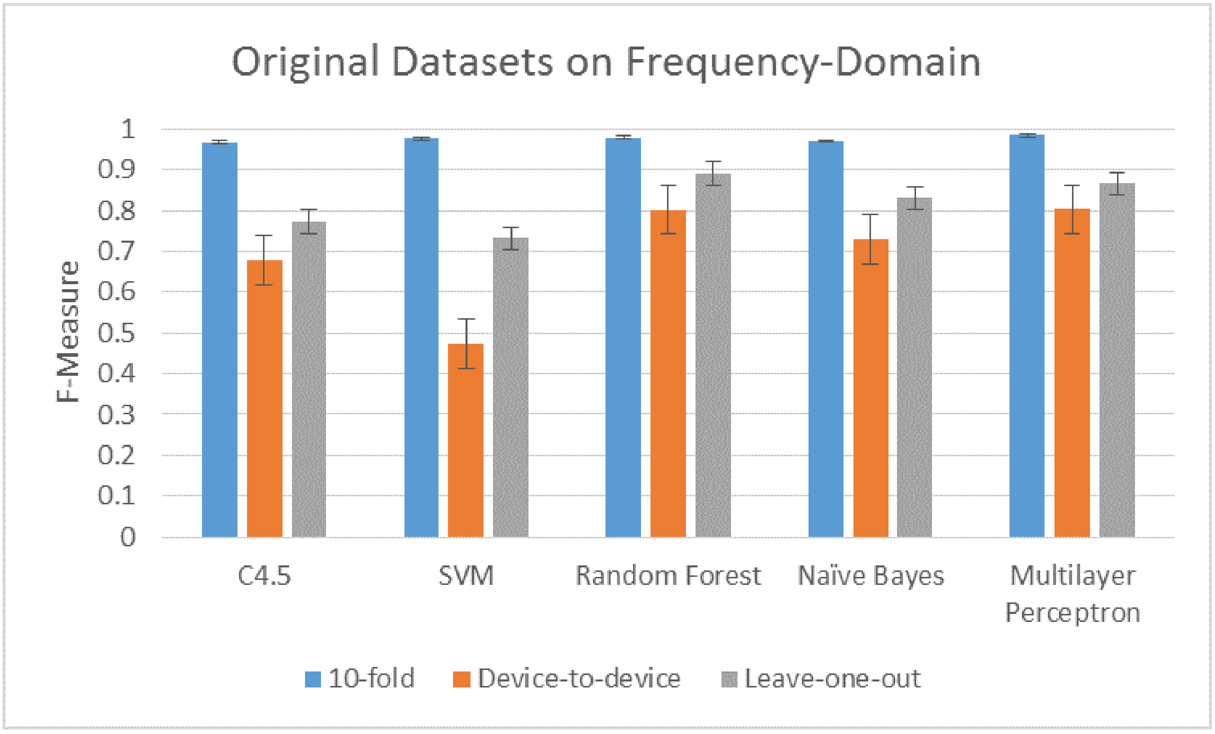


Figure 5.2: The performance evaluation on frequency-domain features

As it can be seen in both Figure 5.1 and Figure 5.2, with the 10-fold cross validation mode, all classifiers reach a nearly perfect performance, which is also reported in many HAR researches. However, the other two validation modes provide a more realistic performance evaluation. In the device-to-device evaluation, the testing is done on a different smartphone type from the training smartphone type carried out by the same subject. It indicates a large degradation of the HAR system's performance. The performance of leave-one-out validation is better than the device-to-device validation as it integrates all the available smartphone models except the testing one. In the real-world scenario, the leave-one-out method seems to be a possible metrics for measuring the performance of HAR systems. But it still takes a lot of efforts and requires the user to collect as much data as possible on various distinct devices.

5.3 Mitigating Techniques for Sensor Differences

In this section, the issue of sensor differences based on the testing results in Section 5.2.4 is stressed. First of all, the sensor deviations along three axes and various smartphone models are demonstrated. Then the reasons causing such performance degradation are discussed. Some mitigating techniques are proposed and investigated in this section.

5.3.1 Sensor Differences

When assessing the impacts of sensor differences on HAR systems, according to the data analyses on the sensor data collected from different smartphone models, significant heterogeneities among those sensor readings are found, which will greatly impair the performance of human activity systems.

According to Stisen et al. [50], the sensor differences can be categorized as follows:

1) Sensor Biases
   Smartphone embedded accelerometer and gyroscope sensors have different specifications in precision, resolution, and range, which also generate various biases. Although initial calibration is done by the manufacturers, considering gains and offsets on each of the three sensor axes, errors can still exist due to the rotation of the accelerometer package relative to the circuit board and the errors in soldering and assembly process. Unexpected shocks like falls on the ground may also make the device sensor misaligned and cause unwanted biases.

2) Sampling Rate Heterogeneity
   Smartphones from different manufacturers use various sensor models that support distinct maximum sampling rates. Training HAR systems individually for each sampling rate is impractical as the ground truth data collection is time consuming and costly.

3) Sampling Rate Instability
   One main reason for the sampling rate instability is the multitasking effect. As smartphones support running a large number of applications, this may cause high CPU loads during periods of time. The Android operating system often gives different priorities among currently running tasks and this may affect the sensor's sampling rate for HAR applications running on the device. A high CPU load impacts actual sampling rates and the instability varies differently across devices models even if they run the same OS version and the same concurrent programs.

To demonstrate the sensor biases, we draw the box plots of acceleration and gyroscope magnitudes for the investigated smartphones (shown in Figure 5.3 and Figure 5.4).

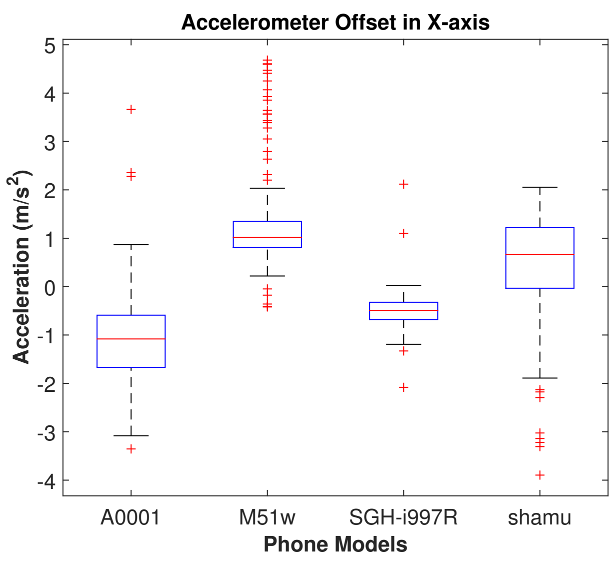


Figure 5.3: The box plots of acceleration offset in X-axis

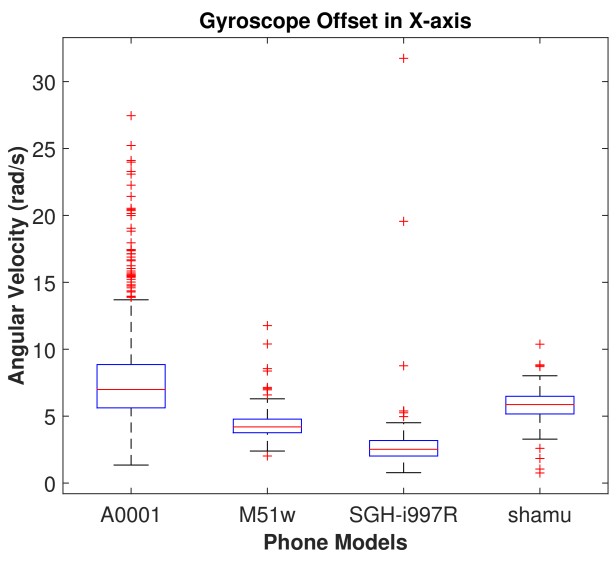


Figure 5.4: The box plots of gyroscope offset in X-axis

The box plots in Figure 5.3 and Figure 5.4 show examples of sensor biases. Labels in X-axis are four different sensor models embedded in four different smartphones. The Y-axis represents the value of accelerometer and gyroscope readings for walking. As it can be seen in the picture, four sensors have different mean values and different quartile range as well as different whisker length. The red-cross marks in Figure 5.3 and Figure 5.4 indicate the outliers in the dataset. Points are drawn as outliers if they are larger than q3+w(q3-q1) or smaller than q1-w(q3-q1), where q1 and q3 are the 25th and 75th percentiles, respectively, and w is the maximum whisker length. The default w is 1.5 corresponding to approximately 2.7 and 99.3% coverage if the data are normally distributed. All these sensor biases would significantly lower the performance of human activity detection system.

5.3.2 Possible Mitigating Techniques

Three possible mitigating techniques to alleviate the sensor differences are investigated and assessed: outlier removal, interpolation, and a simple low pass filter (LPF). All these techniques are applied in the time domain at the data preprocessing stage. Their effectivenesses are examined and discussed.

1) Outlier Removal

   As indicated in Figure 5.3 and Figure 5.4, the red-cross marks are the outliers present in the datasets. These outliers from raw sensor readings are supposed to be generated by the sampling instability, which means in the smartphone's runtime, some unwanted extreme values will be recorded. It is believed that if these outliers are removed, the impairments caused by sensor differences will diminish.

   In this study, the outliers are removed using the formula $Q3 + OF * IQR < xQ3 + EVF * IQ$ to detect outliers, where $Q3$ is the 25% quartile, IQR is the interquartile range, OF is the outlier factor (equals to 3.0), and EVF is the extreme value factor (equals to 6.0).

2) Interpolation

   As our sampling frequency 20Hz is a relatively low rate in HAR studies. To examine the effects of higher frequency the original data are interpolated to 50 Hz via up-sampling. The interpolation technique used is the linear interpolation, which employ linear polynomials to build new data instance between two adjacent input instances.

3) Low-Pass Filter (LPF)

   A low-pass filter is also used to smooth the raw sensor data in order to eliminate noisy scattered data instances. The parameter in the LPF is set to be 0.7.

The starting point of employing these candidate techniques is to alleviate the sensor data differences caused by the sensor biases and sampling instability. Results are shown in Figure 5.5 to Figure 5.12.

5.3.3 Result Analyses and Discussions

Figure 5.5, Figure 5.6, and Figure 5.7 are compared with Figure 5.1. Figure 5.5 to Figure 5.7 show the results for our three candidate solutions. Figure 5.8 to Figure 5.12 show the details for each of the five classifiers.

Clearly, the outlier removal can improve the HAR performance for almost all classifiers and validation methods. It is especially efficient for the SVM, Random Forest, and multilayer perceptron classifiers. Take the Random Forest classifier as an example, the F-score of 10-fold cross validation after removing outliers is 0.98525, which is higher than the original 0.97675. For the device-to-device validation, the F-score is 0.69058, which is 16.5% higher than original one (0.5972). For the leave-one-out validation, the F-score after removing outliers is also better than original score (0.88875 vs. 0.84975).

However, up-sampling linear interpolation cannot improve the F-score. It even impairs the HAR performance as shown in Figure 5.8 to Figure 5.12. The reason for the impairment may be that just simply interpolating raw sensor data would escalate the sensor differences.

The LPF, for most cases, is as efficient of a method as the outlier removal. In the case of Random Forest, the F-score of 10-fold cross validation after removing outliers is 0.9865, higher than the original F-score. For the device-to-device validation, the F-score of LPF is 0.7445, nearly the same as the original 0.74075. For the leave-one-out validation, after processing data using filters, the F-score is also better than original score (0.8835 vs. 0.8585).

To conclude, both outlier removal and LPF have positive effects in alleviating the sensor differences among various smartphone models. But interpolation in time-domain features will not improve and may even harm the HAR performance. In addition, the performance of leave-one-out validation is higher than the device-to-device validation. However, in the real-world scenario, it is impractical to collect and train data from various devices for each user, while testing is done on an unknown device, especially for the situation of the fast evolution of smartphone hardwares. Therefore the mitigating techniques should focus on the device-to-device validation scenario.

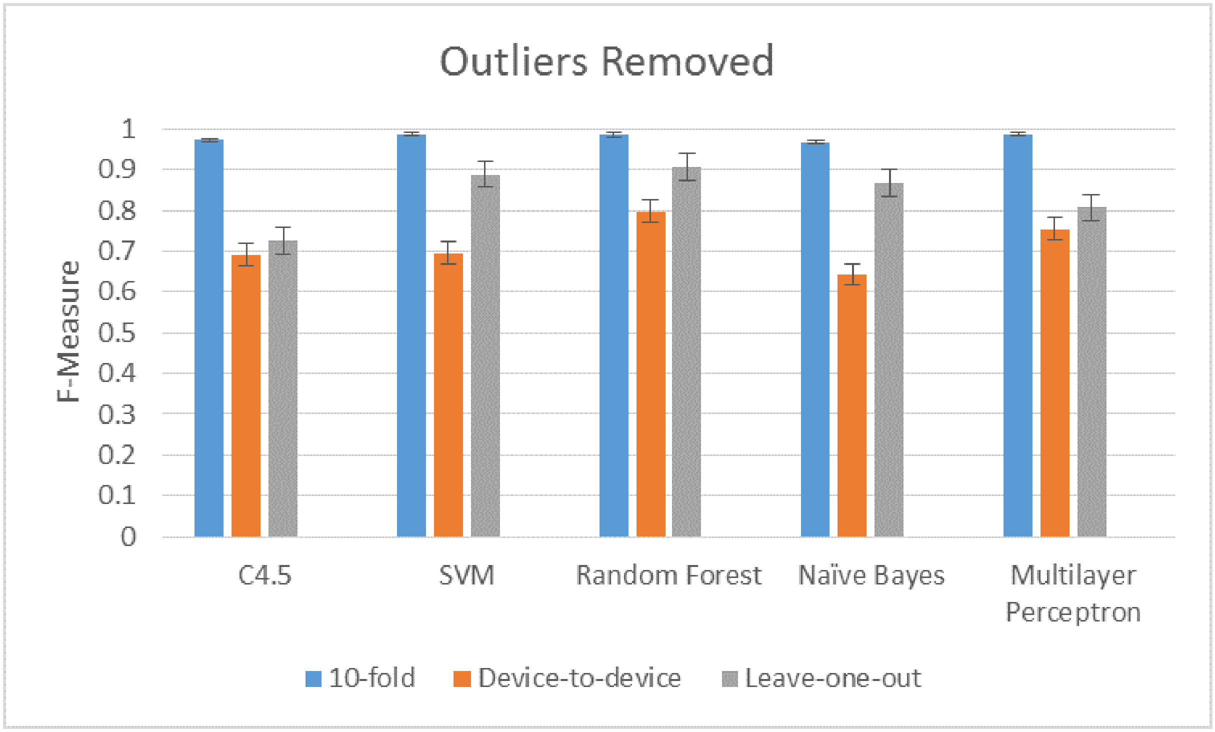


Figure 5.5: Activity recognition performance using outlier removal

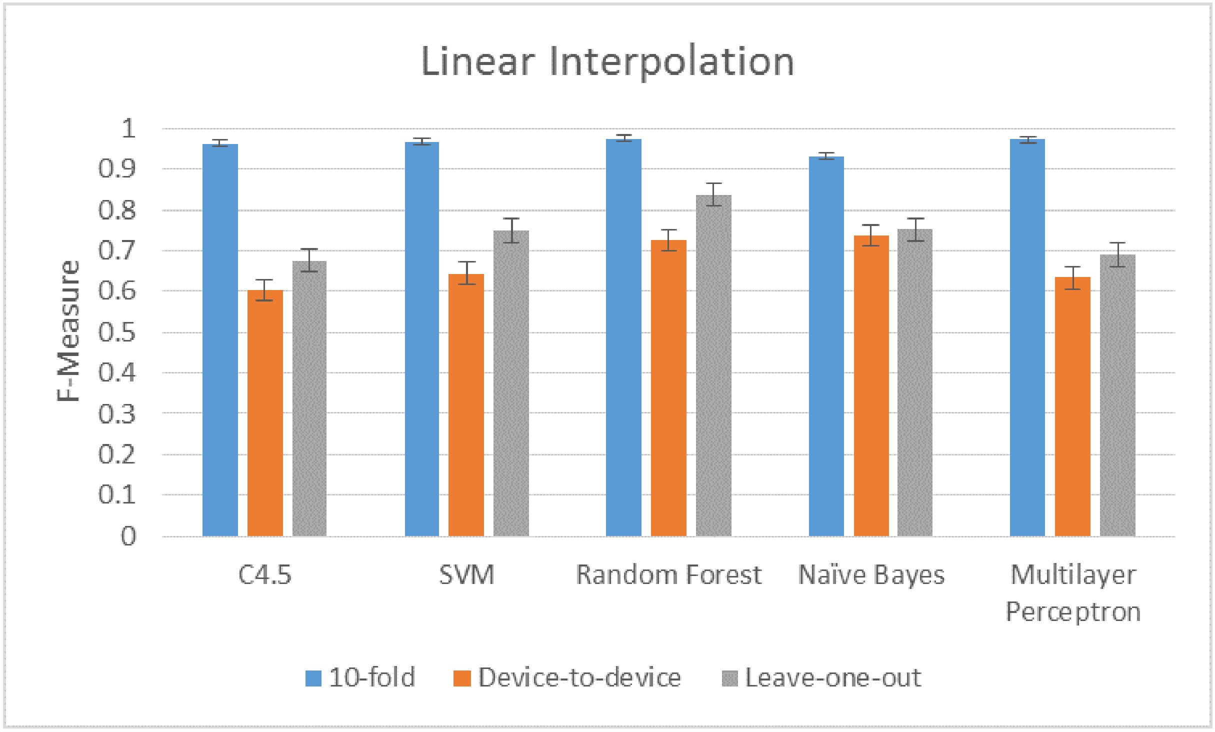


Figure 5.6: Activity recognition performance using linear interpolation

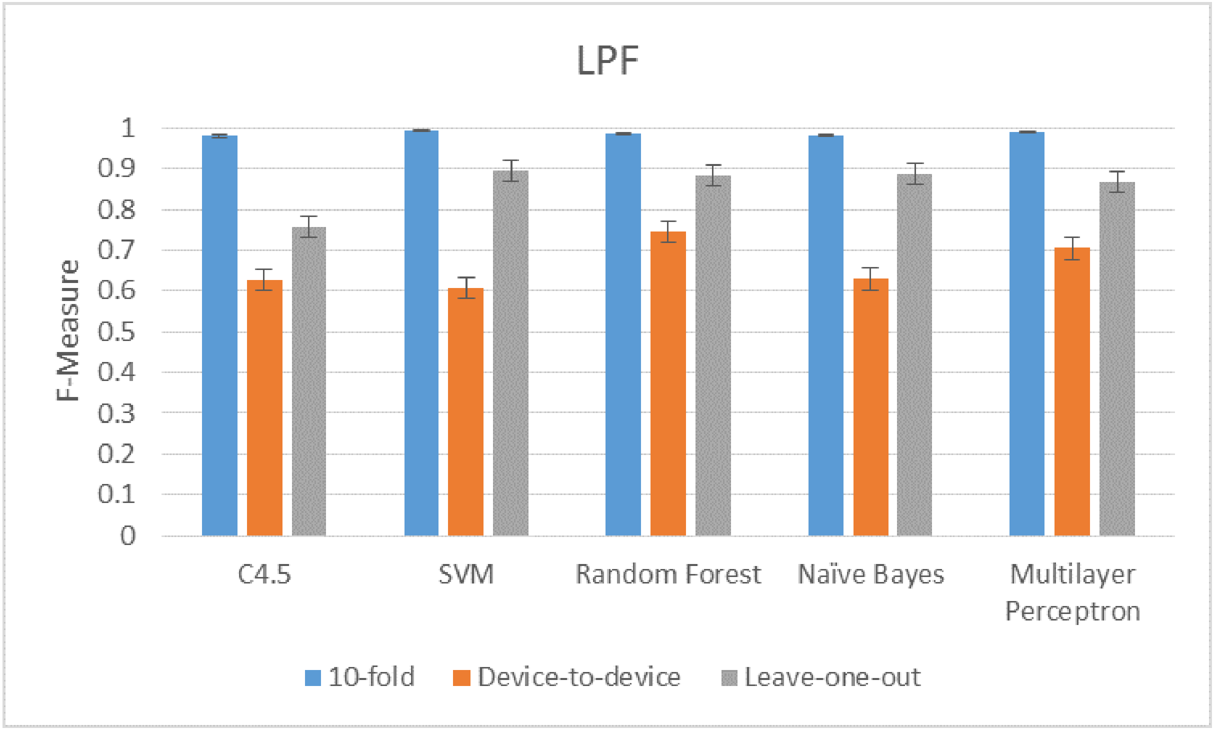


Figure 5.7: Activity recognition performance using low-pass filter

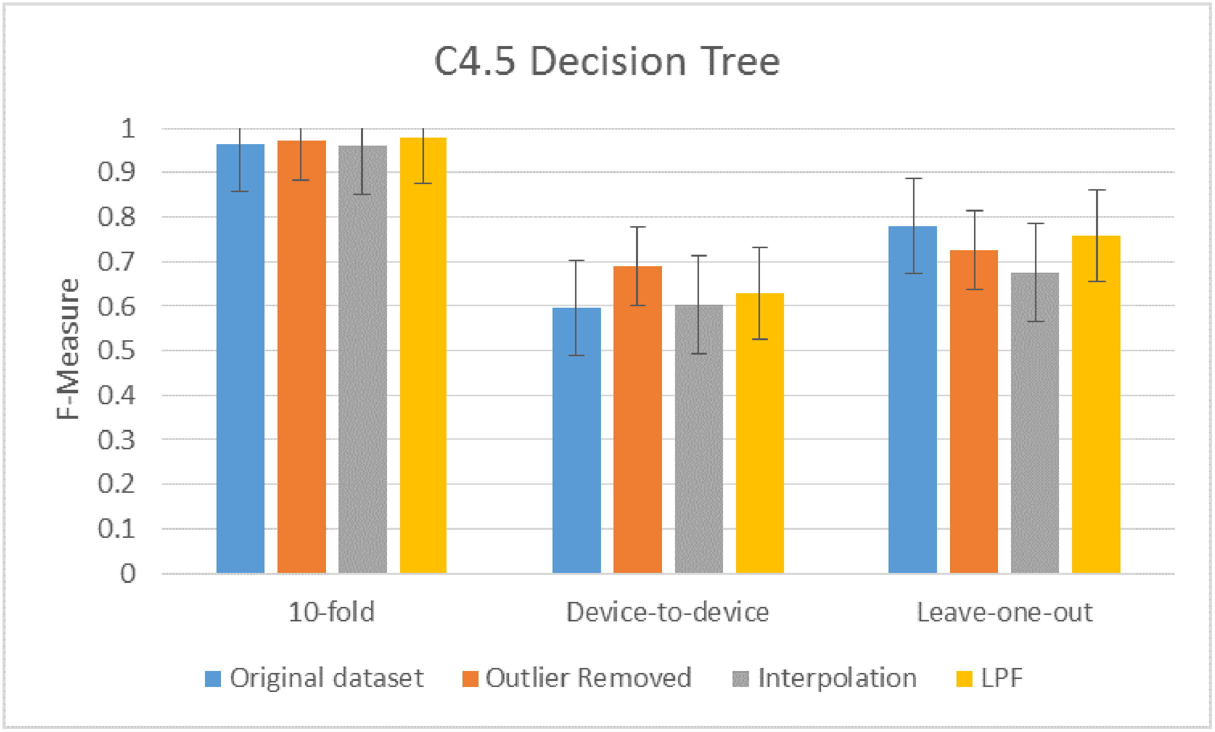


Figure 5.8: Activity recognition performance using C4.5 decision tree classifier

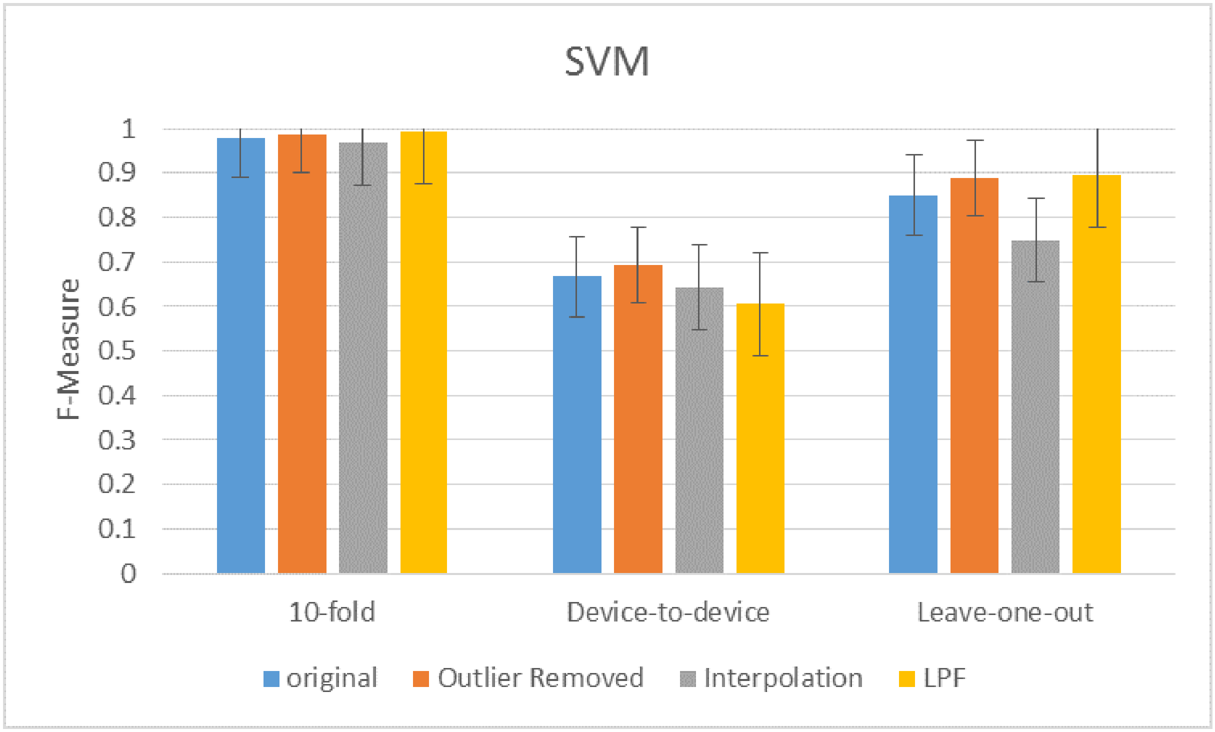


Figure 5.9: Activity recognition performance using SVM classifier

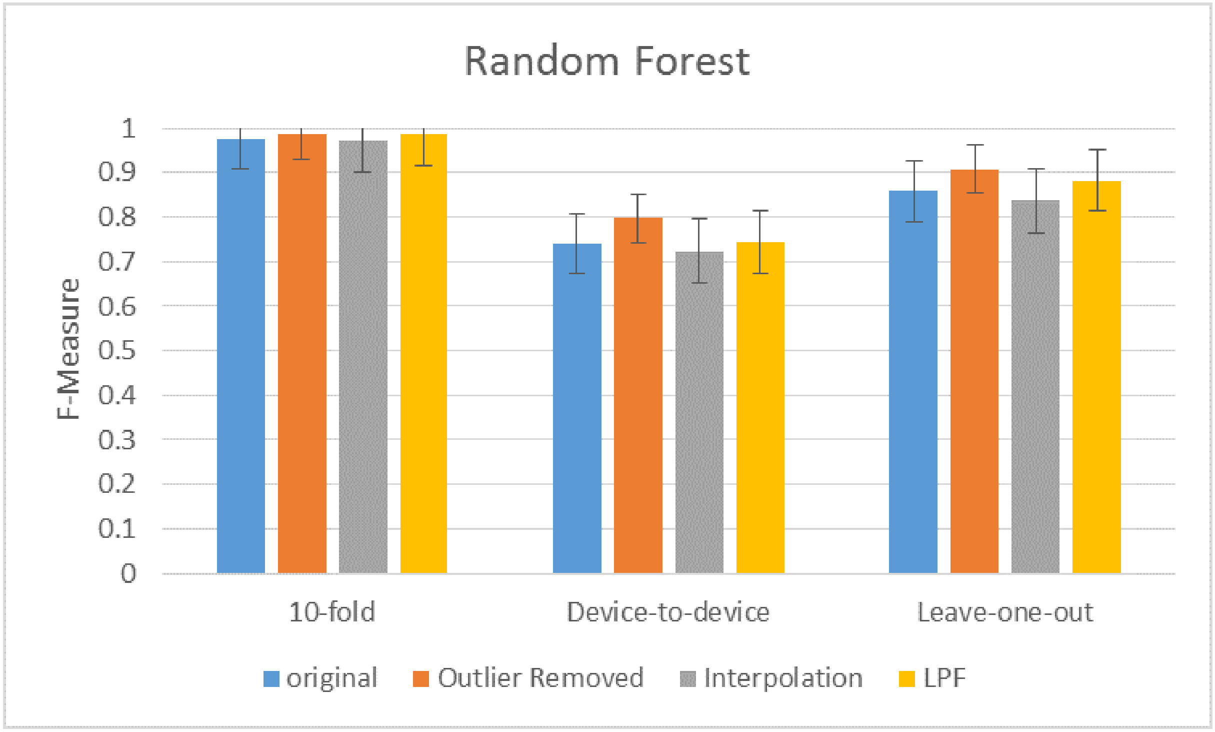


Figure 5.10: Activity recognition performance using random forest classifier

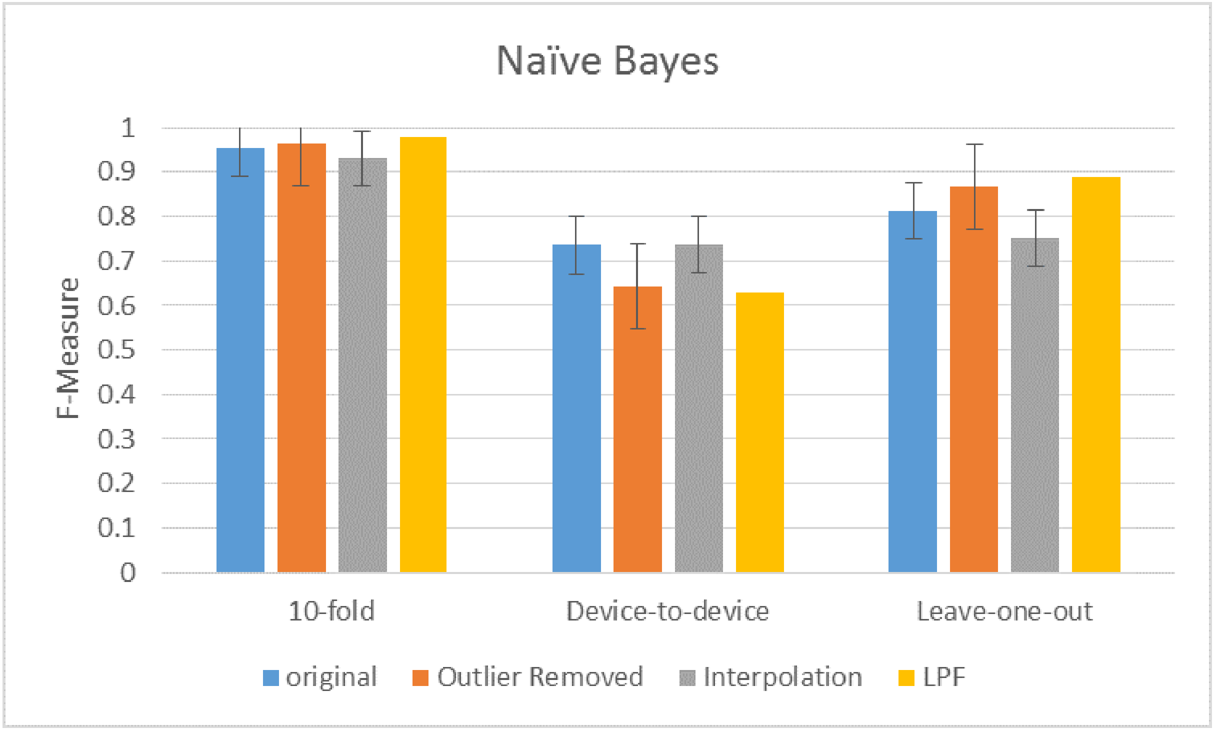


Figure 5.11: Activity recognition performance using Naive Bayes classifier

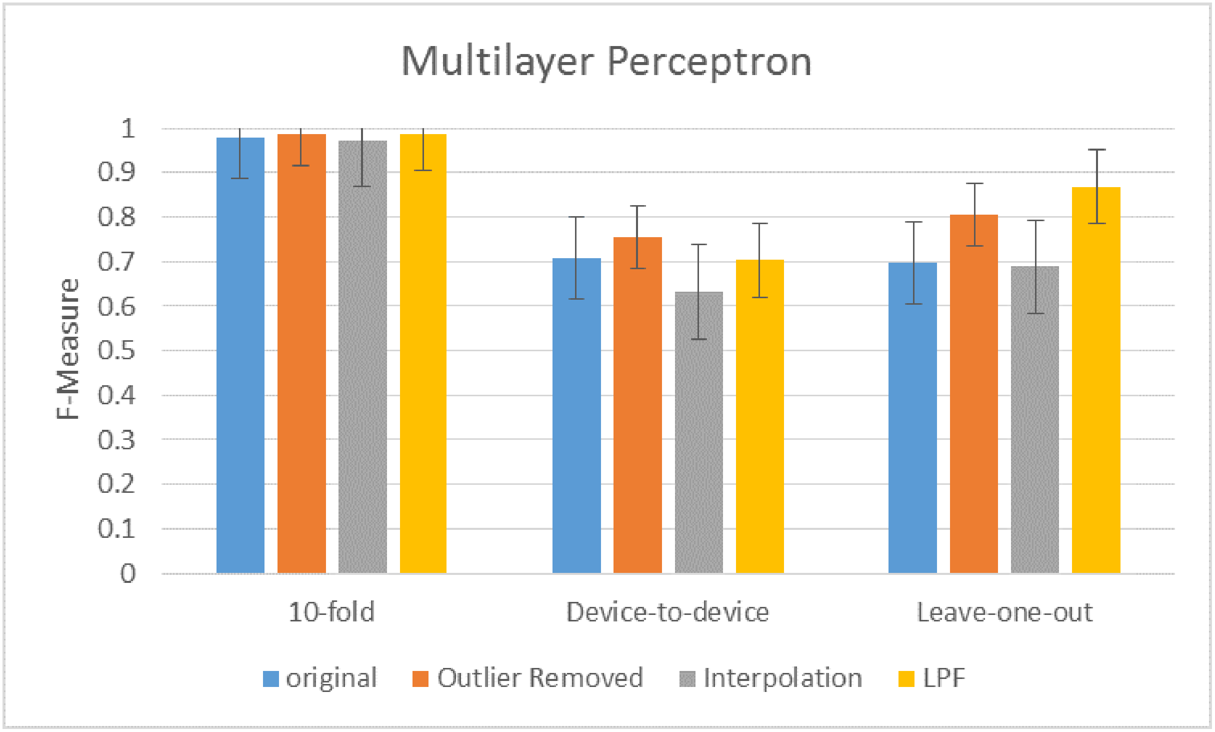


Figure 5.12: Activity recognition performance using multilayer perceptron classifier

5.4 Summary

In this chapter, the issue of sensor differences when applying HAR systems across multiple devices is stressed. A case study involving four distinct smartphones is conducted to investigate the sensor differences and assess some potential mitigating methods. By analyzing the impairments caused by smartphone sensor differences, we show the necessity to introduce mitigating techniques for multi-device activity recognition systems. To alleviate the sensor biases and sampling instability, three preprocessing techniques are examined. Among them, the outlier removal and LPF have positive effects, while up-sampling interpolation does not help improve the system performance.

Chapter 6

Conclusions and Future work

6.1 Conclusions

Smartphone based human activity recognition (HAR) has a wide range of applications including healthcare, fitness, and anomalous situations alerting. This study focuses on detection of human activities including walking, running, sitting, going upstairs, and going downstairs. Embedded smartphone sensors including a tri-axial accelerometer and a gyroscope sensor are used for motion data collection. Both time-domain and frequency- domain features are extracted and analyzed.

Firstly, the research issue on human activity recognition based on smartphone sensors is investigated. We found that smartphone based human activity recognition systems take advantage of the evolutionary software and hardware of smartphone devices and are more flexible compared with other wearable devices. Various data processing and machine learning techniques are explored and supervised learning approach for the HAR problem is examined. Our data collection, analyses, and experiments show that the time-domain features extracted from the raw smartphone sensor readings are good enough to recognize some basic human activities with satisfied classification accuracy. By investigating some popular machine learning algorithms (C4.5 Decision Tree, Support Vector Machine, Naive Bayes, and Multilayer Perceptron), we conclude that for detecting basic human activities, C4.5 Decision Tree is a better solution to be integrated in smartphone applications. In addition, compared with time-domain and frequency-domain features, time-domain features are more reliable and practicable for real-time recognition, though both of these features can generate good classification results.

It is believed that there is a need for HAR applications to not only work in a supervised manner but also in an unsupervised way, and preferably work incrementally, which can reduce the computation burden of the mobile devices. Therefore, an incremental clustering algorithm for human activity recognition is proposed. The proposed two-stage incremental algorithm deals with streamed data from smartphone sensors and perform clustering within mobile devices. It first works in a single pass manner, and then it increases the number of final clusters one by one with the criterion of minimizing the distortion error until the user pre-defined number of clusters is reached. The first stage can help save memory space and computation power when performing clustering in the mobile devices, while the second stage helps alleviate the performance loss in the first streaming stage. The proposed algorithm can reach comparable results with classical clustering methods. The effects of different numbers of initial points and window sizes are also explored. This work is promising as it can be used for data auto-annotation without any priori knowledge.

In addition the impacts of sensor differences on multiple mobile devices when deploying HAR applications are studied. A case study is conducted to assess the effects of smartphone sensor differences. Using three different validation methods, we show how the sensor differences impair the HAR performance. Thus it is necessary to introduce some mitigating techniques. Three pre-processing techniques are examined (linear interpolation, outlier removal, and low pass filter). Both the outlier removal and the low pass filter have positive effects on alleviating the sensor differences, while up-sampling interpolation does not help improve the system performance.

In summary, this thesis provides solutions for human activity recognition employing smartphone embedded sensors. For the supervised learning approach, various features and classifiers are examined and satisfied results are obtained. However the best sampling rate and window size for smartphone sensing systems remain to be determined. For the unsupervised approach, we show the effectiveness of the incremental clustering method. It is promising to work on multiple devices as it does not need the labelling work and training process. The proposed incremental clustering method is based on the classic K-Means algorithms, which is simple and efficient but may not be capable of clustering more activities. Although some techniques are provided to alleviate sensor differences and have shown positive effects, these techniques are the most basic and simple ones. To see significant improvements of HAR across multiple devices, either more advanced filters or more complex interpolation needs to be introduced. In addition, detecting outliers in the dataset more efficiently is another issue we should pay attention to. All these open problems lead to the future work described below.

6.2 Future Work

The following ideas can be the potential future work based on our discussion in the previous section:

- It may be promising to incorporate more sensors (e.g. video and mircophone sensors) to recognize more complex human activities, such as meeting, in elevator, driving, in hand (dangling), and in hand (texting). Besides the supervised learning algorithms investigated in the thesis, the deep learning approaches are also inspiring to help train high dimensional sensor data.

- Due to the limitation of fixed sampling rate and window size for smartphone sensing systems, our work can highly benefit from more robust and adaptive sampling methods. Sensing adaptively can bring us benefits of reducing energy consumptions without sacrificing the recognition accuracy.

- In order to mitigate sensor differences, more complex and adaptive filters seem promising and need further investigation. Interpolations for both up-sampling and down-sampling at different frequency are needed. In addition, the criterion of detecting outliers in the dataset can be extended statically. The concept of autonomic systems [76] may also be brought into smartphone based HAR applications to alleviate sensor differences across multiple devices.

- Further interesting future work for this study would be the investigation of high level human activities. It has been noticed that with the recognized basic human activities, high level human activities can be identified using topic models. The ability of sensing high level activities based on smartphone and topic models can be an exciting avenue to explore.

Bibliography

[1] "Android coordinate-system," 2016. [Online; accessed 27-May-2016].

[2] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, p. 33, 2014.

[3] Google, "Android api description," 2016. [Online; accessed 28-March-2016].

[4] C. G. Ryan, P. M. Grant, W. W. Tigbe, and M. H. Granat, "The validity and reliability of a novel activity monitor as a measure of walking," *British journal of sports medicine*, vol. 40, no. 9, pp. 779–784, 2006.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[6] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[7] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[8] R. Lippmann, "An introduction to computing with neural nets," *IEEE Assp magazine*, vol. 4, no. 2, pp. 4–22, 1987.

[9] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of clinical epidemiology*, vol. 49, no. 11, pp. 1225–1231, 1996.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[11] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, 2007.

[12] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 338–345, Morgan Kaufmann Publishers Inc., 1995.

[13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[14] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[15] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[17] "Weka developer manual," 2016. [Online; accessed 18-June-2016].

[18] M. Zhang and A. A. Sawchuk, "Usc-had: a daily activity dataset for ubiquitous activity recognition using wearable sensors," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 1036–1043, ACM, 2012.

[19] Z. Zhou, X. Chen, Y.-C. Chung, Z. He, T. X. Han, and J. M. Keller, "Activity analysis, summarization, and visualization for indoor human activity monitoring," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 11, pp. 1489–1498, 2008.

[20] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pp. 233–240, IEEE, 2010.

[21] D. T. G. Huynh, *Human activity recognition with wearable sensors*. PhD thesis, Technische Universität Darmstadt, 2008.

[22] B. Dong and S. Biswas, "Wearable networked sensing for human mobility and activity analytics: A systems study," in *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pp. 1–6, IEEE, 2012.

[23] D. Curone, G. M. Bertolotti, A. Cristiani, E. L. Secco, and G. Magenes, "A real-time and self-calibrating algorithm based on triaxial accelerometer signals for the detection of human posture and activity," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, no. 4, pp. 1098–1105, 2010.

[24] C. V. Bouten, K. Koekkoek, M. Verduin, R. Kodde, and J. D. Janssen, "A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity," *Biomedical Engineering, IEEE Transactions on*, vol. 44, no. 3, pp. 136–147, 1997.

[25] T. Al-Ani, Q. T. Le Ba, and E. Monacelli, "On-line automatic detection of human activity in home using wavelet and hidden markov models scilab toolkits," in *Control Applications, 2007. CCA 2007. IEEE International Conference on*, pp. 485–490, IEEE, 2007.

[26] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive computing*, pp. 1–17, Springer, 2004.

[27] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2010.

[28] O. Banos, M. Damas, H. Pomares, A. Prieto, and I. Rojas, "Daily living activity recognition based on statistical feature quality group selection," *Expert Systems with Applications*, vol. 39, no. 9, pp. 8013–8021, 2012.

[29] L. Zhang, T. Liu, S. Zhu, and Z. Zhu, "Human activity recognition based on triaxial accelerometer," in *Computing and Convergence Technology (ICCCT), 2012 7th International Conference on*, pp. 261–266, IEEE, 2012.

[30] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[31] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Tröster, "Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection," in *Wireless sensor networks*, pp. 17–33, Springer, 2008.

[32] J. Yin, Q. Yang, and J. J. Pan, "Sensor-based abnormal human-activity detection," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 8, pp. 1082–1090, 2008.

[33] S.-W. Lee and K. Mase, "Activity and location recognition using wearable sensors," *IEEE pervasive computing*, vol. 1, no. 3, pp. 24–32, 2002.

[34] B. Najafi, K. Aminian, A. Paraschiv-Ionescu, F. Loew, C. J. Büla, and P. Robert, "Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly," *Biomedical Engineering, IEEE Transactions on*, vol. 50, no. 6, pp. 711–723, 2003.

[35] A. Subramanya, A. Raj, J. A. Bilmes, and D. Fox, "Recognizing activities and spatial context using wearable sensors," *arXiv preprint arXiv:1206.6869*, 2012.

[36] J. Pärkkä, M. Ermes, P. Korpipää, J. Mäntyjärvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 1, pp. 119–128, 2006.

[37] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in *Wearable Computers, 2007 11th IEEE International Symposium on*, pp. 37–40, IEEE, 2007.

[38] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, 2014.

[39] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, and R. Jafari, "Enabling effective programming and flexible management of efficient body sensor network applications," *Human-Machine Systems, IEEE Transactions on*, vol. 43, no. 1, pp. 115–133, 2013.

[40] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen, "The opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, 2013.

[41] S.-H. Fang, Y.-C. Liang, and K.-M. Chiu, "Developing a mobile phone-based fall detection system on android platform," in *Computing, Communications and Applications Conference (ComComAp), 2012*, pp. 143–146, IEEE, 2012.

[42] J. Cho, J. Kim, and T. Kim, "Smart phone-based human activity classification and energy expenditure generation in building environments," in *Proceedings of the 7th international symposium on sustainable healthy buildings*, vol. 2012, pp. 97–105, 2012.

[43] M. Boyle, A. Klausner, D. Starobinski, A. Trachtenberg, and H. Wu, "Poster: Gait-based smartphone user identification," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pp. 395–396, ACM, 2011.

[44] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 337–350, ACM, 2008.

[45] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living*, pp. 796–799, Springer, 2009.

[46] J.-H. Chiang, P.-C. Yang, and H. Tu, "Pattern analysis in daily physical activity data for personal health management," *Pervasive and Mobile Computing*, vol. 13, pp. 13–25, 2014.

[47] A. Anjum and M. U. Ilyas, "Activity recognition using smartphone sensors," in *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pp. 914–919, IEEE, 2013.

[48] Y. Kwon, K. Kang, and C. Bae, "Unsupervised learning for human activity recognition using smartphone sensors," *Expert Systems with Applications*, vol. 41, no. 14, pp. 6067–6074, 2014.

[49] X. Su, H. Tong, and P. Ji, "Activity recognition with smartphone sensors," *Tsinghua Science and Technology*, vol. 19, no. 3, pp. 235–249, 2014.

[50] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen, "Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pp. 127–140, ACM, 2015.

[51] S. Kwon, D. Lee, J. Kim, Y. Lee, S. Kang, S. Seo, and K. Park, "Sinabro: A smartphone-integrated opportunistic electrocardiogram monitoring system," *Sensors*, vol. 16, no. 3, p. 361, 2016.

[52] C. Medrano, I. Plaza, R. Igual, Á. Sánchez, and M. Castro, "The effect of personalization on smartphone-based fall detectors," *Sensors*, vol. 16, no. 1, p. 117, 2016.

[53] E. Casilari, R. Luque, and M.-J. Morón, "Analysis of android device-based solutions for fall detection," *Sensors*, vol. 15, no. 8, pp. 17827–17894, 2015.

[54] F. Miao, Y. Cheng, Y. He, Q. He, and Y. Li, "A wearable context-aware ecg monitoring system integrated with built-in kinematic sensors of the smartphone," *Sensors*, vol. 15, no. 5, pp. 11465–11484, 2015.

[55] F. Vandewiele and C. Motamed, "An unsupervised learning method for human activity recognition based on a temporal qualitative model," in *International Workshop on Behaviour Analysis and Video Understanding (ICVS 2011)*, p. 9, 2011.

[56] D. Trabelsi, S. Mohammed, F. Chamroukhi, L. Oukhellou, and Y. Amirat, "An unsupervised approach for automatic activity recognition based on hidden markov model regression," *Automation Science and Engineering, IEEE Transactions on*, vol. 10, no. 3, pp. 829–835, 2013.

[57] W.-H. Ong, L. Palafox, and T. Koseki, "An incremental approach of clustering for human activity discovery," *IEEJ Transactions on Electronics, Information and Systems. C*, vol. 134, no. 11, pp. 1724–1730, 2014.

[58] R. Minhas, A. A. Mohammed, and Q. Wu, "Incremental learning in human action recognition based on snippets," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 11, pp. 1529–1541, 2012.

[59] Z. Wang, M. Jiang, Y. Hu, and H. Li, "An incremental learning method based on probabilistic neural networks and adjustable fuzzy clustering for human activity recognition by using wearable sensors," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 16, no. 4, pp. 691–699, 2012.

[60] H. Yu, C. Zhang, and G. Wang, "A tree-based incremental overlapping clustering method using the three-way decision theory," *Knowledge-Based Systems*, vol. 91, pp. 189–203, 2016.

[61] A. M. Bagirov, J. Ugon, and D. Webb, "Fast modified global k-means algorithm for incremental cluster construction," *Pattern recognition*, vol. 44, no. 4, pp. 866–876, 2011.

[62] M. Charikar, L. O'Callaghan, and R. Panigrahy, "Better streaming algorithms for clustering problems," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pp. 30–39, ACM, 2003.

[63] M. Cottrell, B. Hammer, A. Hasenfuß, and T. Villmann, "Batch and median neural gas," *Neural Networks*, vol. 19, no. 6, pp. 762–771, 2006.

[64] C. Gupta and R. L. Grossman, "Genic: A single-pass generalized incremental algorithm for clustering.," in *SDM*, pp. 147–153, SIAM, 2004.

[65] A. M. Bagirov, "Modified global k-means algorithm for minimum sum-of-squares clustering problems," *Pattern Recognition*, vol. 41, no. 10, pp. 3192–3199, 2008.

[66] D. T. Pham, S. S. Dimov, and C. Nguyen, "An incremental k-means algorithm," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 218, no. 7, pp. 783–795, 2004.

[67] D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Personal and Ubiquitous Computing*, vol. 14, no. 7, pp. 645–662, 2010.

[68] M. Mathie, *Monitoring and interpreting human movement patterns using a triaxial accelerometer*. PhD thesis, The University of New South Wales, 2003.

[69] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 1, pp. 156–167, 2006.

[70] "Sequential k-means," 2016. [Online; accessed 28-March-2016].

[71] N. D. Lane and P. Georgiev, "Can deep learning revolutionize mobile sensing?," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pp. 117–122, ACM, 2015.

[72] "Samsung health app," 2016. [Online; accessed 27-May-2016].

[73] "Sony lifelog app," 2016. [Online; accessed 27-May-2016].

[74] "Apple health app," 2016. [Online; accessed 27-May-2016].

[75] T. Maekawa and S. Watanabe, "Unsupervised activity recognition with user's physical characteristics data," in *Wearable Computers (ISWC), 2011 15th Annual International Symposium on*, pp. 89–96, IEEE, 2011.

[76] S. Galzarano, G. Fortino, and A. Liotta, "Embedded self-healing layer for detecting and recovering sensor faults in body sensor networks," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2377–2382, IEEE, 2012.

Curriculum Vitae

Name:	Xizhe Yin
Post-Secondary Education and Degrees:	2014 - 2016, M.E.Sc Department of Electrical and Computer Engineering Faculty of Engineering University of Western Ontario London, Ontario, Canada
	2010 - 2014, B.Eng Department of Automation School of Information Science and Technology University of Science and Technology of China Hefei, Anhui, P.R. China
Related Work Experience:	Teaching Assistant The University of Western Ontario 2014 - 2016

Publications:

[1] Yin, X., Shen, G., Wang, X., Shen, W. Mitigating Sensor Differences for Phone-based Human Activity Recognition, Proceedings of 2016 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2016), Budapest, Hungary, Oct. 9-12, 2016.

[2] Yin, X., Shen, W., Wang, X. Incremental Clustering for Human Activity Detection Based on Smart Phone Sensor Data, Proceedings of 2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (IEEE CSCWD 2016), Nanchang, China, May 4-6, 2016, pp. 35-40.

[3] Yin, X., Shen, W., Samarabandu, J., Wang, X., Human Activity Detection Based on Multiple Smart Phone Sensors and Machine Learning Algorithms, Proceedings of 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (IEEE CSCWD 2015), Calabria, Italy, May 6-8, 2015, pp. 582-587.