PERFORMANCE EVALUATION OF CLASSIFICATION METHODS FOR ONLINE ACTIVITY RECOGNITION ON SMART PHONES

by

Mustafa Köse

B.S., Mathematics, Boğaziçi University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2012

ACKNOWLEDGEMENTS

First of all, I would like to thank to my thesis coadviser Ozlem Durmaz Incel. I am appreciative to her for her continuous support, guidance and efforts since the beginning of this thesis. It was good opportunity for me to work with her during this process. I am also grateful to my thesis supervisor Prof. Cem Ersoy not only for his guidance during this thesis, but also for his continuous support since the beginning of my graduate study. Additionally, I want to thank to Assist. Prof Albert Ali Salah and Assist. Prof Aysegul Erman Tuysuz for their evaluations and contributions for this thesis as well. I am also thankful to my family for their love and patience during my stressful times. I am surely grateful for all my friends' support, especially for their effort during test runs. Lastly, Ebru and Elcin; I am grateful for all your continuous support during my hardest times.

This research is supported by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610, and by the Bogazici University Research Fund under the grant agreement number "6056".

ABSTRACT

PERFORMANCE EVALUATION OF CLASSIFICATION METHODS FOR ONLINE ACTIVITY RECOGNITION ON SMART PHONES

Human activity recognition using sensory data has become an active field of research in the domain of pervasive and mobile computing. It involves the use of different sensing technologies to automatically collect and classify user activities for different application domains. In fact, smart phones with their sensing capabilities can also be used as a platform for human activity recognition. Although many studies have been introduced so far, there are few which consider online training and classification of activities as well as evaluating the online performance of existent classifiers. In this thesis, our aim is to analyse the performance of different classification methods for online activity recognition on smart phones using the built-in accelerometers considering important limitations of the phones, such as battery usage and limited computational power. For this purpose, we developed a mobile application on Android platform which performs online classification. We conducted experiments to investigate the performance of the system under the effect of several important factors including sampling rate and window size on several Android smart phones. The tests are performed on different subjects for activities of *walking, running, sitting, standing* and *biking.* We evaluated the performance of the activity recognition system using the Naive Bayes classifier, and next we utilized Clustered KNN and Decision Tree algorithms. According to the results, Naïve Bayes provides not satisfactory results whereas Clustered KNN gives promising results compared to the previous studies and even with the ones which consider offline classification. Additionally, Decision Tree results are also comparable with the results of Clustered KNN.

ÖZET

SINIFLANDIRMA YÖNTEMLERİNİN AKILLI TELEFONLAR ÜZERİNDE ÇEVRİMİÇİ EYLEM TANIMA İÇİN BAŞARIM DEĞERLENDİRMESİ

İnsan eylemlerinin görsel ve hareket algılayıcı verileri ile tanınması konusu hareketli ve sürekli/yaygın hesaplama alanında çalışılan güncel araştırma konularından biri olarak yer almaktadır. Bu konu, temel olarak, farklı algılama teknolojileri kullanılarak insan eylemleri ile ilgili veri toplanmasını ve toplanan veri ile eylemlerin sınıflandırılmasını içermekte, aynı zamanda sağlık, destekli yaşam, spor ve eğlence gibi uygulama alanlarını hedeflemektedir. Bu alanda pek çok araştırma olmasına rağmen, mevcut sınıflandırıcıların başarımlarını karşılaştıran ve aynı zamanda eylemleri çevrimiçi eğitim ve sınıflandırma yöntemleri ile tanımlayan çok az örnek çalışma bulunmaktadır. Bu tezin amacı, insan eylemlerinin akıllı telefonlar üzerindeki ivmeölçer algılayıcısı kullanılarak tanınmasıdır. Amacımız, yürüme, koşma, oturma, ayakta durma, bisiklete binme gibi temel insan hareketlerinin telefon üzerinde veri işlenmesi ile sınıflandırılmasıdır. Literatürde yer alan çalışmalardan farklı olarak, veri toplama, eğitim kümesi modelleme ve aktivite sınıflandırması çevrimiçi olarak yapılmaktadır. Bunun yanında, Naif Bayes, kümelenmiş KNN ve Karar Ağacı sınıflandırma algoritmalarının çevrimiçi başarımları karşılaştırılmıştır. Bu amaçla hedeflediğimiz telefon modelleri ile uyumlu bir Android uygulaması geliştirilmiş ve literatürde yer alan çalışmalardan farklı olarak, birden fazla telefon modeli ile performans değerlendirmesi yapılmıştır. Sınıflandırma algoritmalarının başarımları, farklı deneklerle test edilmiş ve sonuçlara göre, kümelenmiş KNN tekniği diğer sınıflandırma algoritmalarını doğruluk ve çalışma süresi açısından daha yüksek başarım sergilemiştir. Ayrıca, örnekleme zamanı, pencere büyüklüğü ve bunun gibi önemli sistem parametrelerinin başarım üzerindeki etkisi de incelenmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	Accelerometer Data Matrix
$\mathbf{A}$	Accelerometer set containing all accelerometer data collected
A_k	Accelerometer set containing all accelerometer data of activity type k
A_d	Acceleration applied to the device
C	Candidate Accelerometer Data
F_s	Forces applied to sensor
FN	The number of False Negative classifications

LIST OF ACRONYMS/ABBREVIATIONS

ALC	Adaptive Linear Classifier
CHMM	Chain Hidden Markov Model
CKNN	Clustered K Nearest Neighbour
DT	Decision Tree
DFT	Discrete Fourier Transform
GMM	Gaussian Mixture Models
HMM	Hidden Markov Model
NB	Naïve Bayes
NN	Nearest Neighbour
KNN	K Nearest Neighbour
MD	Minimum Distance
TI	True Inferred
TP	True Positive
TT	Total of Ground Truth
SFS	Sequential Forward Selection
SFFS	Sequential Floating Forward Selection
SI	International System of Units
SVM	Support Vector Machines
WMLT	Weka Machine Learning Toolkit

1. INTRODUCTION

Human activity recognition using sensory data has become an active field of research in the domain of pervasive and mobile computing. It involves the use of different sensing technologies to automatically collect and classify user activities for different application domains. Target application areas include medical applications (monitoring and diagnosis), home monitoring and assisted living (child and elderly care, monitoring and emergency help, assistance for people with cognitive disorders), sports and leisure applications (daily sport activities, martial arts) [4]. Initially, vision-based sensing [5], using cameras has been the focus of research studies on human activity recognition and more recently inertial sensing [6], using movement-based sensors that can be attached to the user's body has been investigated [7]. Motivated by the recent studies and with the release of smart phones equipped with a rich set of sensors, such as accelerometer, GPS, microphone, human activity recognition using mobile phone sensing is being researched [4, 8–16].

Algorithms used in the classification of activities originate from statistical machine learning techniques. However, a trendy algorithm in machine learning research may not exhibit a superior performance in the field of activity recognition [17], especially on the mobile phone platform with limited resources, considering the limited processing power and battery. Moreover, when we look at the literature on activity recognition using inertial sensors, we observe that most of the studies first collect sensory data and apply classification algorithms offline on the collected data, using a large part of the collected data for training. It is clear that larger the amount of overlap between the training data and the testing data, better recognition results will be achieved (unless the over-fitting problem occurs). Offline processing certainly exploits this advantage.

Offline processing can be used for applications where online recognition is not necessary. For instance, if we are interested in following the daily routine of a person, such as in [11], the sensors can collect the data during a day; the data can be uploaded to a server at the end of the day and can be processed offline for classification purposes. However, for applications such as a fitness coach where the user is given a program with a set of activities, their duration and sequence, we might be interested in what the user is currently doing and/or if he is performing the activities with a correct sequence [11]. Another application can be the recruitment for participatory sensing applications [18]. For instance, the application might be interested in collecting information from users that are *'currently walking'* in a particular part of a city. Another example application is given in [19]. The application creates an online activity map of a city by recognizing the transportation modes of the users, biking, travelling with a train, with a vehicle, etc. Therefore, online recognition of activities becomes important, especially for real-world applications running on smart phones to provide the context of the users.

Training requirements of the proposed systems differ in terms of being online or offline. In the literature, training phase is usually performed offline. Even though a proposed system does online classification, training phase can be handled with offline processing. Usually, proper training models are being created in offline training so that these static models can be used in an online classification phase [10]. Offline training phase is not an easy task and it is the common need of an activity recognition system since such systems require large, well-defined and proper training sets to create appropriate models. Such training sets are collected over a long period of time (couple of days, weeks or even years depending on the related work) with dedicated and a sufficient number of test subjects as presented in [20]. Additionally, collected data sets can be too large to be processed online on other devices like smart phones because of their limited capabilities. Considering these challenges, research on human activity recognition systems explore the ways for online training [21]. Online training can be used in systems where it is needed to create a user-friendly structure which does not need user intervention or long training sessions at activity recognition steps. Such systems can also increase the attention of the users to the subject and their adoption of such applications. It is a known fact that if a system is not preferred and used by end users, it does not have any value. Hence, we believe that the future of activity recognition depends on systems which work either with limited data or which do not require long training sessions.

There are various underlying challenges for activity recognition. Human behaviour is the first of them. People can perform multiple tasks at the same time which can affect the activity recognition process negatively. Additionally, various continuous sequences of performing tasks and their periodic variations may result in incorrect classifications. Because of these reasons, accuracy and reliability of sensor data play an important role in activity recognition. Other challenges in this topic are sensor placement and resource constraints. It is well known and clearly shown that orientation and placement of the sensors on the body plays an important role on the success of activity recognition and on the accuracy of results [4]. When mobile phones are considered in terms of processing and sensing capabilities, the most important constraint is the battery limitation. Besides that, devices should have enough memory space in terms of both software components and data needed to be stored. Additionally, they should be computationally capable to execute training and classification algorithms.

In this thesis, we focus on activity recognition using the embedded accelerometers on smart phones. Our objective is the classification of basic movements of a user, such as *walking, running, sitting, standing* and *biking.* As mentioned before, in contrast to the offline processing of data, we focus on online classification of these activities and evaluate the performance of classifiers such as Naïve Bayes, Clustered KNN and Decision Tree. For this purpose, we developed an activity recognition system and implemented these classifiers on the Android platform.

Besides online recognition of activities, the training phase is also performed on the phone instead of processing the data offline to learn the model parameters. As stated earlier, it is known that training phase of a classification method used in activity recognition is costly and there is a general interest in providing an activity recognition system that does not need training [21] or requires only limited training by the end user. In this study, we are interested in the performance of classifiers with limited training data considering the limited memory available on the phones. In the proposed system, training data can be collected only in a few minutes and can be used directly for classification steps which reduce the burden on the users. Being one of the first Android applications used for activity recognition is another important motivation for this thesis. Another contribution is that the performance evaluation is carried out by using several smart phone models with different capabilities instead of focusing on one model.

Performance of the classifiers is tested with ten different subjects in two batches of experiments by using the main accelerometer features such as minimum, maximum, average, average acceleration in the Y-direction and standard deviation. Test results show that the Clustered KNN approach mostly outperforms other classifiers in terms of recognition and execution time on the phones. The impact of sensor sampling rate and window size, which is used for segmenting the data, on the performance of activity recognition is also analyzed. Moreover, we also compared offline processing results of the same classifier set with online classification results with the help of Weka Machine Learning Toolkit [22]. We analyzed user dependency of the system by mixing the training sets of different users. Lastly, effect of using sliding windows for segmenting the data during the online activity recognition process is investigated.

The rest of the thesis is organized as follows. In Chapter 2, we present the background of our study and related works in order to understand the research field of activity recognition and future expectations. Firstly, we provide important definitions used in the thesis. Next, we detail activity recognition steps and different methodologies proposed in the literature. Later, related studies are presented together with a taxonomy and comparisons. In Chapter 3, activity recognition steps and processes used in the thesis are detailed by also providing a background for the proposed methods. Next, we present details of classification methods which are used in this thesis. Lastly, we finish the chapter by providing the details of the system proposed and implementation of the Android application. In Chapter 4, we first present the experiment design to evaluate the performance of the proposed online activity classification system. Next, we provide the results of our experiments, carried with different test subjects, in terms of the performance of classifiers using different sets of parameters. In Chapter 5, we provide our conclusions and directions for future research.

2. BACKGROUND

In this chapter, we first give an overview on the details of the activity recognition process, including the usage of accelerometer for activity recognition and the steps for the classification of activities. Next, related studies on acitivity recognition using mobile phones are summarized and classified.

2.1. Acceleration and Accelerometers for Activity Recognition

Acceleration is defined to be the rate of change in velocity over time. There are two kinds of acceleration called average and instantaneous. Average acceleration is determined over a finite time interval whereas instantaneous acceleration is the derivative of the velocity with respect to time where velocity is the change in displacement with time at a given instant. Constant acceleration is the special case when average acceleration and instantaneous acceleration values are equal. Free-fall acceleration is the constant acceleration which is caused by gravitational force. Lastly, the acceleration of an object relative to an observer, whose net acceleration is zero, is called proper acceleration and it is measured by an accelerometer.

Accelerometer is the sensor device which measures proper acceleration of the object where it is attached. It produces a time series of each axes in 3D $(x(t), y(t), z(t))$ for each clock cycle. An accelerometer measures weight per unit of mass. For example, an accelerometer measures a value of g in the upward direction when remaining stationary on the ground because masses on earth have weight $m \times g$. On the other hand, an accelerometer which is in free-fall gives a value of zero acceleration since it is at rest in a frame of reference in which objects are weightless although its speed is increasing. Accelerometers can measure vibrations, shocks, tilt, impacts and motion of an object.

2.2. Activity Recognition Steps

In the literature, there are many different methods to receive activity information from raw sensor data. Main steps can be categorized as *Preprocessing, Segmentation, Feature Extraction, Dimensionality Reduction* and lastly *Classification* which are given in Figure 2.1. These steps can be used either all together or according to the requirement, only some of them can be selected and applied during activity recognition process. In Figure 2.1, input represents the raw sensor data whereas output represents the activity information.

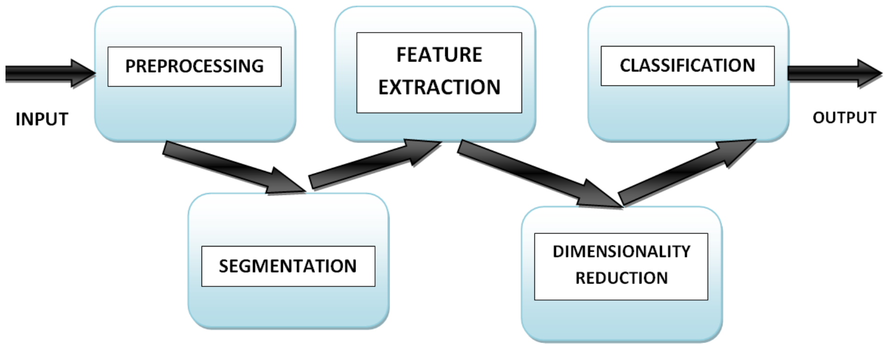


Figure 2.1. Activity recognition steps [4].

Preprocessing. This step contains noise removal and representation of raw data. Initial samples received from any type of sensor are called raw data. Because of the nature of sensors, gathered raw data has a potential noise which prevents us to detect usual behaviour of activity patterns. Because of this reason a preprocessing stage should be applied to the raw data before using it in activity recognition process. There are different methods for noise removal processes such as non-linear, low-pass [23] or high-pass, Laplacian [24] and Gaussian [25] filters.

Segmentation. Retrieving useful information from continuous stream data is a difficult problem. For this purpose, different segmentation methods can be applied to time-series data which enhance signal behaviour and enable us to gather useful information from continuous stream of data. Sliding windows can be considered as one of the segmentation methods. Some examples of stream processing are illustrated in Figure 2.2. Explicit segmentation can be used at two level approaches such as splitting and merging. By this way, the streaming data is divided into chunks before classification starts. The drawback of this approach is to decide the appropriate size for each chunk which increases the complexity. Additionally, one always needs to wait for future data to make decisions on the previously splitted data. As an example, in [26], newly collected data is added to the previous sample set until fit-error rate of the current segment does not exceed the threshold value set by the user.

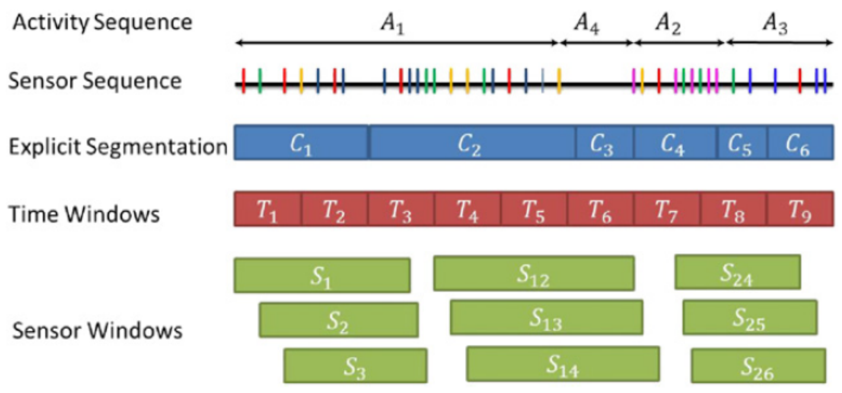


Figure 2.2. Illustration of different windowing approaches [3].

On the other hand, dividing the whole sequence to equal time windows further decreases the computational complexity. Because of this reason, it is presented as a suitable approach in activity recognition studies dealing with continuous streaming data sampled with constant rates. However, one has to deal with finding the optimal time intervals for targeted activities in this approach which directly affects the system performance. The last approach presented in the figure is to divide the streaming data based on the count of the sensor events.

Other approaches like top-down, iterative end point fits, bottom-up [26] and/or any combination of them can be used for this purpose. According to the previous studies, sliding window and bottom-up approaches are the ones which have lower complexity whereas combination of them called SWAB [27] has slightly higher complexity but its performance is as good as the Bottom-up algorithm.

Feature Extraction. Features can be defined as the abstractions of raw data since they are reduced sets of original raw data which basically represent main characteristics and behaviours of it. The reduced subset of large input data can be called as a feature vector and it is the main input for classification algorithms and contain important hints for the activity to be recognized.

Features can be grouped as time-domain, frequency-domain, time-frequency domain and heuristic features. Time domain features such as *mean, variance, root mean square, minimum* or *maximum* are the mostly preferred features in activity recognition using mobile phones as indicated in Table 2.1. They basically represent signal statistics and waveform characteristics. Frequency domain features represent periodic behaviour of the sample such as the FFT coefficients, spectral energy or spectral entropy. Time-frequency domain features like wavelet coefficients can be used to illustrate both time and frequency characteristics of complex signals which are mostly used in studies to detect transitions between activities. There are also examples of heuristic features such as inter-axis correlation, signal vector magnitude and signal magnitude area which are widely used [23].

Dimensionality Reduction. The aim of dimensionality reduction is to reduce the computational complexity and increase the performance of the activity recognition process. After the previous steps, collected data can be used directly in the classification step. But some part of data may not even contribute to the results of the classification process. It may just create a burden on the complexity of the process and just slows down training and classification processes, hence the dimensionality reduction is applied to overcome these issues.

For this purpose, different methods are proposed in the literature which can be grouped as *feature selection methods* and *feature transform methods*. Feature selection methods separate most discriminative features whereas feature transform methods combine two or more features which are not effective individually, but meaningful when they are combined. SVM Based Feature Selection [28], K-Means Clustering and Forward Backward Sequential Search [29] are the well-known feature selection techniques. K-Means Clustering is a method which groups selected samples in a compact set by considering the distance metric. The idea is to gather some selected data in a compact set which are located close to each other while excluding all other samples [30].

Classification. The last and the most important step is the *classification* [4]. Training data sets which are processed at previous steps can be used as an input parameter for the classification step. There are various classification methods which are widely used in the literature. Decision Tree, Decision Tables [8], Hidden Markov Models [9], Naïve Bayes [10], Nearest Neighbor, Support Vector Machines [31] and Gaussian Mixture Models are the well-known and widely-used classification techniques. Additionally, in some of the studies threshold-based techniques are preferred for this purpose [24]. Moreover, artificial neural networks-based activity recognition systems [32] previously became a hot topic.

2.3. Related Works

Previous studies are mainly concentrated on the performances of the proposed algorithms and methods. In the following parts, we first summarize the related studies and then provide a taxonomy along with a detailed comparative table.

In [8], a novel design framework, called EEMSS, for energy efficient mobile sensing is proposed. A hierarchical sensor management strategy is used to recognize user states as well as to detect state transitions. User states may contain a combination of features such as motion (running, walking), location (staying at home or on a free-way) and background condition (loud or quiet) which all together describe user's current context. This study particularly focuses on energy efficient sensing, considering the fact that embedded sensors on mobile phones are the major sources for power consumption.

As an example, fully charged Nokia N95 can support telephone conversations more than 10 hours whereas it can function only around 6 hours while GPS is turned on without taking into consideration whether it is taking samples or not. Because of this reason, sensor sources must be used efficiently to recognize activities on mobiles. There are two important points proposed throughout this paper in order to reduce power consumption. The first one is to turn off unused sensors automatically according to user states. For this purpose, an XML-style state descriptor is taken as input and used by the sensor assignment functional block. In this way, selected sensors are sampled during a specific state whereas differently selected sensors are tracking any possible user state transitions. The second point which is important again in terms of power consumption is *'sensor duty cycle'* which negatively affects the recognition latency metric while decreasing the overall power consumption and extending device battery lifetime. The state transition system implemented on Nokia N95 can define the following states currently; *walking, vehicle, resting, home talking, home entertaining, working, meeting, office loud, place quiet, place speech, place loud.* The sensors being used for activity recognition are accelerometer, Wifi detector, GPS and microphone. EEMSS is able to detect states with approximately 92.56% accuracy by processing testing data offline and improves the battery lifetime by over 75% compared to an existing application, Cenceme [12].

In a similar study, Reddy et al. proposed a different model for activity recognition [9]. In this article, they designed, implemented and evaluated a transportation mode classification system which runs on a mobile phone by using 3-axis accelerometer and GPS sensors. Simply, they focused on outdoor activities and classified them into following five groups such as *walking, stationary, biking, running*, and *motorized transport.* The proposed classification system does not consider a specific mobile phone orientation which makes it convenient to use. In order to reduce energy consumption, sensor readings are enabled only if a user goes outside by being triggered through a connected base station change. In this study, they evaluated different classification algorithms, such as Naive Bayes, Chain Hidden Markov Model, Support Vector Machine, Nearest Neighbour, Hidden Markov Model (HMM), Decision Tree (DT), and compared these classifiers with each other. According to the evaluation, DT followed by HMM provided the best results and because of this reason they decided to use DT and HMM together at the final evaluation instead of others. The classification algorithm combines GPS speed data and variance and frequency components of accelerometer signals. Weka machine learning toolkit (WMLT) [22] and generalized HMM library is employed to train the classifier and then the classifier is transported to work on Nokia N95. The final classifier is programmed by using Python for Symbian S60. Overall, the system provides nearly 93% accuracy.

In [12], authors present design, implementation and evaluation of a social networking application, called Cenceme. Basically, users are able to share their contextual information through social platforms like Facebook and MySpace using this application. Cenceme benefits from offline computational powers of back-end servers for training whereas it additionally performs online classification. In this study, authors have focused on sitting, standing, walking and running activities as well as audio info of the environment. For this purpose they used accelerometer, Bluetooth, audio sensors and GPS which are embedded in Nokia N95. Classifiers used in this study can be grouped as audio and activity classifiers. Audio classifier benefits from Discrete Fourier Transform (DFT) and FFT algorithms and its feature vector consists of mean and standard deviation of DFT power. On the other hand, feature vectors at activity classifier are mean, standard deviation and number of peaks per unit time. Simple 3 level decision tree performs classifications online on the Nokia platform and classifies activities into *walking, running, standing* and *sitting* states. Additionally, back-end classification derives contextual information from collected data. Performance of the proposed system is tested with 22 subjects over three weeks period on the Nokia N95 platform. The overall system performance in terms of recognizing online activities is not stated clearly. However, they emphasize the misclassification of standing and sitting cases.

Different from the previous studies, in [10], a classification algorithm is aimed to run on the iPhone platform. For this purpose three iPhone applications are developed called as iLog, iModel, iClassify respectively. iLog is used for data collection for different activities in real time. iModel is a desktop tool for learning and testing models. Data saved by iLog can be imported into iModel which is a Java application built on

WMLT. By using iModel labeled data can be used to test an existing model or to learn a new model. Lastly, iClassify can be used to classify *walking, jogging, bicycling on a stationary bike*, and *sitting* activities in real time in iPhone applications. In terms of methodology and execution times of classification steps, this study differs from the previous two studies. There is an offline data processing step to learn data models and addition to iPhone 3-axis accelerometer sensor, Nike+iPod Sports Kit is used to collect data which imports the collected data periodically to iPhone via Bluetooth connection. iClassify can report activity classifications once per second. It performs nearly 97% accuracy in recognizing activities.

The following references [11–14] have similarities with the studies stated before in terms of methodology and ideas. Additionally, there are still few examples of works which use only wearable mobile sensors for activity recognition such as in [8, 9, 11–13]. The rest of the studies use extra sensors to recognize activities like Nike+iPod Toolkit, motion bands.

Considering the training phase for the classification algorithms, there are only few examples of related works aiming user independent or limited training of proposed systems. In [33], authors introduced a system called Darwin which aims to decrease the training burden on users, and increase user experience on smart phone while proposing a reliable classification system. Their work is based on *model pooling* which is simple sharing and exploitation of classification models which are already built by other phones. By this way, there is no need to generate training models from scratch so that timeliness of classification decreases substantially. The proposed system works both outside and inside and it is implemented on Nokia N97 and Apple iPhone. Tests are performed with eight people and they achieved accuracy ranges between 80% and 90%.

Another important work in this area is presented in [34]. In this work, authors emphasize the known mistakes and difficulties during labelling the training data. It is known that collecting consistent and reliable data is a very difficult task since some contexts may be marked by users with wrong labels which creates a neccessity for instructing the users before performing any training. In these cases, although the labelled data are unreliable, it still contains valuable information. For this purpose, they propose *Community-Guided Learning* (CGL) which is a framework that trains existing classifiers with unreliably labelled data submitted by different users. Proposed system achieves average performance of 88.7% accuracy and 89.3% in terms of f-measure.

Some of the related works selected from previous studies are summarized in terms of their general properties in Table 2.1 which can be referenced for comparison purposes. The studies are classified according to the methods, and basic properties like activities, external or internal sensors being used, restriction on phone placement or orientation of the sensors, platform, classification algorithms and methods and additionally features being used, number of subjects used for performance evaluation purposes, and performance metrics. As we mentioned, in most of these studies static models are generated through offline training and these models are tested in terms of classification on mobile phones unlike our approach in this thesis. General performance metric is the recognition accuracy which is highly affected from system parameters as indicated in the compared studies.

The most commonly used sensor is the acceleration sensor. WiFi, GPS or any other sensors are added to improve the sensing power and accuracy of the results. *Sitting, Standing, Walking, Running, Driving*, and *Bicycling* are the common activities being targeted to be recognized in the applications. Additionally, commercial targeted applications include more activities to be recognized as in [12]. There are also other studies which target to recognize contextual information, such as location, environmental audio data, of the users as in [31]. Most of the classification algorithms which have online processing capabilities are implemented on Nokia N95 which is one of the first mobile phones with sensing capabilities. Only in [10], classification algorithm is implemented for iPhone. Additionally, Cenceme application detailed in [12] is firstly implemented for Nokia phones and it is later ported for iPhone as well. Most of the classification algorithms implemented on Nokia phones benefit from WMLT in the offline training phase. On the other hand, there are only few works in which the performance of the system is evaluated on a different device model as in [35].

As mentioned, we are interested in developing a real-world application where the user installs the application and does not require to deal with the burden of the training phase, such that the application can be pretrained, or trained quickly without the requirement of a long training phase, and is ready-to-use. The alternative to our approach is delivering an application that initially does not work but *learns* the user's behaviour upon use, which certainly can be a complex and tedious process, as identified in [36]. So our question is "Can we recognize the activities with only a limited set of training data and even without any training data in a user-independent way?" This question is investigated in Chapter 4 where we evaluate the performance of classifiers in a user-independent manner.

Table 2.1. Comparison of Related Works.

Ref	Sensor	Device	Activities	Nr. of Users	Features	Classification Techniques	Orientation	Run Environment	Performance Results
[35]	Accelerometer	Sony Ericsson w715	Transitions' recognition between the states of stand-to-sit and sit-to-stand	12 subjects	Stand up - Sit Down transitions	Waveforms	Anywhere	Trained at desktop and classified at phone	1)One subject recognition rate 100% 2) No false detection detected. 3)Recognition rate 70%. Recognition rate varies between subjects wearing wide trousers or jeans. Subjects wearing jeans have %90 accuracy
[31]	Garmin eTrex Venture GPS, Suunto X6HR wrist-top computer	Embla A10 19-channel recorder, iPaq PDA, Nokia N95	Outdoor bicycling, soccer playing, lying, nordic walking, rowing with the rowing machine, running, sitting, standing,walking	12 subjects	Hip and wrist acceleration signals and the heart rate signal	MD, KNN (k=10), SVM. Feature Selection: SFS, SFFS, ALC	Hip, Wrist, Heart	Trained at desktop and classified at phone	Ranging btw. 70% and 80% depending on the classifier and feature selection methods used
[8]	Accelerometer, microphone, wifi, GPS	Nokia N95	Walking, vehicle, resting, home talking, home entertaining, working, meeting, office-loud, place-quiet, place-speech, place-loud	10 subjects	3 Characteristic features that defines each of state: location, motion, background sound, Wifi: Mac addresses of neighbour devices, GPS: thresholds for moves	FFT based DT	Anywhere	Trained at desktop and classified at phone	%92.5 accuracy, %75 battery lifetime gain
[9]	Accelerometer, GPS	Nokia N95	Stationary, walking, biking, running, motorized transport	16 subjects	Magnitude of 3 axis of accelerometer, mean, variance, energy, DFT energy coefficient btw 1-10 Hz based on magnitude of force vector of accelerometer, speed of GPS	DT followed by first order discrete HMM	Without strict orientation or position requirements	Trained at desktop and classified at phone	Accuracy 93.6%. In outdoor setting, running classifier resulted 8.27 hours of operation

Continued on next page

Table 2.1. Comparison of Related Works (cont.).

Continued from previous page

Ref	**Sensor**	**Device**	**Activities**	**Nr. of Users**	**Features**	**Classification Techniques**	**Orientation**	**Run Environment**	**Performance Results**
[10]	Accelerometer, Nike ipod Sport Kit	Iphone	walking, jogging, bicycling on a stationary bike, and sitting	8 subjects	124 Features; Nike+ipod Packet Payload Features, Accelerometer Magnitude features s.t mean, std deviation, min value , max value, min-max, max-min, Accelerometer Frequency Features s.t. 256 point DFT over last 1.25 sec	Naïve Bayesian Network	Nike+ipod in the shoe, no constraint for iphone	Trained at desktop and classified at phone	Accuracy with person, cross person. Overall accuracy 97%.
[11]	Accelerometer	Nokia N95	Sitting, standing, walking, running, driving, bicycling	4 subjects	Vertical and Horizontal Features s.t. mean,standard deviation, zero crossing rate, 75% percentile, interquartile range, power spectrum centroid and frequency domain entropy	DT, refined by a similarity match from k-means clustering results and smoothed by HMM-based Viterbi Alg. (Weka, DT, NB, KNN, SVM) compared. 10-fold cross validation.	Anywhere	Offline processing	90% with V/H Features and DT, 89% with Magn. Features and KNN, 89% with Magn. Features and SVM
[13]	Accelerometer	Nokia N95	Step Count		Magnitude of acceleration vector	Hill detection and threshold calculation by applying Butterworth Filter	Pocket, Belt clip, Hand	Trained at desktop and classified at phone	
[14]	MotionBand containing magnetometer, gyroscope, accelerometer	Nokia 6630	Resting, typing, gesticulating, walking, running, cycling	3 subjects		Trained with feed forward neural network and classified with neural network	Wrist, Ankle, Hip	Trained at desktop and classified at phone	80% average accuracy

Continued on next page

Table 2.1. Comparison of Related Works (cont.).

Continued from previous page

Ref	**Sensor**	**Device**	**Activities**	**Nr. of Users**	**Features**	**Classification Techniques**	**Orientation**	**Run Environment**	**Performance Results**
[12]	Accelerometer, audio sensors, bluetooth and GPS	Nokia N95	Sitting, standing, walking, running	22 subjects	Audio Classifier feature vector consists of mean and std. deviation of DFT power. Activity Classifier features; mean, std. deviation, no. of peaks per unit time ‘footstep freq.’	Split-level classification and power-aware duty cycling. Audio Classifier: 1) Feature Extraction: DFT(FFT under development) 2)Classification: Machine Learning Algorithm ‘discriminant analysis’	Different places have different effects.	Cenceme app and system consists of a software running on Nokia N95 and back-end infrastructure hosted on server machines.	
[32]	Accelerometer	Samsung SCH-M490	Resting, walking, running	6 subjects	AR coefficients, Kernel Discriminant Analysis is used for feature extraction	Artificial Neural Networks	Shirt’s pocket, jeans’ pocket and coat’s inner pocket	Trained at desktop and classified at phone	Accuracy 96%

3. ACTIVITY RECOGNITION PROCESS

In this chapter, we explain the steps of activity recognition performed in this thesis. We start with explaining the details of accelerometer usage on the Android platform and continue with how the activity recognition process was implemented, along with the details of features extracted, segmentation and classifiers used.

3.1. Built-in Accelerometer of Android Platform

In this thesis, we focused on Android devices and performed our tests by using their built-in accelerometer sensor. The accelerometer is capable of working on different sampling rates which can be given specifically by the user and generate values in x, y, z directions. Figure 3.1 gives an idea of the axes of the built-in Android sensor.

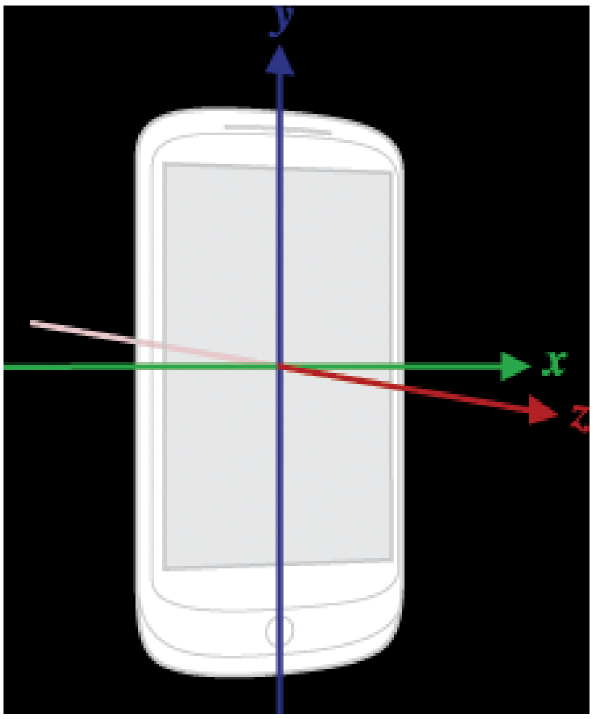


Figure 3.1. Accelerometer built-in sensor axes on Android platform [48].

The coordinate-system is defined relative to the screen of the phone in its default orientation. The axes are not swapped when the orientation of the screen changes. The

X axis is horizontal and points to the right, the Y axis is vertical and points up and the Z axis points towards the outside of the front face of the screen. In this system, coordinates behind the screen have negative Z values. All values are being provided in SI units (m/s^2). Acceleration values are provided on Android phones through an interface called *Sensor Manager* in the following array format;

- values[0]: Acceleration minus G_x on the x-axis.
- values[1]: Acceleration minus G_y on the y-axis.
- values[2]: Acceleration minus G_z on the z-axis.

The sensor measures the acceleration applied to the device (A_d). Conceptually, it does so by measuring forces applied to the sensor itself (F_s) using the relation:

$$A_d = -\frac{\sum F_s}{mass} \tag{3.1}$$

In particular, the force of gravity is always influencing the measured acceleration:

$$A_d = -g - \frac{\sum F_s}{mass} \tag{3.2}$$

Because of this reason, when the device is sitting on the table (and not accelerating), accelerometer reads a magnitude of $g = 9.81 m/s^2$. Similarly when the device is at free-fall and therefore accelerating towards to the ground at $9.81 m/s^2$, its accelerometer reads a magnitude of $0 m/s^2$. Some examples are listed below;

- When the device lies flat on a table and is pushed on its left side toward the right, the x acceleration value is positive.
- When the device lies flat on table, the acceleration value is $+9.81$, which correspond to the acceleration of the device ($0 m/s^2$) minus the force of gravity ($-9.81 m/s^2$).
- When the device lies flat on a table and is pushed toward the sky with an acceleration of $A m/s^2$, the acceleration value is equal to $A + 9.81$ which correspond to

the acceleration of the device ($+Am/s^2$) minus the force of gravity ($-9.81m/s^2$).

Sampling. It is a process that reduces a continuous signal to a discrete signal with a given sampling rate. On Android platform, sampling rates are adjustable. Android API Level 3 and above enables developers to change event delivery rates by passing the rate parameter to the *Sensor Manager* interface. Basically, the *Sensor Manager* interface controls all existing sensors on the device. One can register a sensor event listener with a specific sensor and specific rate to the sensor manager. The rate is the time interval at which sensor events are being delivered. The rate must have one of the following values.

- SensorManager.SENSOR_DELAY_FASTEST : get sensor data as fast as possible.
- SensorManager.SENSOR_DELAY_GAME : rate suitable for games.
- SensorManager.SENSOR_DELAY_NORMAL : rate (default) suitable for screen orientation changes.
- SensorManager.SENSOR_DELAY_UI : rate suitable for the user interface.

There are no standard values for these rates. They are highly dependent on the device model, their capabilities and performance. Except these static values, one can provide the desired delay between events in microseconds to the sensor interface. However, it is clearly stated in [48] that these values are only a hint to the system. Events may be received faster or slower than the specified rate. Usually, events are received faster which is highly dependent on device and inner-sensor capabilities.

3.2. Steps of the Implemented Activity Recognition Process

In this thesis, we applied a combination of the selected methods which are detailed in Section 2.2. We selected applied methods carefully which are most suitable to be performed on smart phones considering limited capabilities. We should emphasize that our aim is to propose a user friendly activity recognition system which performs each step in real-time. While doing it, application performance should never be affected from computational complexities of the proposed methods.

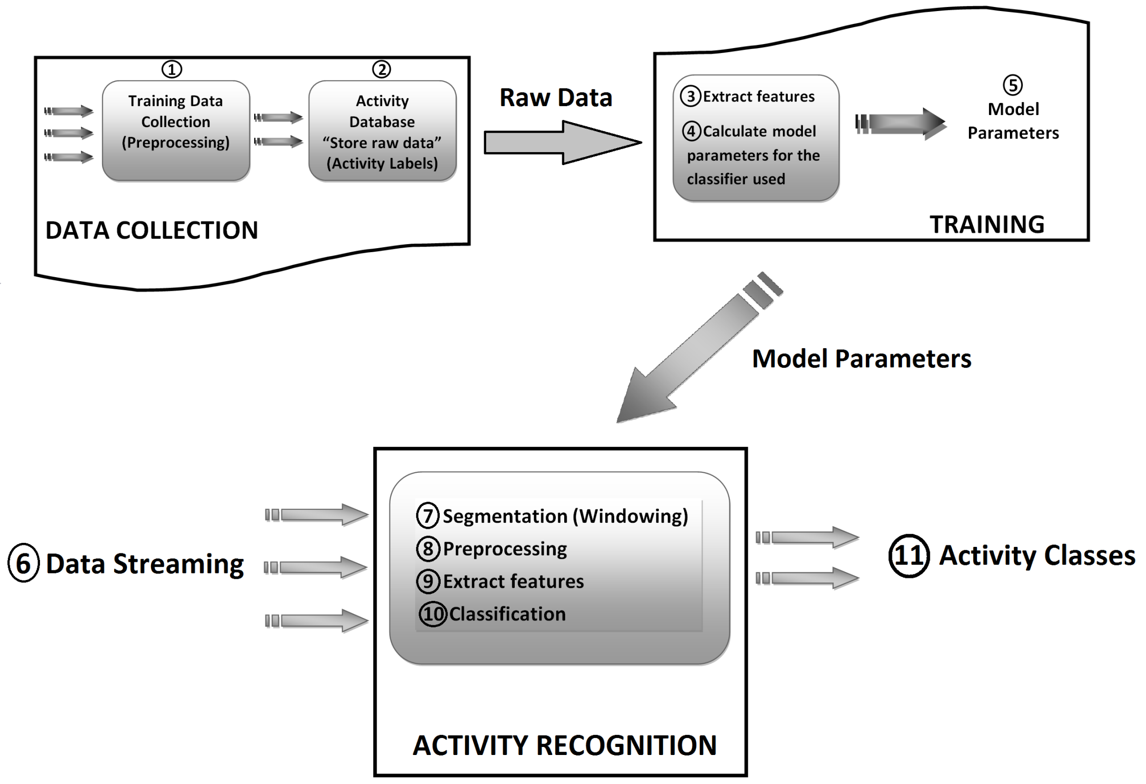


Figure 3.2. System Overview.

The proposed system contains three blocks as indicated in Figure 3.2 which are *Data Collection*, *Training* and *Activity Recognition* respectively. We implemented two different Android applications responsible for each block which are described in Section 3.4. First application is responsible with the *Data Collection* block whereas second application is responsible with *Training* and *Activity Recognition* blocks. Since we use supervised learning techniques, our system requires labeled data collection for approximately three to five minutes for each activity before starting the recognition. Data collection and training blocks are functioning independently from each other so that each process can be repeated at any time, whereas training and activity recognition blocks are executed sequentially. All of the blocks including the activity database reside on the smart phone.

Data Collection steps contain data segmentation and preprocessing as indicated in Figure 3.2. At the end of the data collection process, the processed activity data are stored in the activity database on the smart phone for training and classification purposes. During the *Training* phase, the collected raw data are read from the activity database and necessary features are extracted firstly. Later, model parameters are calculated for each classifier. Model parameters are used as input parameters together with streaming data for the *Activity Recognition* phase. Streaming data are first segmented into timing windows and later, preprocessing, feature extraction and classification steps are applied respectively for each window. System output is indicated at last step (11) which is the activity classes. Details of the main steps for the activity recognition are explained in the following paragraphs.

During the *segmentation* step, illustrated as step 7 in Figure 3.2, we used continuous non-overlapping timing windows presented in Section 2.2. As stated before, this type of windows are suitable when an accelerometer sensor with constant sampling rates is used. Other types of windows are not preferred considering the computational advantage of static windows. We also evaluate the impact of the length of timing windows as one of the most important system parameters which affect the overall performance as indicated in Section 4.3. In Figure 3.3, an illustration of the windowing approaches used in this thesis is presented. The collected raw data are segmented into predefined length of non-overlapping, continuous timing windows. As soon as a timing window finishes (T_1), data collection process continues within the next timing window (T_2). The next steps after segmentation, namely feature extraction, dimensionality reduction and classification start whenever a window is completed. According to Figure 3.3, feature extraction and classification of time window T_1 start at the beginning of T_2. Because of this reason, remaining steps of T_1 must fit into slot T_2, otherwise activity recognition steps would iterate to other windows which causes degradation in the system performance. This is one of the main challenges which should be solved on the target platforms. Additionally, we also applied sliding windows with changing overlapping ratios. The effect of this improvement was tested online and results are compared with those with non-overlapping windows in Section 4.4.

Later, with the *preprocessing* step (1 and 8 in Figure 3.2) and apply a simple

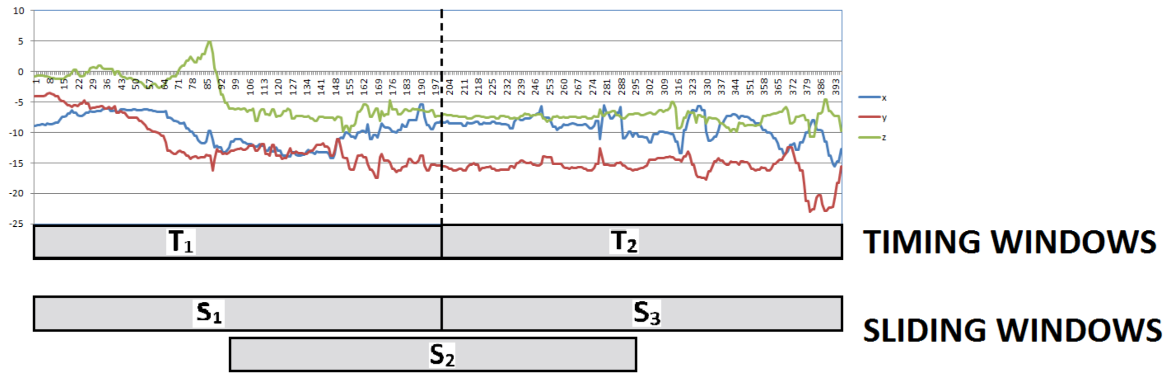


Figure 3.3. Timing windows with 2 seconds at 100Hz.

discretized low pass filter to smoothen the signals coming from the accelerometer which is described in Equation 3.3. The smoothing factor α is calculated as $\frac{t}{t+\Delta_T}$ wheras t is the time constant of the low-pass filter and Δ_T is the event delivery rate.

$$y_i = \alpha x_i + (1 - \alpha) y_{i-1} \qquad \text{where} \qquad \alpha \triangleq \frac{t}{t + \Delta_T} \tag{3.3}$$

In the *feature extraction* step, given as step 9 in Figure 3.2, we used simply extracted time domain features: *mean, maximum, minimum* and *standard deviation* since we perform both training and classification phases online on the smart phone. This decision is mainly because of the limited capabilities of mobile phones as discussed previously. Since these features were commonly used in the previous studies, using the same feature set makes it easier to compare our findings with the similar studies. Moreover, because of the fact that this thesis is mainly focused on classifier performances rather than the performance of feature selection techniques, the detailed evaluation of feature selection methods is out of the scope of this work. Feature selection methods and effect of other features with different combinations on system performance are being investigated in another ongoing thesis [49].

Lastly, for the classification step 10 in Figure 3.2, we used Naïve Bayes, clustered KNN and Decision Tree classifiers which are described in Section 3.3. Their performances are compared in Section 4.3. Among other classifiers, KNN is challenging to be performed online because of its high complexity. For this purpose, we applied the dimensionality reduction step by using the K-Means Clustering method during KNN tests which we call Clustered KNN. Implementation details are provided in Section 3.3.3.

3.3. Classification Methods

Processed data during activity recognition steps identified in Section 2.2 can be used as input parameters for the classification step. In this section, we describe the selected classification methods in detail. For the performance evaluation, we mainly selected Naïve Bayes, Min Distance, KNN and Decision Tree classifiers considering limited processing and storage on smart phones. Other classifiers like HMM, GMMs or ANN are excluded mainly because of their computational complexity either at training or classification steps. In the future, online classification systems may benefit from the strengths of such classifiers as smart phone capabilities improve[1] but the current limitation of smart phones in the market does not allow us to use such classifiers. As we mentioned previously, the training phase is performed offline in most of the related works (Table 2.1) which benefit from strengths of such classifiers. By this way, previously generated static models can be applied afterwards during online tests on the phone. Since we target to apply both training and classification steps on the phone itself, we mainly focused on these classifiers.

Additionally, it is also difficult to find comparisons of such classifiers using the same parameter settings in the same study in the literature. Often, one of the classifiers is selected and all tests are performed only with the selected classifier. There are few studies which compare and present performance of more than one classifier implemented on similar platforms [9]. Because of this reason, we additionally aim to

[1]Newly released Samsung Galaxy SIII smart phones has 1.4GHz Quad Core Processor with Android v4.0.4(Ice Cream Sandwich) OS

present comparisons of selected classifiers in terms of their performance in this thesis.

3.3.1. Naïve Bayes

Naïve Bayes is a probabilistic classification model. Basically, it is based on the Bayesian theorem with independence assumptions. It ignores possible dependencies, namely correlations, among the inputs and reduces a multivariate problem to a group of univariate problems. Since we don't know underlying principle which generates the data, we are not able to calculate it deterministically. Decisions can be made only based on observable values, in our case training data. According to Bayesian rule probability of the reading X being a member of a class C_i can be formulated as follows;

$$P(C_i|X) = \frac{P(X|C_i) \times P(C_i)}{P(X)} \tag{3.4}$$

$P(C = 1)$ is called prior probability that C takes the value 1, which in our case corresponds to the probability that a reading is from C_i regardless of the X value. It is called prior probability since it is the knowledge we have as to the value of C_i before looking at observables X, satisfying

$$\sum_{c} P(C) = 1 \tag{3.5}$$

$P(X|C_i)$ is called the class likelihood and is the conditional probability that an event belonging to C_i has the associated observation value X. In other words, it is the probability of seeing X as the input when it is known to belong to class C_i . It is what the training data tell us about the class.

$P(X)$ is called the evidence which is the marginal probability that an observation X is seen, regardless of which class it belongs. It can be formulated as follows;

$$P(X) = \sum_{c} P(X, C) = \sum_{i} P(X, C_i) \times P(C_i) \tag{3.6}$$

Combining the prior probability and likelihood in the name of Bayes' rule, we can calculate the posterior probability of the concept, $P(C_i|X)$, after having seen the observation, X.

$$posterior = \frac{prior \times likelihood}{evidence} \tag{3.7}$$

Once we have the posterior probability for each class, then we can finalize the decision. Bayes classifier chooses the class with the highest posterior probability to minimize the error. Choose C_i if $P(C_i|X) = max_k P(C_k|X)$. More details are available in [50].

In this thesis, we have utilized an NB implementation which was already used in one of the previous studies [15]. However, in that study for offline processing purposes the algorithm was implemented in Matlab so that we needed to import the same code to Java which can be used for our online classification purposes on Android phones. One minor improvement made in [15] was to exploit the temporal correlation of the readings. Since the readings are frequent, two subsequent readings are likely to be generated by the same action. Therefore a multiplication step is added to the system, in which we multiplied the probability that a reading is a member of class i with a coefficient α, only if the previous reading was determined to be a member of that class. This is called the feed forwarding approach.

Additionally, the previous implementation was just grouping the activities as fall or not falls. It is revised so that the new framework is able to distinguish multiple activities and mark them respectively which correspond, in our case, to five different activities. At the end of the classification process, we get *precision, recall, accuracy* and *f_measure* values immediately.

3.3.2. Minimum Distance and KNN

The minimum distance classifier is used to classify test data according to training data set to minimize the error rate between test and training data sets. The distance of the data to training sets is defined as an index so that the one with the minimum distance is identical to the maximum similarity. There are different methods being used for this procedure such as Euclidean, Normalized Euclidean and Mahalanobis distance.

Differences between Euclidean and normalized Euclidean distances are illustrated in Figure 3.4 which is adapted from [51]. According to this figure, unknown pixel data X is nearer to class B by the Euclidean distance, but it is better to clasify to class A when the normalized Euclidean distance is used.

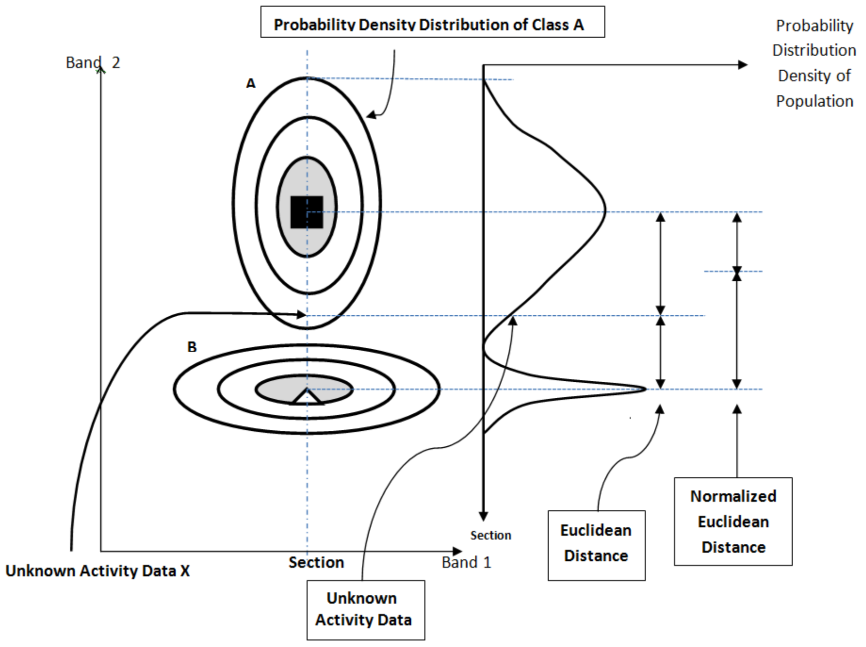

Probability Density Distribution of Class A
Probability Distribution Density of Population
Band 2
A
B
Unknown Activity Data X
Section
Band 1
Section
Unknown Activity Data
Euclidean Distance
Normalized Euclidean Distance

Figure 3.4. Normalized Euclidean Distance.

In this thesis, we always used the *Euclidean Distance* to evaluate the distance between any two points. Basically, the Euclidean distance between any two points p and q is the length of the line segment connecting them $(\bar{pq})$; so $d(q(i), c(j)) \triangleq q(i) - c(j)$ if the sequences are one dimensional and $d(q(i), c(j)) \triangleq \sqrt{(q(i)_1 - c(j)_1)^2 + (q(i)_2 - c(j)_2)^2}$ if the sequences are two dimensional etc.

K-Nearest Neighbor classification is similar to the minimum distance procedure. Basically, the testing data are being compared with all training data in terms of their distances. The Euclidean distance can be used during this procedure. After distance calculations, k nearest neighbors to the test data are being selected. The decision is made for the class set which has more sample in the last data set with k samples in it. Although this method is analytically tractable, simple to be implemented, it requires large storage requirements. Besides that, high computational burden and being highly susceptible to the curse of dimensionality are the other disadvantages of the K-Nearest Neighbor.

3.3.3. Clustered KNN

In the literature, it has been reported that the minimum distance classifier does not perform well when used alone [9]. Additionally, KNN results are always better than the minimum distance in terms of accuracy. However, KNN is not an online classifier since it requires high computational burden and especially considering the limited resources on smart phones, it does not appear as a preferable method.

Considering these mentioned facts, we propose to combine the advantages of the two classification methods and propose to use Clustered KNN for online classification. According to the new method, called Clustered KNN, the training data are first preprocessed and four features, which are average, minimum, maximum, and standard deviation, are extracted. After this step, we apply additionally the dimensionality reduction step, namely K-means clustering to these extracted feature sets. During this process, we aim to limit the sizes of minimum, maximum and average sets so that we decrease the computational complexity of the online system during the classification step. Predefined cluster sizes are selected as 10, 50 and 100 in this thesis.

Preprocessing in Clustered KNN. In the preprocessing step, our objective is to define activity sets from the training data based on the mentioned features. For the training data sets, we do not have any limitation in terms of the sample count. Additionally, we aim to decrease the burden of comparisons with the training set generated within the preprocessing step for an online classifier. In this way, instead of comparing all the data in the training set, we compare the test data only with the compact training data set that we selected from the original training set which represents the original data. During the preprocessing step, compact training sets are created for

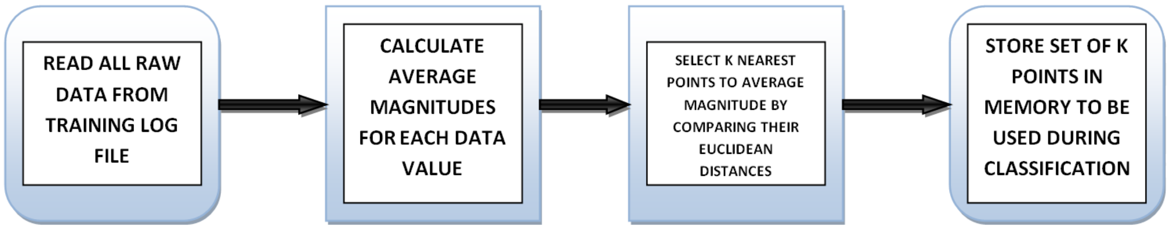

READ ALL RAW DATA FROM TRAINING LOG FILE
CALCULATE AVERAGE MAGNITUDES FOR EACH DATA VALUE
SELECT K NEAREST POINTS TO AVERAGE MAGNITUDE BY COMPARING THEIR EUCLIDEAN DISTANCES
STORE SET OF K POINTS IN MEMORY TO BE USED DURING CLASSIFICATION

Figure 3.5. Preprocessing for the average feature.

each feature and for each activity. For each feature, except the standard deviation, K data points are selected from the training data. This process is summarized in Figure 3.5 for the *average* feature and just for one activity. In general, for the minimum feature set, K *minimum* data points are selected from the training data. Similarly, we create a *maximum* set by selecting the K maximum data points. For the third set, called average set, the average value of the training data is calculated and the nearest K data points are included. These sets are created for each activity seperately. Lastly, we create another set by calculating standard deviation of each activity using the previously collected training data. As an example, if K is selected as 10, we have in total 40 samples after the preprocessing step for four activities per feature except the standard deviation. K value is an important system parameter and when K is smaller, the computational complexity and classification execution times decrease. On the other hand, we expect a decrease in accuracy of the results with smaller K values. Because of this reason, there is an important tradeoff between accuracy and execution time considering the value for K which impact is analyzed in Section 4.5.

Classification in Clustered KNN. In the classification step, we collect test data during a predefined window size, i.e. we segment the data.[2] After the window is filled, classification starts, and *average, minimum, maximum, standard deviation* values of

[2]Same segmentation procedure is also applied to other classifiers whose performance is evaluated in Chapter 4

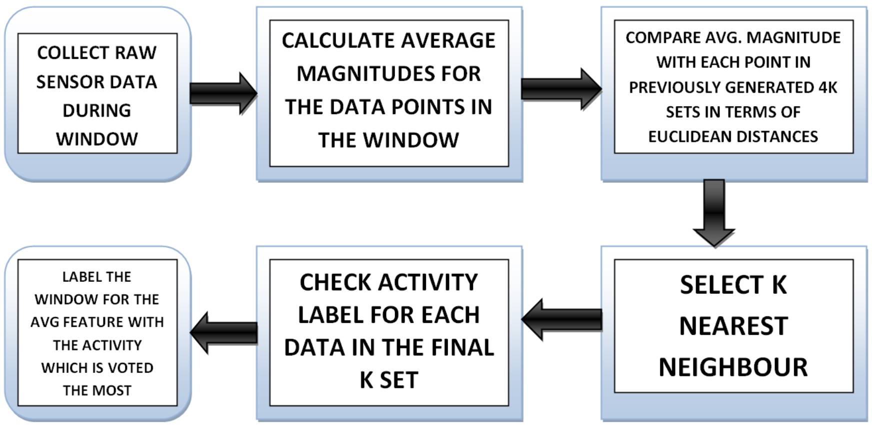

COLLECT RAW SENSOR DATA DURING WINDOW
CALCULATE AVERAGE MAGNITUDES FOR THE DATA POINTS IN THE WINDOW
COMPARE AVG. MAGNITUDE WITH EACH POINT IN PREVIOUSLY GENERATED 4K SETS IN TERMS OF EUCLIDEAN DISTANCES
SELECT K NEAREST NEIGHBOUR
CHECK ACTIVITY LABEL FOR EACH DATA IN THE FINAL K SET
LABEL THE WINDOW FOR THE AVG FEATURE WITH THE ACTIVITY WHICH IS VOTED THE MOST

Figure 3.6. Classification process for average feature.

the data in the window is calculated. These values are compared one by one with the values in the compact training sets which were created during the preprocessing step. K nearest samples to the test data are selected from the training sets and voting is made by checking the final list of activities. We label the data in the related window as the activity for which we have the maximum amount of data in the final K set. For instance, if K is 10 and the final list is as 1 1 5 3 (1 vote for running, 1 vote for walking, 5 votes for sitting and 3 votes for standing) for the average feature, then the activity is labeled as sitting according to the average feature. This process is summarized in Figure 3.6 for the average feature. The same process is applied to *minimum* and *maximum* data sets as well. We make one last comparison for the *standard deviation* coming from the related window with the *standard deviation* of each training set for different activities. The one which is closer to the *standard deviation* of that window is selected as the recognized activity by the *standard deviation* feature. At the end, we have four labels coming from voting results of each feature. We label the window as the activity for which we have the highest number of votes and finalize the classification.

3.3.4. Decision Tree

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including event outcomes, resource costs, and utility. It consists of nodes and branches which enables us to help make reasonable choices and decisions by assigning specific numerical values to each alternative considering uncertainties, costs and payoffs. Decision trees are widely used in cognitive science, artificial intelligence, data mining, statistics, medical diagnosis, formal problem solving, machine learning or game theory.

In the literature, there are many decision-tree algorithms. The main decision tree algorithms can be listed as follows:

- ID3 algorithm [52]
- C4.5 algorithm [53]
- C5.0 algorithm [54]
- CHi-squared Automatic Interaction Detector (CHAID) [55] which performs multi-level splits when computing classification trees
- Multivariate adaptive regression splines (MARS) [56]: extends decision trees to better handle numerical data

ID3 is one of the first DT implementation developed by Ross Quinlan. Algorithms C4.5 and C5.0 are iterations of ID3 and have some improvements on each other. Algorithms constructing decision trees are using the *top-down approach* by choosing the next *best* variable at each step which splits the existing sets at that step into well defined homogenous subsets. There are different methods to decide the next best variables. *ID3*, *C4.5* and *C5.0* DT algorithms use *Information Gain* as given in Equation 3.8 which is based on the concept of entropy in information theory.

$$I_E(f) = -\sum_{i=1}^{m} f_i \log_2 f_i \tag{3.8}$$

There are many reasons for us to choose the decision tree for classification. Its simplicity to understand and interpret, remarkable performance at related works [9], possibility to validate the results easily with simple statistical tests, robustness and performance are the main advantages of decision trees. In this thesis, we aim to benefit from such advantages of the decision tree classifier. The most important feature is that the decision tree is able to handle even large data sets easily by using standard computing resources which has utmost importance in terms of this work considering online classification and training processes. In this thesis, we have used $C4.5$ DT algorithm based on $jaDTi$ which is a Java implementation of decision trees. Its source code is freely available in [53].

3.4. Android Implementation of Activity Recognition Process

In this work, two separate applications are implemented for different purposes which are called as *Activity Logger* and *Activity Recognizer* respectively. Activity Logger application is being used to collect sample data for each activity and Activity Recognizer application is being used to classify user activities according to the training data collected by the other application. Different methods and techniques specifically explained at previous sections are used during classification steps.

3.4.1. Activity Logger

As stated above, purpose of this application is to collect sample data for each activity separately. Interface of the application is user-friendly and does not require any expertise to be used. Before starting the application, user selects the activity to be performed, press the start button, puts the phone into the pocket and starts to perform the related activity. Any data being collected after that time is being labeled with the activity that the user selected. It expects the user to perform only that type of activity until the stop button is pressed. The user interface of the application is presented in Figure 3.7.

The application collects only the acceleration data by using the *Sensor Service* interface. A low-pass filter is applied to the raw data to isolate the force of gravity and measure real acceleration values of the device as stated in Section 2.2. The accelerometer sensor is registered to *Sensor Manager* interface with a static sampling rate of 50 Hz.

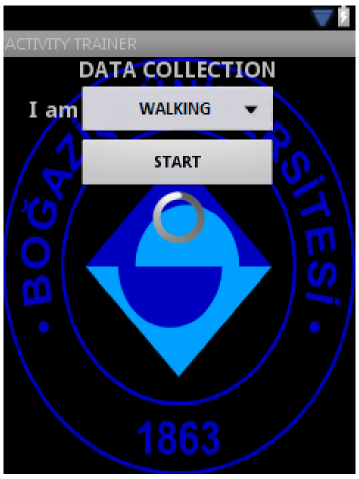


Figure 3.7. Interface of Activity Logger.

For each activity, the application creates different training data files in which raw data from the 3-axes of the accelerometer is being logged after preprocessing step. All readings are gathered in one folder under the SD card named as *HumanTracking*. At any time, any file under this folder can be reached and sampled data can be analysed. Each activity-associated data set is written to separate text files which are named as *training_ActivityLabel* under this folder. Before starting the activity recognition tests, a few minutes of training data for each activity should be collected by each subject.[3] The training data that belong to the activity can be collected at any time without the need of continuously performing the related activity. If a user continues to collect data after a while, new data are written starting from the end of the previously collected data set instead of replacing the already existing data in the file. Figure 3.1 presents the format of the training data being written to the associated file which is called in this specific case as *training_1.txt*. Each column in these files is separated with a space and each of them represents x, y, z acceleration values and the activity label respectively.

[3]In fact, we also performed tests when no user training data is available but the training data from other users are used.

Table 3.1. Representation of Training Data.

1	-8.5862665	-14.459362	-7.5402246	1
2	-12.040388	-20.681137	-10.4168415	1
3	-10.384153	-22.642466	-3.1272316	1
4	0.8281174	-20.528587	1.6889238	1
5	-10.144435	-19.76585	-5.088562	1

3.4.2. Activity Trainer and Recognizer

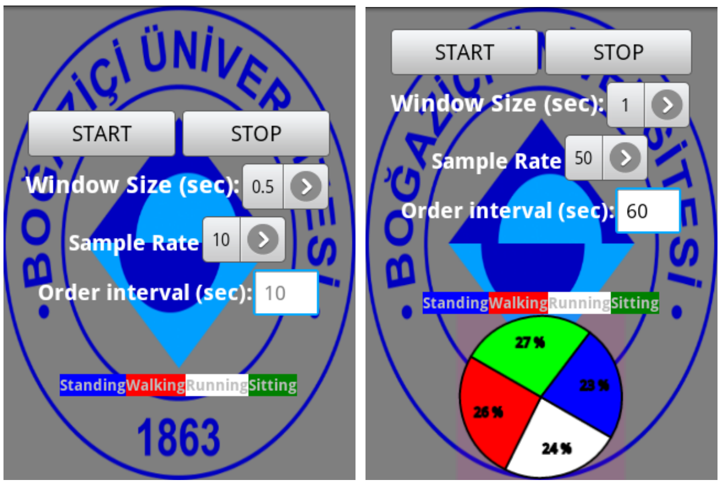
This application basically processes the training data and performs the classification. Training data sets created by the activity logger application are processed at each start up of the application only once. By this way, users can collect the training data any time and they can start to use the activity recognition application right after training data sets are created or updated without the need of any extra manual operation. These two processes highly depend on the classification method which will be used for activity recognition. The main purpose of the training data processing at the start up is to prepare application for the classification method which will be used online. First, the application extracts the necessary features of training sets for each activity according to the classification method being used and calculates the model parameters for each classifier. Depending on the size of the training set and the processor performance of the phone, this step may take only a few minutes at most. These training periods match perfectly with our goal in this thesis by making the application more user-friendly and easy to use by everyone.

As mentioned, we have designed the training process to be fired at each start up of the application since new training data can be collected at any time by the user. Expanding the training data sets will increase the accuracy. However, because of the limited computation power and memory of the smart phones, we had to restrict sizes of training files. According to our initial tests with T-Mobile Android phones, maximum 5000 lines of training data in each file would be a good compromise which can be parsed and processed by the phone easily. However, more powerful phones in the current market like Samsung Galaxy SII can process larger training files with 20000 up to 30000 lines easily which leads to better results. In our system, we have restricted the size of the training files up to 5000 lines to be able to compare the results of any phone included in our study. However, in the future, we are planning to set the minimum Android API level (expression of an application's compatibility with one or more versions of the Android platform) for this application to be used so that we can benefit from the strengths of advanced smart phones. We are also planning to apply changes in the data collection process. Our idea is to collect and replace new data only if it perfectly fits the previously collected data sets and exclude the ones which are regarded/considered as noisy data, even though we have a limitation on the sizes of the training set. By this way, our aim is to create more useful labelled training data.

As stated previously, NB, clustered KNN and DT classifiers are implemented in this thesis. For NB, each training data set is being read from associated files and mean, covariance arrays are calculated respectively during the preprocessing step to be used later during the online classification. On the other hand, for clustered KNN and DT, average, min, max data sets and standard deviation are calculated to be used later at online classification steps.

In the second stage, activity recognition is performed using the selected classifier. Main layout of the *Activity Recognizer* application is presented in Figure 3.8. The user interface of the application allows the user to select the system parameters, such as the sensor sampling rate, window size, order interval (explained later in this section) and extracted features. These features are mainly added for test phases to track the performance of the classification methods. After providing the necessary system parameters to the application, one can manage it simply by using start and stop buttons.

Possible input values for each input field at *Activity Recognizer* application are explained below:



Window Size (sec): 0.5
Sample Rate 10
Order interval (sec): 10

Window Size (sec): 1
Sample Rate 50
Order interval (sec): 60

Figure 3.8. Interface of Activity Classifier.

Window Size. It is provided to the system in terms of seconds. It is the time interval during which data are being collected without any classification. Application forces users to select one of the existing values at drop down menu which are 0.5, 1 and 2 seconds respectively. Other values could be added but as we explain in Section 4.2, these intervals are reported to be the most suitable intervals to recognize *standing, walking, running, sitting* and *biking* activities.

Sample Rate. It is the sampling rate provided directly to the internal accelerometer sensor. During initial tests, we used static sampling rate values of Android API like GAME, UI or NORMAL. Later, the system was modified so that we provide the desired delay between events in microseconds directly. Input values at layout can be selected as one of the predefined values in the drop down menu which are defined as 10, 50, 100 all indicating the delay between two events in unit of milliseconds. The accelerometer sensor is being registered to *Sensor Manager* interface directly with the values being taken from this user input.

Order Interval. We have created a test scenario before performing the system tests. According to this scenario, all tests are performed in a specific sequence with a specific activity order in a predetermined period of time. Activity order is defined as *RUNNING , WALKING, BIKING, STANDING, SITTING*. All experiment subjects performed these activity sets respectively. The predefined time during which activity is performed is called Order Interval in this thesis. The Order Interval parameter can be provided by user in unit of seconds to the system.

In order to monitor the recognition performance of the classifier, the ground truth data, i.e. which activity is actually performed by the user, is logged. For this purpose, the application gives voice commands repeatedly to the user to perform an activity. Application itself gives voice orders for each activity to be performed whenever an order interval is completed. Voice orders which are currently defined in the system are presented in Table 3.2. Finally, using these ground truth values, i.e., activity tags, the activity recognition performance and other performance metrics of the classifiers are calculated.

Table 3.2. Existing Voice Orders at Activity Recognizer Application.

STATIC VALUE	**VOICE COMMAND**
TEXT_TO_SPEECH_RUN	Start running!
TEXT_TO_SPEECH_STANDING	Start standing!
TEXT_TO_SPEECH_BIKING	Start biking!
TEXT_TO_SPEECH_SITTING	Start sitting!
TEXT_TO_SPEECH_WALKING	Start walking!

Important sampling data are collected in separate log files for further analysis, and written under the *HumanTracking* folder with different file names. They are easily accessible under the SD card so that one can check these files easily after processing is completed. Log files are detailed in Appendix A.

After user clicks on the stop button, overall results are calculated and written directly to the files detailed in Appendix A. After all results are derived, a pie chart with percentages of the overall classification is being shown on the main layout as presented in Figure 3.8. Lastly, all running services, as well as the sensor service stop running.

4. PERFORMANCE EVALUATION

In this chapter, we aim to evaluate the performance of the proposed system based on selected system and workload parameters as indicated in Figure 4.1. First we present the performance metrics used in the thesis. Later, we elaborate on preliminary tests and their results which give us the direction for our final tests. Based on the preliminary tests we designed our final experiments through which we evaluate the performance of the classifiers and performance of the overall system in terms of CPU and battery usage. Lastly, we compare our results with tests performed offline both in classification and training phases. We conclude this chapter by providing the final results which evaluate user dependency of training data sets.

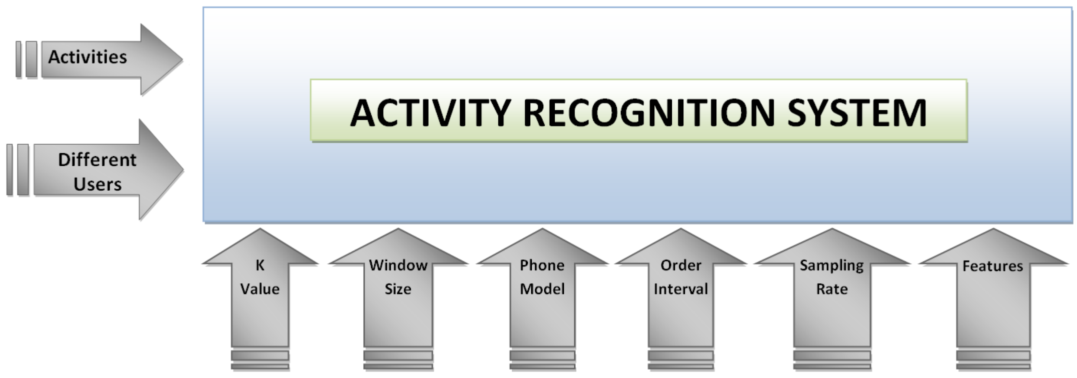


Figure 4.1. Workload and System Parameters.

Main tests were performed in two batches with a different set of activities. First tests were performed with *running, walking, standing*, and *sitting* whereas in the second group of tests we added the *biking* activity. Test designs will be detailed in Section 4.2 with the selected system and workload parameters. In all the tests, subjects firstly collected training data for each activity, around 4-5 minutes for each set, using the *Activity Logger* application described in Section 3.4.1. Later, voluntary subjects were asked to repeat a predefined test scenario which is presented in Table 4.1. During these tests, subjects used *Activity Recognizer* application which automatically gives the activity order to the subject with a voice command whenever a transaction should be performed. As mentioned, we were able to hold ground truth of performed activities during tests and collect reliable results. With the help of the ground truth, we are able to evaluate the real performance of the proposed system as well.

Table 4.1. Test Scenario.

Order	**Action**
1	Running
2	Walking
3	Standing
4	Sitting
5	Biking

4.1. Performance Parameters and Metrics

In this section, we describe the factors which may affect the outcome of the recognizer. We use the two-phase experiment design described in [57]. In the first phase which we call the preliminary test phase, we aimed to create a stable test environment by selecting a proper set of system factors. Then in the second phase of experiments, the factors which may affect the outcome are investigated deeply.

Factors which may affect the results are presented in Figure 4.1. In this figure, activities and different users can be grouped as workload parameters whereas sampling rate, phone models, classifiers, features, window size, order interval and K value for Clustered KNN can be defined as system parameters. After our initial studies, we identified several factors that may affect the performance of the activity recognition process on smart phones:

- *Phone Model:* Device model is highly important in the sense of sensors being embedded to the device and device capabilities. Sensor hardware changes according to the model and its manufacturer which may directly affect the performance as different sensors may have different accuracy and noise characteristics. Additionally, computational power of the device should be enough to handle the selected system parameters and test cases appropriately. Thus, device selection is important.

- *Activity:* It is the target action being performed during tests which we try to recognize. Selected activity sets are also important in terms of accuracy of the system. We aim to recognize the activities that are mostly investigated in similar studies (given in Table 2.1) for comparison purposes. Adding new activities to the current system may affect the recognition performance.
- *Classifier:* Classification is the most important step at activity recognition process as stated in Section 3.3. Type of selected classifiers play an important role on the system results. We have selected classifiers which are most suitable with the online system we proposed.
- *Feature:* Features are the signatures of the activities which have an important effect to identify the activities and affects the results directly.
- *Sampling Rate:* It is the rate at which the data is gathered. It affects the ability of the accelerometer to capture the necessary information for target activities.
- *Window Size:* It is the duration during which we only collect data without performing any classification. Each human activity has a pattern except stationary activities. Because of this reason, collected data during a window plays an important role to identify activities.
- *Order Interval:* Order Interval is the duration of activity to be performed.
- *Test Cycle:* Test cycle consists of activities to be performed according to a predefined test scenario. Depending on our tests, one cycle consists of either four or five activities.
- *K Value:* It is the cluster size used for Clustered KNN tests. This parameter is directly associated with computational complexity of the system and accuracy of the results.

Table 4.2 indicates the classifier-based system parameters being applied during our tests. As presented, we applied K value as the cluster size parameter only during clustered KNN tests to decrease the computational complexity of KNN classifier.

Table 4.2. Classifier Based System Parameters.

CLASSIFIER	**SYSTEM PARAMETERS**
Naïve Bayes	*Window Size, Sample Rate*
Min Distance	*Window Size, Sample Rate*
Clustered KNN	*K Value, Window Size, Sample Rate*
Decision Tree	*Window Size, Sample Rate*

Throughout the tests, performance evaluations of our models are made using *precision, recall, F-measure,* or *accuracy.* These values can be computed using the confusion matrix shown in Table 4.3.

Table 4.3. Confusion matrix showing TP, TT and TI for each class [58].

		INFERRED				
		1	**2**	**3**	**4**	
	1	TP_1	ϵ_{12}	ϵ_{13}	ϵ_{14}	TT_1
GROUND	**2**	ϵ_{21}	TP_2	ϵ_{23}	ϵ_{24}	TT_2
	3	ϵ_{31}	ϵ_{32}	TP_3	ϵ_{34}	TT_3
TRUTH	**4**	ϵ_{41}	ϵ_{42}	ϵ_{43}	TP_4	TT_4
		TI_1	TI_2	TI_3	TI_4	Total

More detailed confusion matrices will be provided later in the results section while providing our test results. In the confusion matrix, the rows show the ground truth labels as provided by a human annotator, while the columns show the labels inferred by the model. The diagonal of the matrix shows true positives (TP). The sum of each row provides us the total ground truth of each label (TT). Lastly, the sum of each column gives us total of inferred labels (TI). The precision and the recall are separately calculated for each class and then the average is taken over all existing classes.

In Equation 4.1, precision is calculated as follows:

$$Precision = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TI_i} \tag{4.1}$$

where N is the number of classes. Similarly, recall can be computed as given in Equation 4.2;

$$Recall = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TT_i} \tag{4.2}$$

F-Measure calculation is based on precision and recall values which is provided in Equation 4.3.

$$F - Measure = \frac{2 \times precision \times recall}{precision + recall} \tag{4.3}$$

$$Accuracy = \frac{\sum_{i=1}^{N} TP_i}{Total} \tag{4.4}$$

Since we deal with unbalanced data sets, it is important to use these particular measures. Some classes may appear much more frequent than other classes whereas our measure takes the average precision and recall over all classes and therefore considers the correct classification of each class equally important. Additionally, the overall accuracy is used for comparison which is calculated over all classes by using the true positive values and the total data given in Equation 4.4. The accuracy represents the percentage of correctly classified time slices.

4.2. Experiment Design

4.2.1. Preliminary Test

Before starting with the main tests, we have performed preliminary tests to observe the main difficulties of working in the Android environment. Firstly, we have implemented the NB classifier and performed initial simple tests with different variations by using different window sizes, sampling rates and different subjects. Test setup and parameters are provided in Table 4.4.

Table 4.4. Preliminary Test Parameters.

Parameter	**Value**
Activity	Running, Walking, Sitting, Standing
Phone Model	T-Mobile G2 Touch
Classifier	NB
Feature	Average
Window Size	Changing window size depending on Sample Rate
Sample Rate	Game, Normal, UI
Order Interval	5,10,30,60 (sec)
Test Cycle Count	1,3,5
Windowing	Non-overlapping, continous

All initial test results are gathered in Tables 4.5, 4.6 and 4.7 respectively. Table 4.5 shows the classification results of performed activities of a subject which is trained online by the same subject. In Table 4.6, we provide the results of the system which is trained with different set of data collected from two senior undergraduate students and tested by the subject who performed the first tests. Table 4.7 indicates the results of the system which is trained and tested by a different subject, i.e., the training and testing were done by the same person again but this subject used a phone which was a different model. In all tests we used the Naive Bayes classifier. Window sizes are selected according to sampling rates which represent the data size collected during that window in Tables 4.5, 4.6 and 4.7. We used static sampling rates of the Android platform during these initial tests which were stated in Section 3.1. When we look at the results, we could not get any similar behaviour based on system parameters with different subjects. We have contradictory results between two subjects considering the sampling rates and the window sizes. Additionally, we observe very poor performances at some specific cases. Based on the detailed analysis, we derived the following main conclusions which shaped the final test setup.

Table 4.5. Tests performed on private training set.

Sampling Rate	**Window Size**	**Accuracy**	**F_Measure**	**Precision**	**Recall**
GAME (20 msec)	11	0.39	0.42	0.45	0.39
	22	0.38	0.40	0.41	0.38
	44	0.29	0.21	0.17	0.28
NORMAL (200 msec)	26	0.39	0.39	0.40	0.39
	52	0.41	0.42	0.44	0.41
	104	0.37	0.34	0.32	0.37
UI (60 msec)	16	0.44	0.44	0.44	0.44
	32	0.42	0.42	0.43	0.42
	64	0.37	0.29	0.24	0.37

Table 4.6. Tests performed on different training data set.

Sampling Rate	**Window Size**	**Accuracy**	**F_Measure**	**Precision**	**Recall**
GAME (20 msec)	11	0.44	0.41	0.37	0.45
	22	0.27	0.30	0.32	0.27
	44	0.45	0.41	0.38	0.45
NORMAL (200 msec)	26	0.39	0.36	0.33	0.40
	52	0.41	0.38	0.34	0.42
	104	0.33	0.34	0.35	0.34
UI (60 msec)	16	0.41	0.37	0.33	0.42
	32	0.27	0.24	0.22	0.28
	64	0.26	0.24	0.22	0.27

Table 4.7. Tests performed on private training set by a different subject.

Sampling Rate	**Window Size**	**Accuracy**	**F_Measure**	**Precision**	**Recall**
GAME (20 msec)	11	0.46	0.53	0.62	0.46
	22	0.50	0.57	0.66	0.50
	44	0.38	0.46	0.58	0.38
NORMAL (200 msec)	26	0.31	0.47	0.56	0.40
	52	0.54	0.50	0.58	0.44
	104	0.30	0.09	0.08	0.12
UI (60 msec)	16	0.40	0.42	0.44	0.40
	32	0.51	0.59	0.66	0.64
	64	0.47	0.51	0.52	0.50

The most important result that we observed after the preliminary tests was the difference of total number of collected and processed data between devices. With different devices we have had really different results. These differences were mainly caused by using already defined sampling rates of sensor manager and device capabilities since these values are just hint for the system and highly depending on the performance of the phone models [48]. Because of this reason, we decided to use static sampling rates in our main test sets and revised the sampling rate such that it could be given by the user input to the system in terms of the time unit (millisecond) between each sensor event. 10, 50 and 100 milliseconds are decided to be used for the next tests. While deciding on the sampling rates, we choose 10 msec sampling interval as the basis since most of the smart phones are not even capable of rising sensor events with this interval. On the other hand, sampling rates between 10 Hz and 100 Hz are indicated as the most suitable rates in the literature for human activity recognition.

Another important change is applied to the window size. During the initial tests we used window sizes which are defined according to the static sampling rate of Android devices. Basically, it counts the sample size collected and finishes windowing as soon as the predefined size is achieved. So after the change on sampling rates, we started to use the time interval in unit of seconds which is more suitable for our tests. To control window sizes, a new thread was implemented at the application level. The same thread is controlling the classification as well. According to the new model, test data are being collected during a window of size x seconds; which is provided by the user. After the window is completed, all collected data during that interval is being classified. We may call the data collected during the window size as the test data set. Possible window sizes are selected as 0.5, 1, 2 seconds in our main tests.

Another important conclusion we derived after our initial test runs was the importance of the order interval on the accuracy of the results. During our initial tests we selected different order intervals which ranges between 5 and 60 seconds. It has been observed that as the order interval increases towards 60 seconds, accuracy of the results improved in parallel. The main reason of this change at accuracies was the transitions between activities during tests. We have more transitions during a test where the order interval is selected as five seconds than the one which has the order interval selected as 60 seconds. Each transition between activities causes unbalanced data in test set for corresponding windows which results in wrong results. Because of this reason, we decided to use a longer order interval which is 60 seconds during our final tests. We did not increase the order interval further since increasing the order interval prolongs the tests and does not improve the final results. It is already difficult to continuously perform related activities for 60 seconds long considering especially running and biking cases. We also improved our system by defining guard intervals around each transition. Basically, the first and the last two windows for each order interval is ignored and not included in the performance evaluation process.

4.2.2. Final Test Setup

As we mentioned, tests were performed in two batches with different set of activities. During the first group of tests, subjects performed four different activities which are *running, standing, sitting* and *walking*. These tests were performed with seven different subjects whose average age is 27 (2 female, 5 male). Each subject performed the same predefined activity pattern *running, walking, standing, sitting* during the classification step. Each test set is performed for four minutes where each activity is performed for 60 seconds. Setup system parameters are listed in Table 4.8.

Table 4.8. Setup parameters for the first group of final tests.

Parameter	**Value**
Activity	Running, Walking, Sitting, Standing
Phone Model	Samsung Galaxy SII, Samsung Galaxy W, Turkcell MaxiPlus5
Classifier	NB, DT, Clustered KNN
Feature	avg, min, max, std dev., avg. acc. at Y axis
Window Size	0.5, 1, 2 (sec)
Sample Rate	10, 20, 100 (Hz)
K Value	10, 50 ,100
Order Interval	60 (sec)
Test Cycle Count	1
Windowing	Non-overlapping, continuous

During the second batch of tests, subjects performed one more additional activity which is *biking*. These tests are performed with nine different subjects whose average age is 29 (2 female, 7 male). During these tests, each subject performed the same predefined activity pattern *running, walking, standing, sitting* and *biking*. Each experiment lasted five minutes where each activity was performed for 60 seconds. Setup parameters for the second group of tests are the same as indicated in Table 4.8 except activity parameters. All subjects carried the mobile phone in the front pocket of their pants during both test and training phases. The same test scenario is repeated nine times with different system parameters based on the window size and the sampling rate. Additionally, K value is varied as 10, 50, and 100 for clustered KNN tests which resulted in 27 tests in total. Window sizes are selected as 0.5, 1, 2 seconds whereas 10, 50, and 100 msec are used for the sampling interval. With these changes, our objective was to examine the effect of the window size and the sampling interval on the performance of the activity recognition.

Each subject performed all the tests on the same Android platform, although different subjects could use different phone models. At the end of the classification process, precision, recall, accuracy and F-measure metrics are calculated and results are written in a final result file. As mentioned, classification results are also shown with a graphical pie chart on the phone screen (Figure 3.8).

All activities during the tests are performed in an indoor[4] laboratory environment as shown in Figure 4.2 except the biking activity. Biking experiments are performed in an outdoor environment with all the subjects using the same bike.

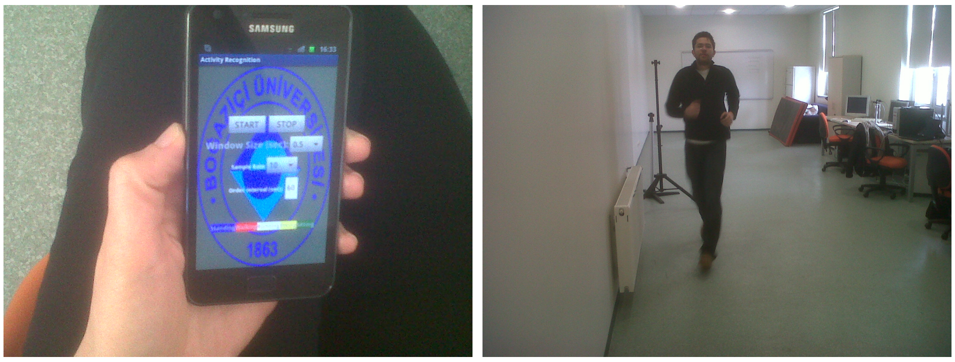


Figure 4.2. Snapshots from the experiments.

4.3. Performance Evaluation of Classifiers

In this section, we explain the results obtained by comparing the performance of the classifiers. Individual performance of the classifiers will be evaluated under each subsection separately. Moreover, as stated before, two batches of tests are performed and their results will be given separately.

[4]We could perform the tests in an outdoor environment as well but because of the weather conditions we preferred to perform the tests in an indoor environment.

4.3.1. Performance of Naïve Bayes

Performance of NB without biking activity. We observe that the accuracy rates ranged from 40% to 66% when we apply the Naïve Bayes classification method which is highly dependent on the system parameters as given in Table 4.10. Naïve Bayes achieved a 66% average accuracy rate for all subjects with different sampling rates and window sizes. Similarly, the performance in terms of F-measure was close to 65% as indicated in Table 4.9. Moreover, the best performance results are obtained with window size of 2 seconds with smaller sampling rates as presented in Table 4.10.

Table 4.9. Comparison of Clustered KNN, NB and DT without biking.

	CLUSTERED KNN	**NAIVE BAYES**	**DECISION TREE**
ACCURACY	92.13%	66.33%	85.52%
PRECISION	92.45%	68.05%	83.56%
RECALL	92.09%	65.10%	81.22%
F-MEASURE	92.27%	65.26%	82.37%

Table 4.10. Impact of window size and sampling interval on the accuracy rates of classifiers without *biking*.

Window size(sec)	**0.5**			**1**			**2**		
Sampling Interval(msec)	10	50	100	10	50	100	10	50	100
Clustered KNN	91.1	90.0	91.4	91.9	**92.1**	90.8	88.9	89.4	91.0
Decision Tree	80.2	85.0	81.8	70.0	79.5	74.1	77.2	78.6	**85.5**
Naïve Bayes	40.9	53.8	52.3	41.7	52.5	51.4	53.7	64.2	**66.3**

Performance of NB with biking activity. After adding the biking activity to the test set, we observed considerable decrease in the activity recognition process in terms of accuracies. It ranges between 33% and 47% in these cases with different window sizes and sampling rates. According to the results, NB performed in average 46% accuracy while recognizing five different activities as indicated in Table 4.11. Its F-Measure performance is nearly 44% according to the results. Similarly with first tests, best results are achieved whenever the window size is selected as 2 seconds as indicated in Table 4.12.

Table 4.11. Comparison of Clustered KNN, NB and DT with biking.

	CLUSTERED KNN	**NAIVE BAYES**	**DECISION TREE**
ACCURACY	72.91%	46.34%	75.33%
PRECISION	77.82%	41.69%	67.32%
RECALL	78.06%	46.42%	73.14%
F-MEASURE	77.82%	43.68%	69.95%

In conclusion, Naive Bayes revealed poor results compared to other classifiers with online classification and training. Mainly, poor results derive from the limited and small training data sets which are a part of the methodology we follow in this thesis. This result is derived from observations stated in Section 4.6, where we explain the performance of Naive Bayes with offline training using 10-fold cross validation.

Table 4.12. Impact of window size and sampling interval on the accuracy rates of classifiers with *biking*.

Window size(sec)	**0.5**			**1**			**2**		
Sampling Interval(msec)	10	50	100	10	50	100	10	50	100
Clustered KNN	44.2	57.1	66.7	48.8	66.1	70.4	58.4	72.1	**72.9**
Decision Tree	63.1	70.6	67.0	54.9	68.3	71.9	66.3	71.1	**75.3**
Naïve Bayes	33.0	43.9	42.7	34.7	41.6	38.9	42.0	**46.3**	43.4

4.3.2. Performance of Clustered KNN

Performance of Clustered KNN without biking activity. The confusion matrix for clustered KNN is presented in Table 4.13 to evaluate the classification performance of the classifier for each activity. Compared to the performance of activities of running, standing and sitting, the classifier presents slightly worse performance for walking where it is sometimes classified as running or standing. However, the overall performance for clustered KNN is around 92% accuracy considering all activities as presented in Table 4.14.

Table 4.13. Overall Confusion Matrix of Clustered KNN without Biking.

		CLASSIFICATION			
		RUNNING	**WALKING**	**STANDING**	**SITTING**
GROUND TRUTH	**RUNNING**	94.1%	4.1%	1.2%	0.6%
	WALKING	15.7%	72.9%	10.8%	0.6%
	STANDING	5.8%	1.0%	91.6%	1.6%
	SITTING	1.6%	0.2%	2.6%	95.6%

We also evaluated the impact of K value on the classification performance of clustered KNN. As expected, increasing the K value affected the accuracy rates positively. We observed on average 87% accuracy with $K = 10$ whereas it increased to 91% when K is selected as 50. Although the accuracy rates were not affected from further increase of K values, classification times significantly increased because of this change[5] . Table 4.14 summarizes the performance of clustered KNN in terms of average accuracy rates with changing K values.

Table 4.14. Average Accuracy Rates of Clustered KNN(%) without Biking.

Window size(sec)	**0.5**			**1**			**2**		
Sampling Interval(msec)	10	50	100	10	50	100	10	50	100
K = 10	87.9	87.8	87.7	88.4	90.3	89.5	88.6	87.8	89.3
K = 50	91.1	90.0	91.4	91.9	**92.1**	90.8	88.9	89.4	91.0

[5]We observed classification times up to 210 msec with Samsung GalaxyII whenever we selected K value as 50 whereas it increased up to 330 msec with K equals to 100

Considering all the performance results, we observed the best results with K value selected as 50. We further analysed the effect of the sampling rate and the window size on accuracies. In general, slightly bad results are observed in cases when the window size is selected as 2 seconds whereas the best results are obtained with the window size of 1 second. For the window sizes studied, as shown in Table 4.14, sampling interval does not have a significant impact on the accuracy results. When we consider the overall effect of all system parameters, we obtained the best results in the case where K is selected as 50, the window size is selected as 1 second and the sampling interval is selected as 50 msec. According to the tests performed with seven different subjects, we obtained an average 92% accuracy rate for this case. Table 4.14 summarizes the accuracy results with changing parameter values.

Performance of Clustered KNN with biking activity. First test batch clearly indicated that increasing K value affects the results positively so that in the second batch of tests, we only used Clustered KNN with K value set as 50. K value is not selected as 100 since it increased the computation times a lot which caused degradation at application performance although it revealed almost the same performance results as K equals to 50.

Adding the biking activity to the test set affected the clustered KNN performance negatively and the system accuracy is decreased from 92% to 73% (We discuss the impact of adding the *biking* activity in Section 4.3.4). Performance of the Clustered KNN is presented in Table 4.12 with changing system parameters of window size and sampling rate. According to the final results, Clustered KNN gave the best results whenever the window size is selected as 2 seconds with the sampling interval 100 msec. Differently from the previous results one can easily see that variance of the results increased substantially in the case where we added the biking activity.

As indicated in Table 4.2, the K value, the window size and the sampling interval is determined as system parameters for the clustered KNN method. Compared to Naïve Bayes, on average, clustered KNN achieved a much better classification performance, around 92% accuracy, precision, recall and F-measure without biking and 73% accuracy with biking activity. Test results show that the accuracy rates for online classification with Clustered KNN method are highly comparable with the previous studies referenced in this thesis and even with the ones which are considering offline classification.

4.3.3. Performance of Decision Tree

Performance of Decision Tree without biking activity. Decision Tree results are presented in Table 4.9 for comparison purposes. We used the same system parameters except the K value during Decision Tree tests. According to the test results, we have the worst results with window size 1 second as indicated in Table 4.10. On the other hand, the Decision Tree performed better whenever we select the window size as 0.5 seconds. We have also highly comparable results whenever we used the window size as 2 seconds. All sampling rates at 0.5 second window size resulted in better performance which ranges between 81% and 85%. We have the best results with the window size of 2 seconds and sampling interval 100 msec which performed nearly 86% accuracy.

Performance of DT with biking activity. Adding biking to the activity set decreased the Decision Tree performance as observed with other classifiers. The overall results are detailed in Table 4.12. In general, results improved whenever we increase the sampling interval and the window size while using Decision Tree as the classifier. The best results are obtained with the window size of 2 seconds and the sampling interval of 100 msec which performed nearly 75% accuracy.

4.3.4. Overall Performance Evaluation of Classifiers

Considering all the results, we can conclude that Clustered KNN performed the best whenever we exclude the biking activity. It nearly performed 92% accuracy to detect the remaining four activities. On the other hand, the Decision Tree achieved also good results with 85% accuracy in the same environment whereas it performed nearly the same or better accuracies than Clustered KNN when we include the biking activity which resulted in 75% accuracy rates. Additionally, it should be noted that although the Decision Tree performs better in terms of accuracy in this particular case, clustered KNN outperformed the Decision Tree in terms of other metrics like *precision, recall* and *f-measure* as indicated in the overall results in Table 4.11. Lastly, Naïve Bayes exhibited the worst performance in all cases.

Adding the biking activity slightly changed the behaviour of classifiers. First of all, accuracy rates of all classifiers decreased substantially because of the nature of biking activity. Most of the times, classifiers performed misclassification by mainly confusing running, biking and sitting activities. These results are mainly because of the uncontrolled experiments performed by the subjects. In the second group of tests, we did not provide any pre-training to the subjects to be able to measure the test results in a real environment. In most of the cases, we observed that the subjects start the biking activity with initial pedalling whereas they start to sit without pedalling after an initial velocity is achieved. This causes misclassification of biking and sitting cases. On the other hand, subjects start to get on and pedal faster whenever they pass an uphill on the way. This leads classifiers to confuse running and biking cases.

When we look at the overall results without biking, we observe that the window size we used is a more dominant system parameter than the sampling rates. In general, smaller window sizes (0.5 sec) achieved better results regardless of the considerable effect of the sampling rates. We observed that the running and walking activities are better distinguished with this window size which are misclassified most of the times during tests. In Figure 4.3, we plot the acceleration data for running activity with three consecutive timing windows at 0.5 second as an example. It can be seen that same pattern is preserved during consecutive windows.

Additionally, we observe that the sampling rate became a more dominant system parameter after adding the biking activity. In this case, smaller sampling rates (10 Hz) revealed better results. Basically, increasing sampling rates also detailed the nature of the activities but for the selected features and activity set, it created a negative effect

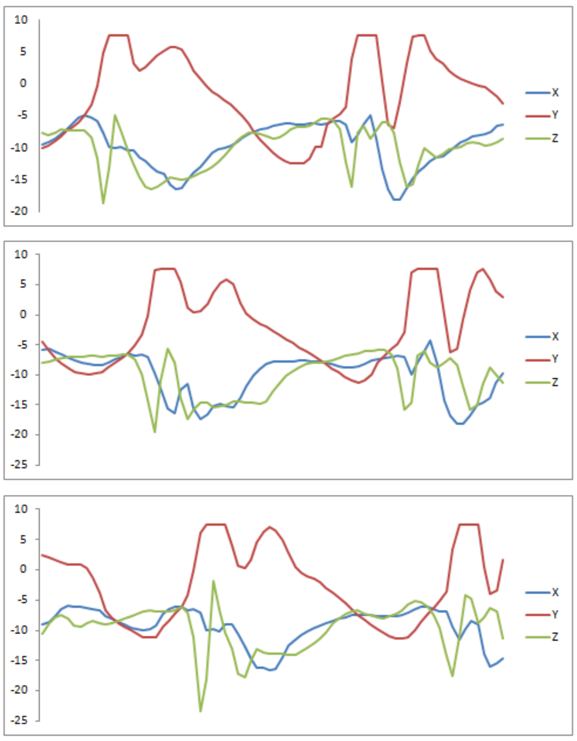


Figure 4.3. Consecutive timing windows of 0.5 sec for running activity.

on the results so that we could distinguish biking, running and walking activities better at smaller sampling rates from each other. It is also an important advantage for the energy efficiency of the proposed system.

Moreover, we can say that classifiers achieved better results with bigger window sizes after we added the biking activity. This is mainly because of the periodicity of the biking activity. From the previous studies, we know that some activities are recognized better with window sizes of 1 and 2 seconds [30] based on features selected for the classification phase. According to our results, we achieved the best results when the window size is selected as 2 seconds.

When we compare our findings with the related works detailed in Section 2.3, we can conclude that our results are comparable and similar. As an example, in [9], the proposed activity recognition system achieved 93%. They analysed similar activities like stationary, biking, walking, running but in addition to the accelerometer sensor, they used the advantage of the GPS sensor which provides the velocity information of the subjects so that they could achieve slightly better accuracies. Since we also consider the energy efficiency of the system and target also indoor activities, we do not use the GPS sensor. Additionally in [10], unlike our study, training models are created with an offline processing phase which increased the accuracy of the results. However, our results are still comparable with their results. When we benefit from the offline processing power, we observe that we also achieve similar results from the same data set we collected as detailed in Section 4.6.

4.4. Effect of Sliding Window Segmentation on the Performance

In this section, we evaluate the effect of the sliding window on the performance of the decision tree, based on previous results presented in Section 4.3. For this purpose, we implemented a simple sliding window algorithm with different overlapping ratios such as 25%, 50% and 75%. In this implementation, we overlap the old data that belong to the previous window with the new data of the current window considering their arrival times. We simply applied the first-in first-out approach. After generating the new sample set, we applied the same activity recognition steps as being done in the previous tests which are detailed in Section 2.2.

We evaluated the impact of parameter changes on the performance based on two experiments. We firstly tested the effect of sliding window with four activities which are *walking, running, standing* and *sitting*. Results in terms of accuracy with changing system parameters like the window size and the sampling rate are presented in Table 4.15. The first three rows in the table shows the accuracy rates whenever we apply the sliding window with different overlapping ratio. The last row indicates the original results without the sliding window which are already presented in Section 4.3. In general, we observe 2.66% increase in accuracy rates after we applied overlapping windows. The main conclusion is that overlapping the sliding window affects the performance of the system more positively as the window size increases. As we stated in Section 4.3, better accuracies are achieved in smaller window sizes in which selected activity patterns better fit to such windows. On the other hand, windows with two seconds are not able to catch activity patterns well so that it is not able to distinguish the selected activities. Overlapping windows eliminate this disadvantage which causes bigger increases in the system performance at larger window sizes. A major improvement (over 5%) is observed with the largest window size (2 sec) at higher sampling rates where the accuracy results improved and approached to the previous results with the smaller sampling rate (10 Hz).

Table 4.15. Sliding Window Effect on DT without Biking.

Window size(sec)	**0.5**			**1**			**2**		
Sampling Interval(msec)	10	50	100	10	50	100	10	50	100
75% OVERLAPPING	78.59	**86.42**	84.14	72.01	80.61	78.39	81.92	82.94	83.34
50% OVERLAPPING	79.96	**86.09**	84.26	72.40	80.48	78.55	81.43	83.63	84.05
25% OVERLAPPING	80.32	**86.47**	83.54	73.52	80.02	78.25	81.64	84.06	84.57
NON-OVERLAPPING	80.16	85.50	81.79	69.99	79.45	74.10	77.19	78.61	**85.52**

In the second group of tests, we added the *biking* activity as well. Results are presented in Table 4.16. The first three row again present the accuracy rates of the proposed system when we applied the sliding window whereas the last row indicates the original accuracy rates without the sliding window effect. According to the results, we have considerable improvements up to 3.1% in the cases where we had the worst results previously. On the other hand, we observe 0.77% improvement considering all results which is negligible.

4.5. Resource Usage of the Application

Besides the classification performance, we also evaluated the performance of clustered KNN in terms of execution times. As expected, classification execution times are considerably reduced as the K parameter is decreased. Moreover, classification times are highly dependent on the device model and capabilities as well.

Tests performed on Samsung Galaxy SII showed that the classification times are

Table 4.16. Sliding Window Effect on DT with Biking.

Window size(sec)	**0.5**			**1**			**2**		
Sampling Interval(msec)	10	50	100	10	50	100	10	50	100
75% OVERLAPPING	64.18	70.90	67.10	55.48	69.51	73.23	67.14	**73.70**	73.60
50% OVERLAPPING	65.20	71.18	65.80	55.78	68.71	70.79	66.03	**73.53**	71.83
25% OVERLAPPING	66.06	71.28	66.20	57.29	69.09	71.62	66.99	72.89	**73.81**
NON-OVERLAPPING	63.11	70.63	66.99	54.86	68.34	71.93	66.32	71.09	**75.33**

increased up to 300 msec whereas it decreased to 50 msec when K is selected as 10. File management and logging times are included to these results as well. On the other hand, the execution times on the less capable T-Mobile G2 Touch varied between 200 and 900 msec for different parameters. The CPU capability is important: Samsung Galaxy SII has a 1.2 GHz dual-core system on a chip (SoC) processor whereas T-Mobile G2 Touch has a 528 MHz ARM11 processor.

During our tests with different classifiers, we observed that the classification times are close to each other and mainly depend on the capabilities of the phone processors. On the other hand, the file read and write process is contributing more on recognition times which differ a lot from device to device. Because of this reason, we excluded Samsung Galaxy GIO and T-Mobile G2 phone models by taking into consideration above results which revealed very poor performance during file read and write process. We completed our tests with Samsung Galaxy SII, Samsung Galaxy W and Turkcell MaxiPlus5 smart phones.

Additionally, we also evaluated the resource consumption of the application. The first part of Table 4.17 summarizes the CPU and memory usage of the *Activity Recognizer* application with different classification methods. All measurements are taken from Samsung Galaxy SII. Resource usage of activity classifier highly depends on the classification methods being used at runtime.

According to the results, CPU and memory usage never exceeded 47% and 22 MB respectively. Application using clustered KNN consumes less resources among others. On the other hand, Naïve Bayes and Decision Tree have considerably higher CPU usage. Text to Speech (TTS) Service, which is being used for voice commands to guide the user about which activity to be performed during runtime, is included in the table for comparison purposes as well.

Table 4.17. CPU & Memory Usage of Activity Classifier and other applications.

Activity Recognizer		
	CPU Usage	**Memory Usage**
Clustered KNN	29%	21.9 MB
Decision Tree	47%	19.9 MB
Naïve Bayes	42%	12.6 MB
TTS Service	13%	12.4 MB
Benchmark Applications		
System	10%	28.8 MB
Norton Mobile	4%	19.4 MB
Internet	2%	37.0 MB
Google Maps	1%	31.2 MB

For benchmarking, resource usage values for common applications are presented in the second part of Table 4.17. According to these results, the performance of the *Activity Recognizer* application is comparable with frequently used applications in terms of resource efficiency.

4.6. Online vs Offline Classification

In this section, we compare the performance of the classifiers in terms of online and offline processing. Our aim is to investigate how close the results we achieved with online tests to the results of activity recognition whenever we process the same data sets offline with the same classifiers and selected features while using the computational power and capabilities of desktop computers. For this purpose, we used the raw data (both training and test data) which is logged during our online tests with the biking activity. We performed offline tests with the help of Weka Machine Learning Toolkit (WMLT) [22].

We performed exactly the same tests with the same system parameters offline for comparison purposes for each subject. Unlike online tests, this time we applied unlimited complete training sets of each subject to train the system. Additionally, we benefit from the offline processing power of a standard computer by using 10 fold cross validation during the classification steps.

According to the results given in Table 4.18, we observe considerable increase in accuracies of DT which ranges between 23% and 40%. On average, we have 35% improvement for all results whenever we apply DT offline with 10-fold cross validation with the help of WMLT.

Table 4.18. Online vs Offline Classification - Decision Tree.

	Online	**Offline**
Accuracy	75.83%	93.04%
Precision	67.32%	93.11%
Recall	73.14%	92.85%
F-Measure	69.95%	92.97%

Additionally, we performed the same tests for Naïve Bayes as well since we had very poor results whenever we used Naïve Bayes in the online classification step. Similar to Decision Tree, we performed 10-fold cross validation with complete training sets of each subject. After offline classification, we observed a noticeable increase at all metrics as shown in Table 4.19. These results showed that Naïve Bayes is not a preferable online classifier whereas it performs far better results whenever it is executed offline with larger training sets.

Table 4.19. Online vs Offline Classification - Naïve Bayes.

	Online	**Offline**
Accuracy	46.34%	77.11%
Precision	41.69%	77.97%
Recall	46.42%	77.99%
F-Measure	43.68%	77.98%

We should remind that the offline results are the best results we may achieve. As smart phones in the market continue to evolve rapidly, those results are likely to be achieved in a near future.

4.7. User and Device Dependency for Training Sets

In this section, we evaluate the user and device dependency of the proposed system. For this purpose, we selected four subjects who performed the same tests previously with their own training data. The overall results are presented in Table 4.20. In this table, each row represents the subject who performed the tests. On the other hand, each column in the table represents whose training data is used during the testing period. The fifth column with "***" indicates the training set of a subject from which only his own training data are excluded.

All subjects have similar physical features in terms of their weight, length and age. On the other hand, subjects used different phone models during tests. Y and W performed the test with the same Android Galaxy W smart phone whereas subject X performed the tests with different smart phone but used the same phone model. On the other hand, subject Z performed the tests with Samsung Galaxy SII.

All of the tests were performed with the decision tree algorithm and with the default system parameters which are window size of 2 sec and sampling interval of 100 msec. As presented previously in Table 4.11, we had the best accuracy with these

Table 4.20. Accuracy(%) Results of Mixed Training Sets.

	X	**Y**	**Z**	**W**	***	**XYZW**	**AVG**
X	*98.4*	39.3	20.0	37.1	20.1	**39.9**	31.3
Y	37.1	*77.1*	50.7	**70.7**	48.6	56.4	52.7
Z	38.4	52.2	*74.7*	66.7	**68.8**	52.2	55.7
W	55.4	**76.3**	21.6	*68.3*	64.0	61.9	55.8

parameters with an average of 75%. We observed interesting results after training sets are mixed such that the average accuracy results decreased substantially to 48% in this case. We observed only in one specific case an improvement. Subject W results improved with the subject Y training set such that biking activity is better distinguished when subject W training sets are used. Previously, it was being misclassified as walking or running.

There are a few cases which should be emphasized in terms of the results. Except individual training sets, mixture of training data of multiple users are the cases which mainly contributes to the average results in which the accuracy ranges between 40% and 62%. In general, we observe worst results when subject X used training sets of other subjects since subject X has very distinct activity patterns compared to other subjects. According to previous tests with his own training data, he achieved one of the best performance among all subjects. On the other hand, we observe over 70% accuracy rate with the tests of subject W and Y. We should note that, these two subjects W and Y performed the tests with the same phone which is the key factor increasing accuracy of the results. Additionally, training set of subject Z lowered the results till 20% since his training data are collected with another phone model where hardware changes of the phone directly affected the results. However, similarity at the results of subject Z and Y is also remarkable although they have used different phone models.

In general, results reveal that the proposed system is highly user dependent which we assumed in the beginning of this thesis. We believe that mobile phones are personal belongings so that such applications would be used for private purposes. Because of this reason, we do not feel any inconvenience of user dependency of the proposed system. However, results are promising and giving an insight to us such that the training data coming from two distinct subjects with same physical features and using the same phone model can be used for each other which is an important hint to be able to create user independent training data sets, as also mentioned in [33]. Sharing data sets between users with similar behaviours could certainly decrease the burden of training on the new users such that they can start to use the application immediately without the requirement of data collection.

5. CONCLUSIONS AND FUTURE WORK

In the literature, there are only a few works on online activity recognition using the sensors on smart phones. In this thesis, we proposed an activity recognition system working on Android platforms that supports online training and classification while using only the accelerometer data for classification. For this purpose, we selected the most suitable methods for activity recognition steps which can be applied online. Our target was to recognize most common human activities such as *walking, sitting, standing, running* and *biking* based on our literature survey.

Accordingly, we first evaluated the online classification performance of the Naïve Bayes classifier. Later, we investigated the performance of other classifiers such as clustered KNN and Decision Tree in the same proposed system. We compared their performances considering the most effective system parameters like the window size and the sampling rate. According to the results, Clustered KNN method exhibited a much better performance than the Naïve Bayes classifier in terms of accuracy, battery usage and resources needed on mobile platforms. On the other hand, its performance is nearly the same as the Decision Tree classifier. These results reveal that compared to the previous studies and even with the ones which are considering offline training and classification, clustered KNN provides promising results. Additionally, we investigated the effects of other system factors like the sliding window and the mixed training data on the system performance. According to the results, sliding window with X% overlapping ratio does not have any significant effect on system performance. On the other hand, other tests performed with mixed training data showed that the proposed system is user-dependent which is mainly related with different accelerometer hardware used in different device models.

As a future work, we are interested in investigating the effects of using different features on our system and mainly concentrate on the training phase. We plan to create a system which can increase the training data size continuously and create better training sets by renewing them with new training data even if we put the data size limitation for online training purposes. By this way, the training set will be improved as long as a user collects new data. Additionally, we want to search for possibilities to use other complex classifiers online in our current system by using newly released smart phones with improved processing and computation power.

APPENDIX A: APPLICATION LOG FILES

Files created after Naïve Bayes Classification. *Rawdata.txt* contains all raw data being sampled by accelerometer sensor. Its format is the same as training files shown in Figure 3.1. Additionally, there is a time stamp at the beginning of each data so that we can measure exact sampling times of each data as well. $Classification.txt$ contains all data classified by the application (format: $x\ y\ z\ classified\_activity\_label$). $Truth.txt$ contains all data with the ground truth activity label. *Results.txt* contains all results like accuracy, true inferred, true positive, total of ground truth, precision and recall.

Files created after clustered KNN Classification. *traininglogger.txt* contains all data after preprocessing step which is fired at application start up. *Rawdata.txt* contains all raw data being sampled by accelerometer sensor. *Classification.txt* file contains average, minimum, maximum, standard deviation values of each sample set. It also contains classification result of each data set. *Results.txt* shows voting result for each activity. Each column in this text file represents *RUNNING, STANDING, BIKING, SITTING,* and $WALKING$ respectively. The last column shows total execution times for each classification.

Files created after DT Classification. *activity.db* file contains filtered information after processing step is completed. Extracted feature values are written to this file with the labelling of each activity. *Rawdata.txt* contains all raw data being sampled by accelerometer sensor. $Classification.txt$ file contains average, minimum, maximum, standard deviation values of each window and the label of classified activity. Last column of this file represents classification execution times which is also useful information for performance evaluation. *decisiontree.txt* indicates generated decision tree after training phase which is being used during classification process to take decisions.

REFERENCES

1. Kose, M., O. D. Incel and C. Ersoy, “Online Human Activity Recognition on Smart Phones”, *Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data (colocated with IPSN 2012)*, pp. 11–15, Beijing, China, 2012.

2. Kose, M., O. D. Incel and C. Ersoy, “Performance Evaluation of Classification Methods for Online Activity Recognition on Smart Phones”, *20th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, Mugla, Turkey, 2012.

3. Krishnan, N. C. and D. J. Cook, “Activity Recognition on Streaming Sensor Data”, *Pervasive and Mobile Computing*, 2012, `http://www.sciencedirect.com/science/article/pii/S1574119212000776`.

4. Avci, A., S. Bosch, M. Marin-Perianu, R. Marin-Perianu and P. Havinga, “Activity Recognition Using Inertial Sensing for Healthcare, Wellbeing and Sports Applications: A Survey”, *23th International Conference on Architecture of Computing Systems, ARCS 2010*, pp. 167–176, 2010.

5. Pentland, A., “Looking at People: Sensing for Ubiquitous and Wearable Computing”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, pp. 107–119, 2000.

6. Krishnan, N. C., C. Juillard, D. Colbry and S. Panchanathan, “Recognition of Hand Movements Using Wearable Accelerometers”, *Journal of Ambient Intelligence Smart Environment*, Vol. 1, No. 2, pp. 143–155, 2009.

7. Zeng, H. and Y. Zhao, “Sensing Movement: Microsensors for Body Motion Measurement”, *Sensors*, Vol. 11, No. 1, pp. 638–660, 2011.

8. Wang, Y., J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari

and N. Sadeh, “A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition”, *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys ’09, pp. 179–192, ACM, New York, NY, USA, 2009.

9. Reddy, S., M. Mun, J. Burke, D. Estrin, M. Hansen and M. Srivastava, “Using Mobile Phones to Determine Transportation Modes”, *ACM Transactions on Sensor Networks*, Vol. 6, No. 2, pp. 13:1–13:27, 2010.

10. Saponas, T. S., J. Lester, J. Froehlich, J. Fogarty and J. Landay, *iLearn on the iPhone: Real-Time Human Activtity Classification on Commodity Mobile Phones*, Tech. Rep. UW-CSE-08-04–02, 2008.

11. Yang, J., “Toward Physical Activity Diary: Motion Recognition Using Simple Acceleration Features with Mobile Phones”, *Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics IMCE ’09*, pp. 1–10, 2009.

12. Miluzzo, E., 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”, *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys ’08, pp. 337–350, ACM, New York, NY, USA, 2008.

13. Mladenov, M. and M. Mock, “A Step Counter Service for Java-Enabled Devices Using a Built-in Accelerometer”, *Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009)*, CAMS ’09, pp. 1–5, ACM, New York, NY, USA, 2009.

14. Fabian, A., N. Gyorbiro and G. Homanyi, “Activity Recognition System for Mobile Phones Using the MotionBand Device”, *Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*,

MOBILWARE ’08, pp. 41:1–41:5, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, 2007.

15. Yavuz, G., M. Kocak, G. Ergun, H. O. Alemdar, H. Yalcin, O. D. Incel and C. Ersoy, “A Smartphone Based Fall Detector with Online Location Support”, *Proceedings of PhoneSense 2010*, Zurich, Switzerland, 2010.

16. Lane, N., M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury and A. Campbell, “BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing”, *Pervasive Health 2011*, 2011.

17. Blanke, U., D. Larlus, K. Van Laerhoven and B. Schiele, “Standing on the Shoulders of Other Researchers - A Position Statement”, *Proc. of the Workshop ’How to do good activity recognition research? Experimental methodologies, evaluation metrics, and reproducibility issues’ (Pervasive 2010)*, Helsinki, Finland, 2010.

18. Reddy, S., D. Estrin and M. Srivastava, “Recruitment Framework for Participatory Sensing Data Collections”, *Proceedings of the 8th International Conference on Pervasive Computing*, pp. 138–155, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

19. Toplan, E., Y. E. Ustev, O. D. Incel and C. Ersoy, “Citisense: Capturing the City Dynamics with Activity and Transport Mode Recognition”, *Third International Workshop on Sensing Applications on Mobile Phones (PhoneSense)*, 2012.

20. Kiukkonen, N., B. J., O. Dousse, D. Gatica-Perez and L. J., “Towards Rich Mobile Phone Data Sets: Lausanne Data Collection Campaign”, *Proc. ACM Int. Conf. on Pervasive Services (ICPS, Berlin)*, 2010.

21. Fitz-Walter, Z. and D. Tjondronegoro, “Simple Classification of Walking Activities Using Commodity Smart Phones”, *OZCHI ’09 Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*, pp. 409–412, 2009.

22. Machine Learning Group at University of Waikato, *Weka Machine Learning Toolkit*, 2012, `http://www.cs.waikato.ac.nz/ml/index.html`, accessed at July 2012.

23. Karantonis, D. M., 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", *IEEE Transactions on Information Technology in Biomedicine*, pp. 156–167, 2006.

24. Bigargaddi, N., A. Sarela, L. Klingbeil and M. Karunanithi, "Detecting Walking Activity in Cardiac Rehabilitation by Using Accelerometer", *International Conference on Intelligent Sensors, Sensor Networks and Information Processing' 07*, pp. 555–560, 2007.

25. Krishnan, N. C., D. Colbry, C. Juillard and S. Panchanathan, "Real Time Human Activity Recognition Using Tri-Axial Accelerometers", *In Sensors Signals and Information Processing Workshop*.

26. Maurya, M. R., R. Rengaswamy and V. Venkatasubramanian, "Fault Diagnosis Using Dynamic Trend Analysis: A Review and Recent Developments", *Engineering Applications of Artificial Intelligence archive*, Vol. 20, pp. 133–146, Mar 2007.

27. Keogh, E. J., S. Chu, D. Hart and M. J. Pazzani, "An Online Algorithm for Segmenting Time Series", *Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM 2001*, pp. 289–296, IEEE Computer Society, Washington, DC, USA, 2001.

28. Wang, S., J. Yang, N. Chen, X. Chen and Q. Zhang, "Human Activity Recognition with User-Free Accelerometers in the Sensor Networks", *International Conference on Neural Networks and Brain, ICNN&B '05*, Vol. 2, pp. 1212–1217, Beijing, China, 2005.

29. Pirttikangas, S., K. Fujinami and T. Nakajima, "Feature Selection and Activity

Recognition from Wearable Sensors", *Ubiquitous Computing Systems, Third International Symposium, UCS 2006*, pp. 516–527, 2006.

30. Huynh, T. and B. Schiele, "Analyzing Features for Activity Recognition", *2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies (sOcEuSAI2005)*, pp. 159–163, ACM Press New York, NY, USA, ACM Press New York, NY, USA, Grenoble, France, 2005.

31. Könönen, J., J. Mantyjarvi, H. Simila, J. Parkka and M. Ermes, "Automatic Feature Selection for Context Recognition in Mobile Devices", *Pervasive and Mobile Computing*, Vol. 6, pp. 181–197, 2010.

32. Khan, A., Y. Lee, S. Lee and T. Kim, "Human Activity Recognition via An Accelerometer-Enabled-Smartphone Using Kernel Discriminant Analysis", *Future Information Technology (FutureTech), 2010 5th International Conference on*, pp. 1–6, 2010.

33. Miluzzo, E., C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu and A. T. Campbell, "Darwin Phones: The Evolution of Sensing and Inference on Mobile Phones", *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pp. 5–20, ACM, New York, NY, USA, 2010.

34. Peebles, D., H. Lu, N. D. Lane, T. Choudhury and A. T. Campbell, "Community-Guided Learning: Exploiting Mobile Sensor Users to Model Human Behavior", *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pp. –1–1, 2010.

35. Bieber, G., P. Koldrack, C. Sablowski, C. Peter and B. Urban, "Mobile Physical Activity Recognition of Stand Up and Sit Down Transitions for User Behaviour Analysis", *Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments, PETRA*, 2010.

36. Lukowicz, P., A. Pentland and A. Ferscha, "From Context Awareness to Socially

Interactive Computing", *IEEE International Conference on Pervasive Computing and Communications*, Vol. 11, Lugano, Switzerland, 2012.

37. Tapia, E. M., *Using Machine Learning for Real-Time Activity Recognition and Estimation of Energy Expenditure*, Ph.D. Thesis, Massachusetts Institute of Technology, 2008.

38. Berchtold, M., M. Budde, D. Gordon, H. Schmidtke and M. Beigl, "ActiServ: Activity Recognition Service for Mobile Phones", *Wearable Computers (ISWC), 2010 International Symposium on*, pp. 1–8, 2010.

39. Consolvo, S., D. W. McDonald, T. Toscos, M. Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, I. Smith and J. A. Landay, "Activity Sensing in the Wild: A Field Trial of Ubifit Garden", *CHI '08 Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pp. 1797–1806, 2008.

40. Iso, T. and K. Yamazaki, "Gait Analyzer Based on a Cell Phone with a Single Three-Axis Accelerometer", *MobileHCI '06 Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pp. 141–144, 2006.

41. Farrahi, K. and D. Gatica-Perez, "Daily Routine Classification from Mobile Phone Data", *MLMI '08 Proceedings of the 5th international workshop on Machine Learning for Multimodal Interaction*, pp. 173–184, 2008.

42. Denning, T., A. Andrew, R. Chaudhri, C. Hartung, J. Lester, G. Borriello and G. Duncan, "BALANCE: Towards a Usable Pervasive Wellness Application with Accurate Activity Inference", *HotMobile '09 Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, 2009.

43. Choujaa, D. and N. Dulay, "TRAcME: Temporal Activity Recognition Using Mobile Phone Data", *EUC '08 Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, Vol. 1, pp. 119–126, 2008.

44. Anderson, I., J. Maitland, S. Sherwood, L. Barkhuus, M. Chalmers, M. Hall, B. Brown and H. Muller, "Shakra: Tracking and Sharing Daily Activity Levels with Unaugmented Mobile Phones", *Mobile Networks and Applications*, Vol. 12, pp. 185–199, 2007.

45. Sohn, T., A. Varshavsky, A. LaMarca, M. Y. Chen, T. Choudhury, I. Smith, S. Consolvo, J. Hightower, W. G. Griswold and E. de Lara, "Mobility Detection Using Everyday GSM Traces", *UbiComp'06 Proceedings of the 8th international conference on Ubiquitous Computing*, pp. 212–224, 2006.

46. Papliatseyeu, A. and O. Mayora, "Mobile Habits: Inferring and Predicting User Activities with a Location-Aware Smartphone", *3rd Symposium of Ubiquitous Computing and Ambient Intelligence*, Vol. 51, pp. 343–352, 2008.

47. Wittke, M., U. Jänen, A. Duraslan, E. Cakar, M. Steinberg and J. Brehm, "Activity Recognition Using Optical Sensors on Mobile Phones", *GI Jahrestagung*, Vol. 154 of *LNI*, pp. 2181–2194, GI, 2009.

48. Google, *Android Developer Tutorial*, 2012, `http://developer.android.com`, accessed at June 2012.

49. Erman Dogan, *Efficient Feature Extraction For Activity Recognition on Mobile Phones*, M.S. Thesis, Bogazici University, 2012.

50. Alpaydin, E., *Introduction to Machine Learning*, MIT Press, Cambridge, USA, 2nd edn., 2010.

51. Japan Association of Remote Sensing, *Remote Sensing Notes*, JARS, 1999.

52. Quinlan, J. R., "Induction of Decision Trees", *Machine Learning*, Vol. 1, No. 1, pp. 81–106, 1986.

53. Jean-Marc François, *jaDTi Decision Trees: a Java implementation*, 2004,

`http://www.run.montefiore.ulg.ac.be/~francois/software/jaDTi/`, accessed at November 2004.

54. RuleQuest Research, *RuleQuest Data Mining Tools*, 2010, `http://www.rulequest.com/`, accessed at July 2010.

55. Greenwood, P. E. and M. S. Nikulin, *A Guide to Chi-Squared Testing*, Wiley-Interscience, New York, NY, USA, 1996.

56. Izenman, A. J., *Modern Multivariate Statistical Techniques : Regression, Classification, and Manifold Learning*, New York Springer, New York, NY, USA, 2008.

57. Jain, R. K., *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley, New York, NY, USA, 1 edn., 1991.

58. van Kasteren, T. L. M., *Activity Recognition for Health Monitoring Elderly Using Temporal Probabilistic Models*, Ph.D. Thesis, University of Amsterdam, 2011.