Human activity recognition using smartphone's sensors and machine learning

Human Activity Recognition using Smartphone's Sensors and Machine Learning

by

Enrique Alejandro García Ceja

THESIS

Submitted to the Escuela de Ingeniería y Tecnologías de Información

in partial fulfillment of the requirements for the degree of

Master of Science in Intelligent Systems

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Monterrey, N.L. December 2012

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Escuela de Ingeniería y Tecnologías de Información

The committee members, hereby, certify that have read the Master Thesis presented by Enrique Alejandro García Ceja and that it is fully adequate in scope and quality as a partial fulfillment of the requirements for the degree of *Master of Science in:*

Intelligent Systems

Thesis Committee:

PhD. Ramón F. Brena Pinero
Thesis Advisor

PhD. Leonardo Garrido
Committee Member

PhD. Marcela Rodríguez Urrea
Committee Member

PhD. Ramón F. Brena Pinero
Director of Research and Graduate Programs. School of Engineering.

December 2012

A mi Familia

Acknowledgments

I would like to thank my advisor Dr. Ramón Brena and the thesis committee: Dr. Leonardo Garrido and Dra. Marcela Rodríguez. To the secretary Isabel for helping me with administrative tasks. And to the National Council of Science and Technology (CONACYT) for the financial support.

Enrique Alejandro García Ceja

Instituto Tecnológico y de Estudios Superiores de Monterrey
December 2012

Human Activity Recognition using Smartphone's Sensors and Machine Learning

Enrique Alejandro García Ceja, M.Sc.
Instituto Tecnológico y de Estudios Superiores de Monterrey, 2012

Thesis Advisor: PhD. Ramón F. Brena Pinero

In this thesis human activity recognition is performed from data gathered through the sensors of smpartphones. Human activity recognition is an important task for ambient intelligence systems. Being able to recognize the state of a person can provide us with valuable information that can be used as input for other systems. For example, in health care, fall detection can be used to alert the medical staff in case of an accident; in security, abnormal behavior can be detected and thus used to prevent a burglary or other criminal activities. This work focuses on physical and daily living activities. The first type refers to activities that can be inferred by just analyzing them for a few seconds (2-10 seconds) and are independent of the situation, e.g., *walking, resting, running, etc.* The second type refers to activities that are composed of a collection of physical activities, e.g., *working, shopping, exercising, etc.* We also used information gathered from Wi-Fi access points to contextualize the physical activities in order to have a better understanding of the user's situation. In this case, context is defined as the additional relevant information that gives sense to the user's actions. The activity recognition task was stated as a classification problem and Machine Learning methods (supervised and unsupervised) were used to perform the classification. For physical activities recognition, K-nearest neighbors, Decision tree and Naïve Bayes methods were used, achieving overall accuracies of 89.33%, 87.33% and 93.33%, respectively. Simple methods and computationally efficient features were looked for given that the implementation is intended for small devices with scarce resources. Once the physical activity is recognized, the next step is to put it in context. Since every Wi-Fi access point has a unique identifier, it is possible to use this information along with the received signal strength to locate the user. For example, if the user is walking and the Wi-Fi access points' identifiers correspond to the ones installed in a library, then it could be inferred that the user is looking for a book. Since the access points' identifiers are not known until runtime, a *lazy* classifier such as K-nearest neighbors was used, achieving

an overall accuracy of 89.7%. For the daily living activities recognition task, a bag of features approach was used. This method consists of modeling the entire activity as a distribution of physical activities called primitives. The average overall accuracy for five different daily living activities performed in an academic environment was 87.61%.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Human activity recognition is an important task for Ambient Intelligence [1] systems. Being able to recognize the state of a person can provide us with valuable information that can be used as input for other systems. For example, in health care, fall detection can be used to alert the medical staff in case of an accident; in security, abnormal behavior can be detected and thus used to prevent a burglary or other criminal activities.

Ambient Intelligence (AmI) is an emerging discipline that brings intelligence to our everyday environments and makes those environments sensitive to us [1]. It builds upon advances in sensors, pervasive computing, and artificial intelligence. AmI technologies should be sensitive, responsive, adaptive, transparent, ubiquitous, and intelligent.

In an AmI environment, devices are expected to work collectively by sharing information and using the history of past events. Lighting, sound, vision, domestic appliances, and personal health care products all cooperate seamlessly to improve the total user experience with the aid of natural and intuitive user interfaces [2].

This thesis is part of the Autonomous Agents in Ambient Intelligence research group from the Computer Science Department at Instituto Tecnológico y de Estudios Superiores de Monterrey, which objective is to devise ways of intelligently use the information about the location and situation of persons and devices, registered by models or by the use of sensors, in order to deliver personalized services and improve the adequacy of technology products.

Below is the definition of some concepts that are going to be used throughout this document:

- **Context.** Anind K. [3] defines it as:

  > Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

- **Physical activities.** These are activities that can be inferred by just analyzing

them for a few seconds (2-10 secs.) [4, 5, 6] and are independent of the situation in which they are being performed. For example a user may be *running* because he/she is exercising or he/she is in a hurry. In this case the activity can be identified regardless of the two situations. Examples of this types of activities are: *walking, resting, running,* etc. In the rest of this document the terms *physical activity* and *simple activity* will be used interchangeably.

- **Activities of daily living (ADLs).** MedicineNet [1] defines them as:

  > The things we normally do in daily living including any daily activity we perform for self-care such as feeding ourselves, bathing, dressing, grooming, work, homemaking, and leisure.

- **Instrumental activities of daily living (IADLs).** This term was introduced by Lawton and Brody [7] for activities that are not necessarily for fundamental functioning but are useful to live independently. Examples of IADLs are: *housework, shopping groceries and clothes, transportation, food preparation,* etc. In the rest of this document the term *complex activities* will refer to ADLs and IADLs and these can be inferred from a set of physical activities.

The present work focuses on determining the *user's situation*. This includes: 1) Recognizing the physical activity a person is performing and then contextualizing it. 2) The recognition of Instrumental activities of daily living which will be modeled as a distribution of physical activities. Machine Learning methods will be used since they offer flexibility to the system to automatically be improved over time and include additional activities without extra effort or configuration.

1.1 Problem Statement

In order to determine the state of a person, first we need a set of sensors to monitor several parameters. In this case, we are interested in the physical activities being performed by people. There have been several approaches that tackle the problem of human activity recognition. Some of them are based on video [8, 9, 10]. Others make use of the commercial Vicon system[2] which consists of spherical retro-reflective markers and infrared light sources. Recently, the Kinect sensor by Microsoft which employs a variant of image-based 3D reconstruction has gained a lot of attention in the research field and has been used for activity recognition [11]. Other approaches use wearable sensors like accelerometers, digital compasses, angular velocity sensors, RFID tags, etc. [12, 13].

[1] MedicineNet.com http://www.medterms.com/script/main/art.asp?articlekey=2152
[2] http://www.vicon.com/

The problem with those approaches is that some of them are expensive and need a fixed infrastructure. Another issue is that some of them require the use of several sensors attached to different parts of the body (e.g., in the wrists, hip, thigh, ankles, arms, etc.). As stated before some of the desired features of AmI technologies is that they should be ubiquitous and transparent but also unobtrusive, unnoticeable and user friendly. In order to comply with those requirements we need a way to perform the recognition in such a form that adheres to the desired features.

In recent years simple human activity recognition has been achieved successfully, but most of the time using sensor units that are not user-friendly. Complex activity recognition is still challenging and is an active area of research.

1.2 Objectives

The main objective of this work is to perform human physical and daily living activity recognition in an unobtrusive way and without the need of a fixed infrastructure. Emphasis is made in the use of simple Machine Learning methods since the implementation is intended for small devices with scarce resources. From this general objective the following particular objectives are derived:

- Perform physical activity recognition in real time using a smartphone's triaxial accelerometer.
- Research which features are computationally simple but, yet effective in order to characterize the activities.
- Evaluate different types of base-level classifiers[3] in order to determine which one is best suited to comply with the general objective. In this case we tested with KNN, Naive Bayes and C4.5 because they are simple enough to be implemented in a small device and we wanted to try with different types of classifiers (a geometric, probabilistic, and tree based).
- Devise a way to contextualize the physical activities in order to have a better understanding of the user's state.
- Perform daily living activities recognition by means of a Bag-Of-Features approach.
- Validate the results in order to tell whether or not the recognition is suitable for real life usage.

[3] A base-level classifier consists of a single classifier, e.g., C4.5, Naive Bayes, KNN, etc. In contrast, a meta-level classifier consists of a set of single classifiers and uses the information provided by each one to produce the final output. Examples of these schemes are bagging and stacking.

1.3 Hypotheses

The hypotheses to be tested with the present work are:

- Physical activity recognition can be performed in real time using a smartphone's triaxial accelerometer.
- Machine Learning methods can be used to recognize human activities within acceptable accuracies to be used in real life scenarios.
- Computationally efficient features can be used to characterize the activities.
- Physical activities can be contextualized using information gathered from Wi-Fi Access Points.
- Daily living activity recognition can be performed through the composition of simple activities.
- Supervised and unsupervised learning methods can be used to recognize daily living activities.

1.4 Research Questions

To achieve the proposed objectives and to verify the hypotheses, the following research questions are addressed:

- Can we use simple smartphone's sensors to perform real time activity recognition?
- Can simple supervised learning methods be used to classify physical activities?
- Which features are adequate for the classification task?
- Can we use information from Wi-Fi Access Points to contextualize activities?
- How can we represent complex activities in order to perform classification?
- Can supervised and unsupervised learning methods be used together to automatically find physical activities within daily living activities?

1.5 Justification

In the last years smartphone ownership has been on the rise. So far to July 2012 smartphone penetration in United States and México is 44% and 20% respectively[4] (Figure 1.1). Figure 1.2 shows the penetration by age in México. According to this graph, any person of age 18 or older would be a possible user of these type of applications.

Figure 1.3 shows the location of smartphone usage for México and United States. From the graph, it can be seen that the most common places in which smarthphones are used are at home, work, and on the go. Smartphones have become a vital device for many people and this tendency is increasing. Therefore, it seems a natural way to perform activity recognition with smartphones since many users are already accustomed to this type of devices. As a consequence, users are freed from having to attach other sensors to their bodies.

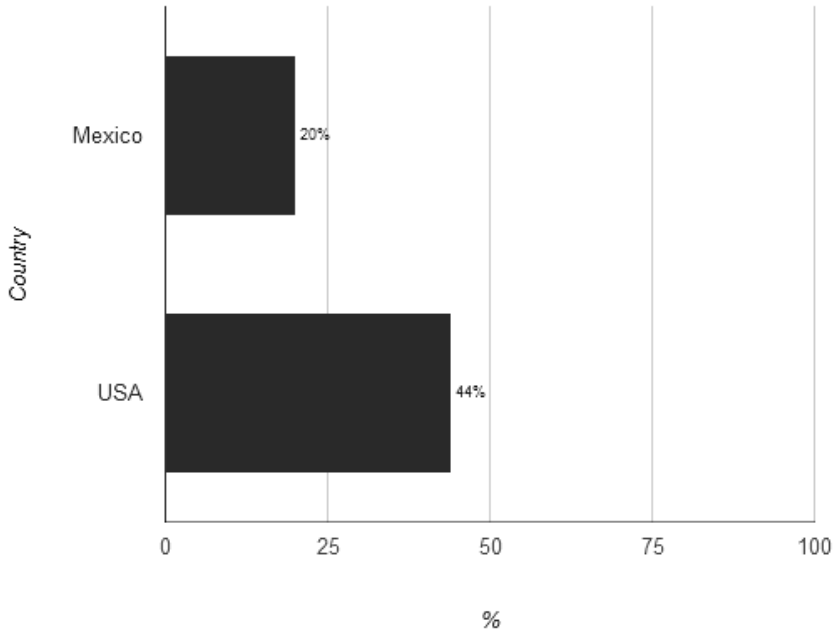

Figure 1.1: Smartphone penetration so far to July 2012

[4]http://www.thinkwithgoogle.com/mobileplanet/ (retrieved July 2012)

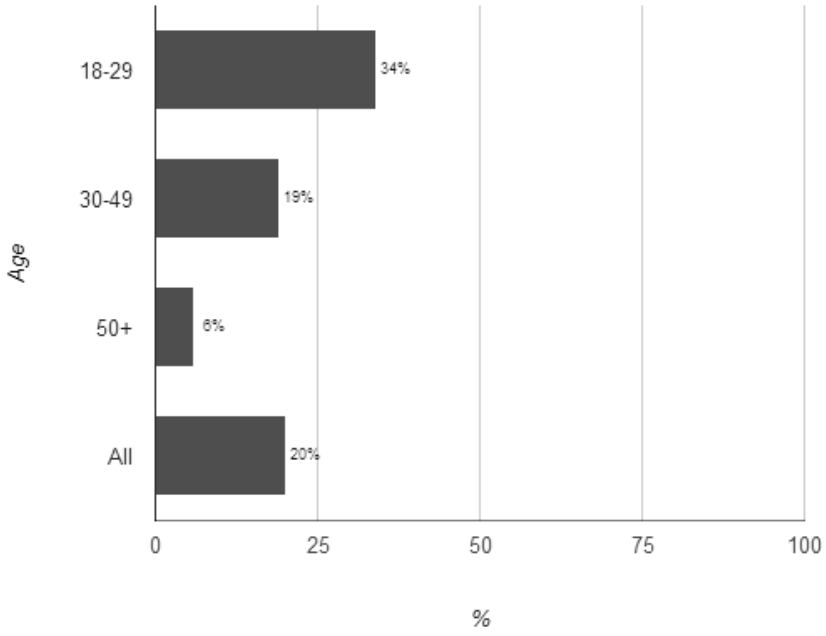

Figure 1.2: México Smartphone penetration by age so far to July 2012

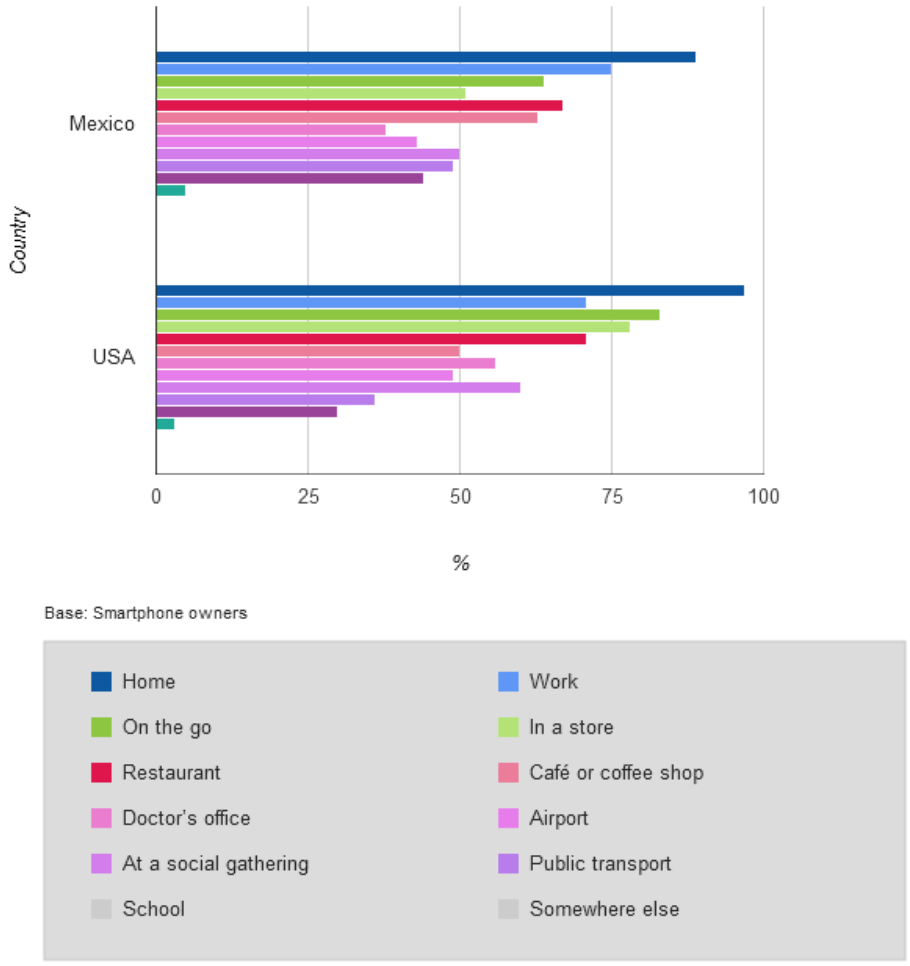

Figure 1.3: Smartphone usage so far to July 2012

1.6 Methodology

In order to fulfill the previous objectives the next methodology is going to be followed:

Step 1: Information gathering and research.

In this step similar works and the state of the art are going to be looked for. Special attention will be put on the type of sensors and classification methods being used in previous works. This information will guide the work towards an original solution−but more important−a practical one.

Step 2: Analysis of the collected information.

The information gathered in the previous step is going to be classified and analyzed. This will help to determine which works are of major significance for this work.

Step 3: Development of the method.

Once the information has been analyzed, we can continue with the following steps:

1. Problem identification
2. Brainstorming
3. Design and development of the method

Step 4: Report writing and corrections.

The thesis report will be written in parallel with Step 3. As Step 3 progresses, the report will be updated accordingly.

Step 5: Results analysis.

In this step the method's results are analyzed and evaluated, conclusions are drawn, and the report is updated. By the end of this step it is expected that the research questions are already answered.

Step 6: Thesis final report and defense.

The final thesis report must be completed and the thesis defense is taken.

Figure 1.4 shows a Gannt graph of the work plan. The work starts in June 2011 and ends in November 2012.

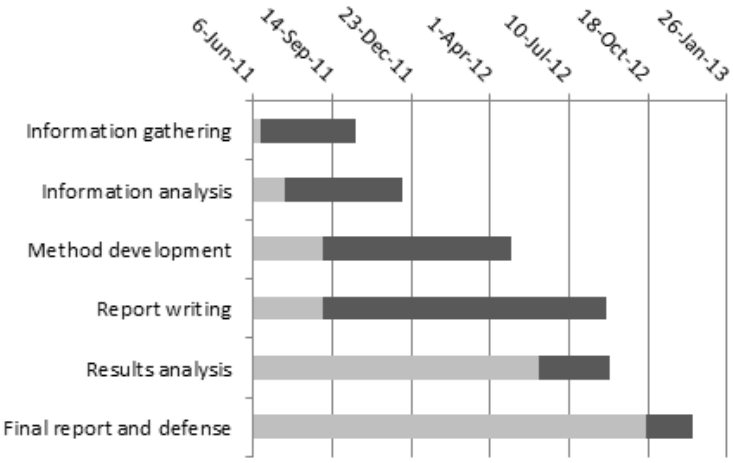

Figure 1.4: Schedule. The dark region shows the start and duration of each task.

1.7 Organization

This work is divided into five chapters. *Chapter 2* introduces the theoretical framework which serves as the basis of the present work. It begins with a general overview of Ambient Intelligence and then it introduces the concept of Machine learning. Then it explores some supervised and unsupervised learning methods and concludes with recent related work in the field of human activity recognition. *Chapter 3* presents the method which begins explaining the sensing platform and then proceeds with three sections: Physical activity recognition, Contextualizing physical activities with information gathered from Wi-Fi access points, and Complex activity recognition. Each section describes the data collection process, the training and classification phases. *Chapter 4* describes how the experiments were performed and presents the results. Finally in *Chapter 5* conclusions are drawn and the future work is defined.

Chapter 2

Theoretical Framework

Activity recognition can be stated as a classification problem, i.e, given a set of activities, we want to assign them a class. The output of the classification task can be used to feed other systems. *Ambient intelligence* systems can take advantage of the state of a person in order to provide more accurate services. This chapter begins with an overview of Ambient intelligence. Then, an overview of *Machine learning* is presented which includes basic concepts and the definition of classification and regression. Then two types of learning are explained: *Supervised learning* and *Unsupervised learning*. Finally, related work on activity recognition is presented which uses Machine learning methods extensively.

2.1 Ambient Intelligence

Ambient Intelligence (AmI) brings together many fields like networks, sensors, human-computer interfaces, pervasive computing, artificial intelligence, robotics, multi-agent systems, etc. to provide flexible and intelligent services to users acting in their environments. The environment needs to be sensitive which means that it must be capable of recognizing the users, learn their preferences, and have a degree of empathy or react to the user's mood. *Smart environment* is the physical infrastructure (sensors, actuators, networks, etc.) that supports the system [14].

Nowadays a user having access to several computers is a common thing. This is not just regarding personal computers or portable computers but other types of appliances with embedded microprocessors like refrigerators, washing machines, mobile phones, watches, tablets, GPS systems, etc. AmI takes advantage of all this computational power and makes all this appliances work together in order to provide user services. It also uses the current infrastructure like Wifi Access Points, security cameras, temperature sensors, etc.

Nowadays we are experiencing changes in the way services are handled. One example is the decentralization of health care where professionals are departing from the hospital centric health care system to support caring for patients closer to home

and within their communities [1]. Another example is the decentralization of education in which virtual courses are offered and students can learn from their homes. AmI has several applications including Smart Homes, health monitoring, transportation, education, workplaces, security, etc. In recent years attention has focused in health care applications.

The following are some definitions of Ambient Intelligence by different authors:

- A developing technology that will increasingly make our everyday environment sensitive and responsive to our presence. [15]
- A potential future in which we will be surrounded by intelligent objects and in which the environment will recognize the presence of persons and will respond to it in an undetectable manner. [16]
- A digital environment that proactively, but sensibly, assists people in their daily lives. [17]

Sensor networks play an important part in AmI. They acquire information from the environment which then is used for high-level reasoning, behavior monitoring, or reaction to the actual situation. Another important part is the middleware. It needs to be flexible and light-weight since it is the glue for sensors and actuators and also provides seamless connection from devices to users. The middleware is in charge of the data conversions, data fusion and data storage.

In order to provide accurate services, the system must be aware of the context which involves 'knowing' the state of the participants and the environment. The state of the environment can be determined by different types of sensors like temperature sensors, pressure, humidity, time of the day, etc. The state of a person involves his current physical activity, the intentions, the mood, schedule, etc. The system also should be aware of what is happening between several users and between groups of persons, i.e, their interactions, intentions and situation.

This work focuses on detecting the state of a person which includes detecting the physical activities he/she is performing. In the last years the use of smartphones has increased substantially. A smartphone is provided with several sensors like accelerometers, light sensors, pressure, proximity, magnetic, Wifi, GPS, etc. Since smartphones have become a 'must have' for many people, the sensors they include can be used to determine the context and the state of a person in a non-obtrusive manner. The raw data gathered from the sensors can be fused and analyzed in order to detect patterns and infer situations. The next section gives an introduction to *machine learning* which consists of several methods for pattern recognition and is the basis for this work.

2.2 Machine Learning

With the advent of information technologies the amount of data that is generated everyday is growing at a fast pace. Trying to extract information and knowledge from that vast cumulus of data is a time consuming (if not impossible) task to do by hand. Given that the computational power of machines has been increasing in the last years, it would be desirable to use that power to process that huge amount of data to generate knowledge from it. This is where machine learning comes into play. Machine learning can be thought of (but not limited to), as a set of algorithms that automatically find interesting patterns and relationships over the data. In our case, the data consists of sensor readings (accelerometer) and we want to find the relationship between those readings and the physical activity that the person bringing the sensor is performing.

Machine learning has been used extensively in human activity recognition. It is a subfield of Artificial Intelligence (AI) and has applications in industry, medicine, economics, natural and technical sciences, ecology, finance, and many others. Machine learning has been used for data analysis and data mining, learning to plan, game playing, text classification, speech recognition, handwriting, image/video processing, etc. *"The basic principle of machine learning is the automatic modeling of underlying processes that have generated the collected data."* [18].

In [18] *learning* is defined as '*any modification of the system that improves its performance in some problem solving task*'. The result of learning is knowledge that the system can use to solve new problems. An algorithm infers the properties of a given set of data and that information allows it to make predictions about other data that it might see in the future. This is possible because almost all nonrandom data contains patterns which allows a machine to generalize [19].

Machine learning can be divided into three major areas: *Supervised, Unsupervised* and *Reinforcement learning* (Figure 2.1). Supervised learning can be divided into two branches: *Classification* and *Regression*. Unsupervised learning also can be divided into branches: *Clustering* and *Associations*. Note that this is not a strict taxonomy of Machine learning areas but will serve as a guide. Associations, Regression and Reinforcement learning are not covered in this work, for more information on the first two topics see [18] and for the last one see [20].

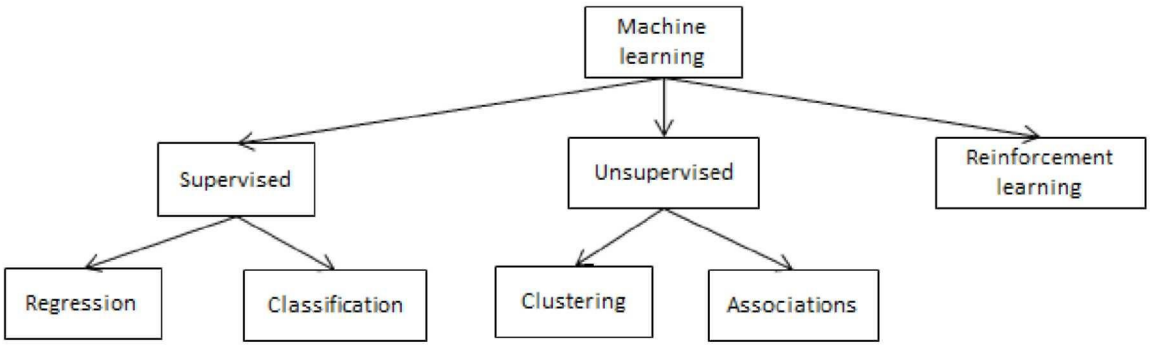

Figure 2.1: Machine learning methods

2.2.1 Concepts

Before going into the details of supervised and unsupervised learning some concepts and common terminology must be defined. In machine learning two types of algorithms may be defined: *learning* and *execution* algorithms. The former generates new knowledge from the set of input data or background knowledge. The latter uses the generated knowledge to solve new problems. The generated knowledge is known as the *model* sometimes also called *hypothesis* or *theory*.

In machine learning, given a model space and an optimality criterion, a model satisfying this criterion is sought. A criteria to measure the quality of the hypotheses is needed. A common criteria is the Occam's razor principle which states that the simplest explanation is also the most reliable. Different criteria are used for different learning problems and some of them are [18]:

- maximizing the prediction accuracy,
- minimizing the hypothesis' size,
- maximizing the hypothesis' fitness to the input data,
- maximizing the hypothesis' comprehensibility,
- minimizing the time complexity of prediction,
- minimizing the number of parameters required for prediction,
- maximizing the hypothesis' probability according to the background knowledge and input data.

To define the next concepts, suppose we have a set of persons in tabular form like in Table 2.1:

Table 2.1: A table of persons

gender	height (m)	weight (kg)	degree
male	1.7	70	Bachelor
female	1.55	65	Master
male	1.65	77	PhD.
female	1.6	55	Master

Each row represents an *instance* (also called a *feature vector* since they are usually represented as vectors). An *instance* 'encapsulates' the information for each person and has specific values for each of the columns. In machine learning the columns are usually called *attributes*. In the example, there are four attributes: gender, height, weight, and degree. If we wanted to classify a person as being male or female based on the other 3 attributes then *gender* becomes the *class*(also called *type* or *label*).

Attributes may take either Numerical or Nominal values. Numeric attributes also called continuous attributes are either integer or real valued numbers. Nominal attributes also called categorical attributes take values from a finite set of possibilities. In the example, height and weight are numeric attributes and gender and degree are nominal.

2.3 Supervised Learning

In supervised learning the algorithms are presented with a set of classified instances from which they learn a way of classifying unseen instances. It is called *supervised* because the scheme operates under supervision by being provided with the actual outcome for each of the training instances. The success of the classification can be measured by testing the generated model with an independent set of instances for which the true classifications are known but are hidden to the classifier [21]. When the attribute to be predicted is numeric rather than nominal it is called *regression*.

When evaluating the performance of different methods, it is a common practice to divide the set of instances into two sets. The training and the test set. The training set is used to build the classifier model, then the test set is used to measure the accuracy of the classifier, i.e, how well it generalizes to unseen instances. This method works well when we have a huge amount of data. If the data is limited *cross validation* can be used to test the built model. *k-fold cross validation* consists of splitting the entire data set (training set + test set) into k partitions and k iterations are performed. In each iteration, $k-1$ partitions are used to build the model and 1 partition is used to test it. At the end of the process each partition was used once to test the model and $k-1$ times to build the model. When k = total number of instances this is called

leave-one-out cross validation. The typical value for k is 10 and this has become like a 'standard' value.

In the remaining of this section three base-level classifiers are described: Naïve Bayes, Decision Trees, and K-Nearest Neighbors.

2.3.1 Naïve Bayes

This method is based on Bayes' theorem:

$$\Pr(H \mid E) = \frac{\Pr(E \mid H)\Pr(H)}{P(E)} \tag{2.1}$$

Where $\Pr(H \mid E)$ is the *posterior*, the probability of the hypothesis H given the evidence E. $\Pr(H)$ is the *prior* probability of H. $\Pr(E \mid H)/\Pr(E)$ represents the support E provides for H.

Now suppose there are two classes C_1 and C_2 and we want to classify a new instance. First we compute:

$$\Pr(C_i \mid E) = \frac{\prod_{e \in E} f(e|C_i)\Pr(C_i)}{P(E)}$$

for each class and the result will be the class with higher probability. The denominator can be ignored because the probabilities can be normalized to sum 1.

Usually, numeric values are assumed to have a Gaussian probability distribution, so the function f can be defined as:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ and σ are the mean and standard deviation for each class and each attribute.

The method is called Naïve because it assumes independence between attributes, but despite of this assumption the method performs very well in practice.

2.3.2 Decision Trees

Decision tree learning was developed by Quinlan [22] and he has improved the algorithm over the years. ID3 is the name of the method that introduced the use of *information gain* concept. Quinlan made improvements over the ID3 to handle numeric attributes, missing values, generate rules from trees, etc. This improved version is called C4.5. Decision trees are simple to understand and interpret since they can be graphically represented. They are good for discovering patterns inside the data since they use a white box model, i.e., the output of a classification can be traced back in order to understand the logic of the decision. An example of a black box model is

an Artificial Neural Network because it is difficult to understand the explanation of a result.

In a decision tree each internal node corresponds to one attribute and has edges that correspond to each possible value for that attribute. A leaf represents the predicted class by following the nodes from the root to that leaf. To construct a decision tree first an attribute must be chosen to be at the root. Then add an edge for each possible value for that attribute, this will divide the instances set into subsets, one for every value of the attribute. The process is repeated recursively for each edge until all instances at a node have the same classification. The crucial step is how to choose the best attribute to be at the root of every sub tree such that the depth of the final tree is small and still consistent with the data. In order to choose the best attribute we need a measure of how 'good' an attribute is and this is where the concept of *information gain* comes into play.

The amount of information depends on the prior knowledge. For example if we toss a fair coin the information we get should be greater that the information we would get if we knew the coin comes heads 99% of the times, i.e., as more prior knowledge the less information we will gain from the actual result. The information of an answer is computed as:

$$I(\Pr(v_1), \ldots, \Pr(v_n)) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i) \tag{2.2}$$

where v_i is a possible answer with probability $\Pr(v_i)$ and the result is expressed in units called *bits*. For the toss of a fair coin we have:

$$I\left(\tfrac{1}{2}, \tfrac{1}{2}\right) = -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \text{ bit},$$

If the coin is loaded such that 99% of the times it comes heads then:

$$I\left(\tfrac{1}{100}, \tfrac{99}{100}\right) = -\tfrac{1}{100}\log_2\tfrac{1}{100} - \tfrac{99}{100}\log_2\tfrac{99}{100} = 0.08 \text{ bits}.$$

For an example consider the NAND logical gate (Table 2.2). To construct a decision tree first we need to choose an attribute as the root node. The best attribute is the one with the greatest information gain value. At the beginning we have two attributes to choose from: A and B. Figure 2.2 shows the two options we have. The leafs show each instance's class. For example, in Figure 2.2.a there are two instances with A=0 and two with A=1. The two instances with A=0 have a class=1 and the two instances with A=1 have a class of 1 and 0 respectively.

Next the information gain for attribute A is computed. Since it has 2 possible values we compute the information for each:

$I\left(\frac{2}{2}, \frac{0}{2}\right) = -\frac{2}{2}\log_2\frac{2}{2} - 0 = 0$ bits. Similarly, $I\left(\frac{1}{2}, \frac{1}{2}\right) = 1$ bit. Now we calculate the average information value of these as follows:

Table 2.2: NAND truth table

A	B	output
0	0	1
0	1	1
1	0	1
1	1	0

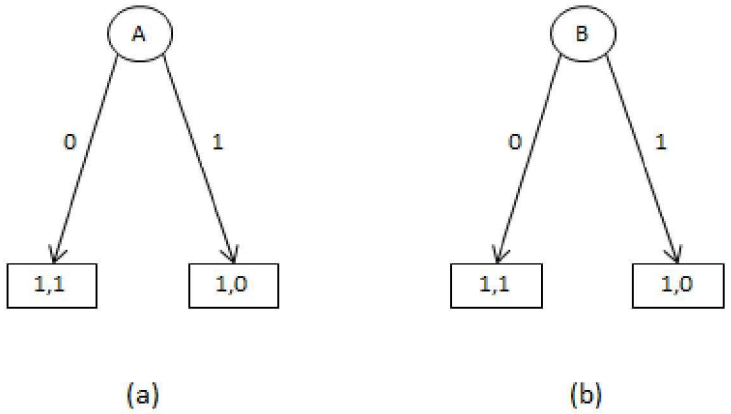

Figure 2.2: Partial decision trees

$$avginfo\left(\left[\tfrac{2}{2},\tfrac{0}{2}\right],\left[\tfrac{1}{2},\tfrac{1}{2}\right]\right) = (2/4)\times 0 + (2/4)\times 1 = 0.5 \text{ bits.}$$

The initial information without considering any attribute is $I\left(\frac{3}{4},\frac{1}{4}\right) = 0.8$ bits, thus attribute A has an information gain of: $gain(A) = 0.8 - 0.5 = 0.3$ bits. Now we compute the information gain for attribute B $gain(B) = 0.3$, and choose the one with the greatest value. In this case both are the same so A is arbitrarily chosen.

Now the process is repeated for each of the edges. All instances of the edge A=0 have class=1 so we are done. For the edge A=1 we have to choose a new root for that subtree. Since there is only attribute B left we choose it. Attribute B has two possible values so we create two edges (Figure 2.3). Since there is only one instance in each of the leafs we are done.

The NAND example consisted of just nominal attributes. One way to deal with numeric attributes is to sort them and split them at the middle. Then the information gain is calculated as usual. Another option is to split the values every time the class changes. Sometimes the numeric attributes can be discretized as a preprocessing step.

2.3.3 Instance Based Learning

In instance base learning also called lazy learning all training instances are stored and a model is not built in the training phase. One of the advantages of lazy classifiers is

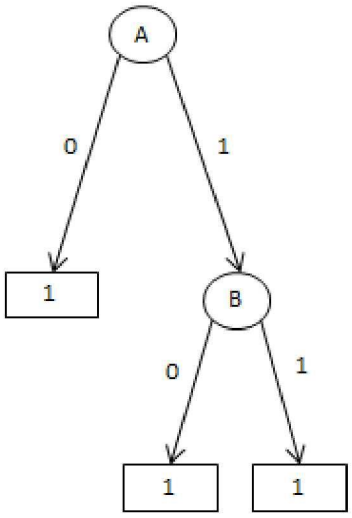

Figure 2.3: Final decision tree

that the training phase is very fast(just add the new instances!). K-Nearest Neighbors (KNN) is the most common classifier of this type. The process consists of exhaustively searching for the k nearest instances from the query instance and assign it the most common class among those k neighbors. The two most commonly used metric distances are the Euclidean and Manhattan which can be generalized with the p-norm:

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \tag{2.3}$$

where p is a real number and $p \geq 1$. To get the Manhattan distance p is set to 1

$$\|x\|_1 := \sum_{i=1}^{n} |x_i|, \tag{2.4}$$

and for the Euclidean distance set $p = 2$

$$\|x\|_2 := \sqrt{\sum_{i=1}^{n} |x_i|^2} \tag{2.5}$$

Let q and t be two instances. q is the query instance (the one with unknown class) and t is an instance from the training set. To calculate the euclidean distance between them we use:

$$d(q,t) = \sqrt{\sum_{i=1}^{n} (q_i - t_i)^2}, \tag{2.6}$$

where n is the number of attributes, and q_i and t_i is the value of the i^{th} attribute of the instances q and t respectively. It is common to omit the square root computation since for classification it is not required to obtain the actual distance since we are interested in just how 'different' two instances are. Since numeric attributes are not always in the same scale it is a good practice to normalize them to lie between 0 and 1. When comparing the difference between nominal attributes a and b we can use the following rule:

$$dif(a,b) = \begin{cases} 1 & \text{if } a \neq b, \\ 0 & \text{if } a = b. \end{cases} \tag{2.7}$$

In real life not all attributes are equally relevant. To compensate for this a weighting value can be added to each attribute:

$$d(q,t) = \sqrt{\sum_{i=1}^{n} w_i (q_i - t_i)^2}, \tag{2.8}$$

this w_i can be determined according to some domain knowledge or by automatically learning it. Another improvement is to use weighted neighbors to compensate for the fact that the algorithm may be using neighbors that are too far away [19].

The parameter k is specified according to the application but usually it is set to an odd number. In real life applications there may be noisy instances. To deal with them we can increase the value of k, but if it is too big the probability that different instances from the query one are selected will increase. Another method to deal with noisy instances is to monitor their performance and delete the ones that have a poor performance. The performance increases if the instance contributes to a correct classification, otherwise it decreases.

The KNN algorithm is linear in the number of training instances. To make it more efficient a kD-tree can be used [21]. Another improvement is to reduce the number of training instances. Some instances may be redundant and may be deleted. In [23] several reduction techniques are presented.

2.4 Unsupervised Learning

In unsupervised learning the class of the instances is not known. Unlike in supervised learning, there is no training set with labels assigned to the instances. The task of unsupervised learning is to find those 'hidden' classes that may arise naturally from the data itself. One type of unsupervised learning is *clustering* which is the process of

grouping the data into classes or clusters, so that objects within a group have high similarity but are very dissimilar to objects in other clusters [24]. The discovered groups may be of several types:

- **Exclusive.** Any instance belongs to only one group.
- **Overlapping.** An instance may be in different groups.
- **Probabilistic.** An instance belongs to each group with a certain probability.
- **Fuzzy.** An instance has a degree of membership to each cluster.
- **Hierarchical.** A tree like structure of instances with more general instances at the top and more specific instances at the leafs.

There are many types of clustering methods and can be categorized as follows:

- **Partitioning methods.** These methods construct k partitions where each partition represents a cluster and $k \leq n$. Additionally, the following requirements must be met: (1) each group must contain at least one instance, (2) each instance must belong to exactly one group (some methods relax the latter requirement by allowing an instance belong to each group in some degree, e.g., in fuzzy clustering). This methods generally start by creating k arbitrarily chosen groups and then iterate a predefined number of times or until convergence. In each iteration the instances are moved from one group to another. Examples of these methods are k-means and k-medoids which find spherical-shaped clusters.
- **Hierarchical methods.** This methods build a hierarchy of clusters and they can be divided in two types: *1) Agglomerative* also called bottom-up approach. This starts by assigning an instance to its own cluster and then it merges objects or groups that are close to each other until a termination condition or when all groups are merged into one. *2) Divisive* also called top-down approach. This starts with all instances in one cluster which is then recursively split into smaller clusters until a condition is met or when each instance is in one cluster. After the clustering is finished one can see the results as a dendrogram.
- **Density based methods.** In partitioning methods, instances are grouped based on the distance between them. Density based methods starts by growing a given cluster until the *density* of the neighborhood exceeds a threshold. The *density* is the number of instances within a radio. This can be used to detect outliers and unlike partitioning methods, the resulting groups can have arbitrary shapes (Figure 2.4)[1]. Examples of these methods are DBSCAN and OPTICS, among others.

[1]Taken from: https://commons.wikimedia.org/wiki/File:DBSCAN-density-data.svg

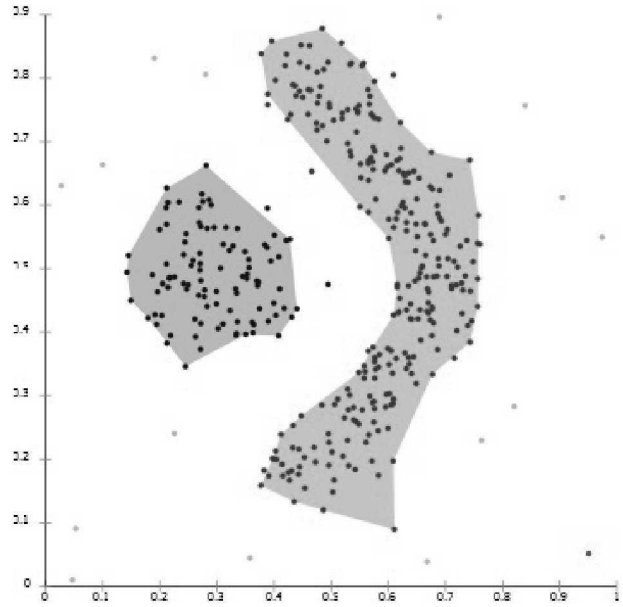

Figure 2.4: Found clusters by DBSCAN

2.4.1 k-Means

With this method you specify the number k of groups to be formed. Then the algorithm starts by choosing k random points called *centroids* each instance is assigned to the closest centroid (usually with the Euclidean distance). Next, it calculates the mean of the instances in every cluster and this will become the new centroid. The process repeats until convergence or until a predefined number of steps. The aim is to minimize the within cluster sum of squares.

$$E = \sum_{i=1}^{k} \sum_{p \in Ci} |p - m_i|^2, \tag{2.9}$$

where p is the point that represents an instance and m_i is the mean of cluster C_i. The algorithm is shown in the next code snippet.

Algorithm 1 k-means

1: choose k centroids at random
2: **repeat**
3: assign each instance to the closest centroid
4: update the centroid to be the mean value of the instances belonging to that centroid
5: **until** convergence

The convergence criterion is when the centroids no longer move. The algorithm always terminates but not necessarily finds the global optimum. The algorithm is

sensitive to the initial randomly selected centroids. Sometimes it is convenient to run it several times to reduce this effect.

2.5 Related Work

The goal of activity recognition is to recognize common activities in real life environments. In recent years simple human activity recognition has been achieved successfully, however complex activity recognition is still challenging and is an active area of research. In [25] they pose the following challenges regarding the nature of human activities: *Recognizing concurrent activities, recognizing interleaved activities, ambiguity of interpretation and multiple residents.*

- **Concurrent activities.** People can do several activities at the same time. For example, watching TV while exercising.
- **Interleaved activities.** If someone is cleaning a house and he receives a phone call he may stop cleaning and answer the phone call and then he may continue with the cleaning.
- **Ambiguity of interpretation.** For example, the activity 'running' could mean that a person is doing exercise or he/she may be in a hurry.
- **Multiple residents.** The activities that are being performed by several residents as a whole need to be recognized.

2.5.1 Physical Activity Recognition

Generally, simple activities do not depend on the situation, i.e., they can exist by themselves and they last only a few seconds. Examples of this type of activities are: *running, walking, resting, sitting,* etc.

Brezmes, Gorricho and Cotrina [5] implemented a real time activity recognizer on a mobile phone. This was one of the first works to take advantage of a mobile's phone accelerometer without the need of attaching several sensors to the body. They used a K-nearest neighbors approach achieving accuracies ranging from 70% to 90% for several activities (walking, climbing-down stairs, climbing-up stairs, sitting down, standing up, falling). They improved the accuracy by adding more training sessions to the database, however their paper does not include information of which features they used or which data is taken into account to compute the distances by the KNN algorithm.

Mannini and Sabatini [26] used five bi-axial accelerometers located at the hip, wrist, arm, ankle, and thigh and they reported accuracies between 93% and 98.5% for

seven different activities (sitting, lying, standing, walking, stair climbing, running and cycling). They did a very extensive work and tried different approaches (probabilistic, geometric and binary decision).

Karantonis, et al. [27] presented an implementation of a real time activity classifier capable of computing the metabolic energy expenditure. In this work a wireless sensor unit is used which sends the information to a local computer for display and evaluation. Ravi, et al. [4] made a comparison of base-level classifiers and meta-level classifiers and concluded that combining classifiers using Plurality Voting turned out to be the best choice for the recognition of simple activities. Mi Zhang [28] proposed a Bag-of-Features approach which builds activity models using histograms of primitive symbols. Recently, the Bag-of-Features approach has gained significant interest.

2.5.2 Complex Activity Recognition

Complex activities are composed of a collection of simple activities and may consider information from the context, time, and interactions between other persons and objects. The recognition of these activities generally requires more sensors and a fixed infrastructure (video cameras, RFID tags, several accelerometers, magnetic sensors, etc.).

Tao Gu, et al. [13] built activity models by mining a set of Emerging Patterns from a sequential activity trace and used them to recognize sequential, interleaved, and concurrent activities achieving accuracies of 90.96%, 87.98% and 78.58%, respectively. In their work they measured user's movement, location, the living environment (temperature, humidity and light) and human object interaction, i.e., the objects a user touches. They used RFID tags, and several sensors attached to a user's both hands and waist.

Tam Huynh, et al. [29] used topic models to recognize activities such as: dinner, commuting, lunch and office work. They automatically extracted activity patterns from sensor data (3D accelerometer, clock, binary tilt switches, temperature sensor, and two light sensors) to enable the recognition of daily routines as a composition of such activity patterns. A topic model is a statistical model that is often used in document classification to automatically discover the 'themes' that emerge from a set of documents [30]. Experimental results obtained by Tam Huynh, et al. [31] suggest that the recognition of complex activities can be achieved with the same algorithms of simple activities. The complex activities they recognized were preparing for work, going shopping and doing housework. Tian, et al. [32] use accelerometer and GPS information to automatically send updates to a micro-blogging website. They used Hidden Markov Models for the activity recognition and increased the accuracy by constraining the context using GPS location data.

Stefan, et al. [33] recognized simple and complex activities like cleaning, cooking,

medication, sweeping, washing hands and watering plants. They used fixed time window lengths with an overlap of one half of the time window length. They used simple statistical features from the accelerometer and orientation sensor of a smart phone. For simple activities they achieved accuracies of 90% and 50% for complex activities. The accuracy for complex activities increased to 78% when they used the entire activity without dividing it into fixed window lenghts. They also showed that orientation information increased the accuracy over pure acceleration data.

2.5.3 Summary of Different Approaches

The next Table shows that due to its simplicity and accuracy accelerometers are the main sensors used in activity recognition. For complex activities more sensors (RFID, clock, temperature sensor, light sensors, etc.) are required.

Table 2.3: Different Approaches for Activity Recognition

Ref.	**Activity Type**	**Sensors**	**Approach**	**No. Activities**	**Accuracy [%]**
[5]	Simple	1 tri-axial accelerometer	KNN	6	70-90
[26]	Simple	5 bi-axial accelerometers	HMM	7	93-98.5
[27]	Simple	1 tri-axial accelerometer	Decision Tree	12	90.8
[28]	Simple	1 tri-axial accelerometer, 1 tri-axial gyroscope	Bag of Features	9	92.7
[9]	Simple	Video sequences	Bayesian Classifier	5	1.5 error rate
[13]	Complex	3 iMote2 sets, 2 RFID wristband readers	Emerging Patterns	26	78.58-90.96
[29]	Complex	3D accelerometer, real time clock, 9 binary tilt switches, temperature sensor, 2 light sensors	Topic Models	34	72.7
[31]	Complex	2D accelerometer, 9 binary tilt switches	K-means, SVM, Nearest Neighbor, HMM	3	80.6-91.8

Chapter 3

Method

The aim of this chapter is to describe the methodology for recognizing human activities. First, it begins with an explanation of the sensing platform and then with the data collection process, how was the training performed and the classification process. This work was divided into three phases:

1. Physical activity recognition.
2. Contextualizing physical activities using information gathered from Wi-Fi access points.
3. Complex activity recognition.

Each phase has its own section which includes the data collection process, the training and classification steps. The experiments and results are also divided into each of the three phases and are presented in Chapter 4.

3.1 Sensing Platform

For the data collection an LG Optimus Me smartphone was used. This model comes with Android 2.2 Operating System[1]. It includes an STMicroelectronics triaxial accelerometer. It returns the acceleration value for each of the axes(x,y,z) and a *timestamp*. Its maximum range is $\pm 19.60 m/s^2$.

An accelerometer is an electromechanical device that measures acceleration forces. These forces may be static, like the constant force of gravity pulling at your feet, or they could be dynamic caused by moving or vibrating the accelerometer[2]. The x-axis runs parallel to the width of the smartphone, the y-axis parallel to the height of the phone and the z-axis perpendicular to its face (see Figure 3.1). Figure 3.2 shows an example of 10 samples returned by the sensor. First column is the timestamp in nanoseconds, and the acceleration in x,y,z axes respectively.

[1]Android 2.2 Platform, http://developer.android.com/sdk/android-2.2-highlights.html
[2]Dimension Engineering, LLC. http://www.dimensionengineering.com/accelerometers.htm

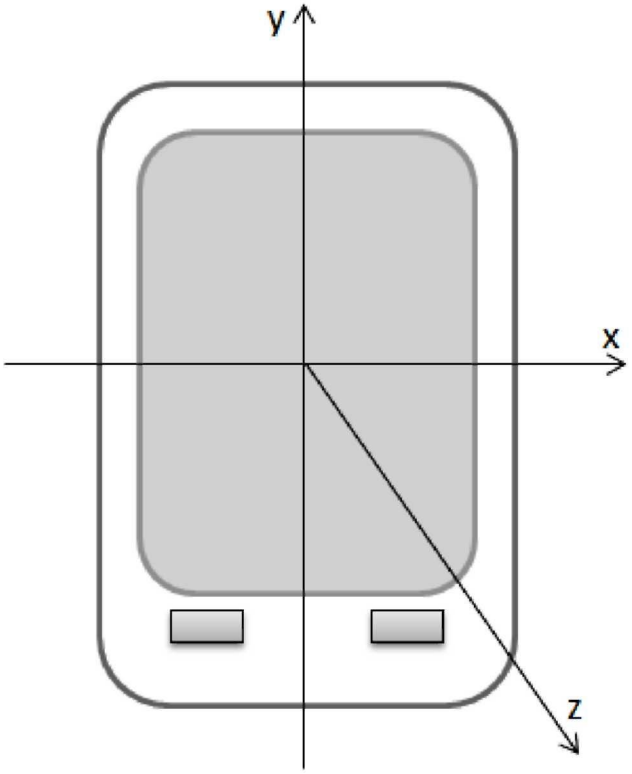

Figure 3.1: Acceleration axes

```
16939977023294,-0.3138128,9.316318,-4.8052583
16940001298295,-0.2745862,8.149326,-4.2070527
16940025464963,-0.97085834,11.218807,-2.5399222
16940049528297,-1.147378,11.983726,-2.128043
16940073684964,-1.1964113,12.170053,-2.0299766
16940098031631,-1.4317709,10.993255,-2.5301156
16940123279965,-1.4906107,10.699055,-2.657602
16940149138298,-1.5004175,10.630408,-2.687022
16940173304966,-2.2163029,7.139241,-1.9613299
16940185709299,-2.3339827,6.560649,-1.8436501
```

Figure 3.2: Example of values returned by the accelerometer for each of the axes

3.2 Physical Activity Recognition in Real Time

Here, the method for physical activity recognition is presented. Examples of physical activities are *walking, running, climbing-up stairs, resting, etc.* These type of activities last a few seconds and do not depend on the situation, i.e, they can exist by themselves. The recognition was performed by means of a supervised learning approach and several base-level classifiers were compared.

3.2.1 Data collection

The activities to be recognized were five: *walk, run, climbing-up stairs, climbing-down stairs and resting.* Those activities were chosen because they are very common, can be performed in almost any place and are the building blocks for higher level activities such as exercising, commuting, working, etc. Training sessions were performed for each of the five activities. Each training session consisted of collecting data during 1 minute. An Android application was developed in order the ease the data collection process, as shown in Figure 3.3. The sampling rate was set at 40Hz (a new sample every 25 milliseconds) so each minute corresponds to 2400 samples. The cellphone was placed in the users' belt in a vertical position. This is because it is common to use the cellphone in this position and placing it in the pants' pocket may not give accurate results since it will be constantly shifting. Figure 3.4 shows the graphs for the acceleration for each of the five activities. It can be seen that y-axis values are generally greater, this is because when the smartphone is in vertical position without movement the values are approximate to 9.81 which is equivalent to 1g.

Figure 3.3: Application for the data collection process

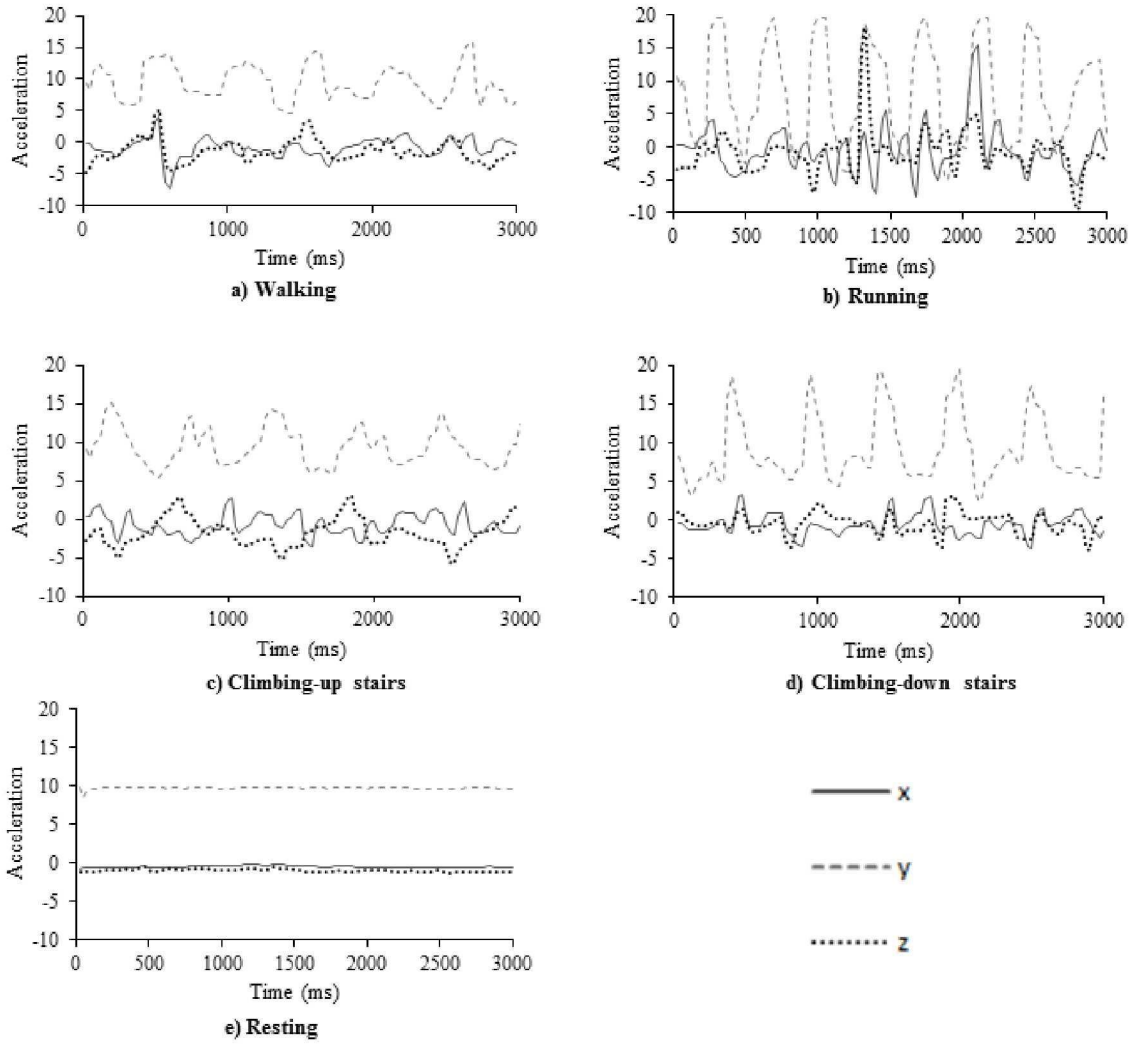

Figure 3.4: Accelerations for each of the five activities

3.2.2 Training

The training phase consists of extracting a feature vector from a fixed length window. Each instance consists of the data gathered from 2 seconds (80 samples from each of the axes). The 2 second window length was chosen because in 1 second there is still not enough information to be able to differentiate each activity. From the 2 second window 9 features were extracted which are the mean and standard deviation for each of the axes and the correlation between each pair of axes. In similar works they have used more attributes like the average resultant acceleration, time between peaks, binned distribution, etc. [6], but since the aim is to perform the recognition in real time simple features were used in order to diminish the smartphone's computational work load. During 1 minute 30 instances for the activity are generated. For KNN algorithm

once the instances are generated it is said to be already trained. For decision trees and Naive Bayes before the classification can be done, their models must be built. The classifiers models were built using the Weka data mining software[3].

3.2.3 Classification

First, the classification was performed 'offline' with 3 classifiers: KNN, C4.5 decision tree, and Naive Bayes. Additionally, a prototype running on a cellphone was developed which runs in real time. The prototype was implemented using the KNN algorithm because being a *lazy* classifier it allows the introduction of new training instances in real time without incurring in training time. For the classification in real time, the training instances are loaded into memory. Then, every second a query instance is created by extracting the feature vector from the last 2 seconds of acceleration data. Finally, the query instance is classified.

3.3 Contextualizing physical activities using information gathered from Wi-Fi access points

This section explains the implementation of a real time activity recognizer running on a cellphone. First, physical activity recognition from accelerometer data is performed and then, this information is fused with data from Wi-Fi access points to classify the activity being performed by the user.

The number of Wi-Fi access points around the world has increased significantly in the last years. They are installed in many places such as restaurants, hotels, schools, parks, airports, etc. Since every access point has a unique identifier namely, the BSSID (Basic Service Set Identifier), it is possible to use this information along with the signal strength for localization and tracking purposes [34, 35, 36].

The objective is not to infer the spatial location of the user but to determine the user's context by means of a supervised learning approach. For example, if walking is detected as a physical activity and the Wi-Fi yields information that tells us that the user is in a library, then we can classify the whole activity as looking for a book and another application can use this information maybe to suggest related titles. If the physical activity is resting the whole activity may be classified as reading a book. In this case the user's cell phone could block unimportant incoming calls to avoid interruptions. The role of the access points is to aid in the discrimination process providing approximate location information, i.e., we can use a fixed set of physical activities as the basis and combine them with 'location' information to generate contextualized activities.

[3]Weka Data Minig Software. http://www.cs.waikato.ac.nz/ml/weka/

The approach lies between physical and complex activities in the sense that first, the physical activity is recognized and then Wi-Fi information is added to contextualize it. This work differs from the ones presented in the related work section in the following aspects: First, we take advantage of existing infrastructure (Wi-Fi access points) so our approach does not require the addition of sensors to the environment like RFID tags, video cameras, etc. Second, aside from the presence of in range access points, we do not need a fixed configuration of the environment. Since we are just reading the BSSID and signal strength to classify the activities, we do not need to configure each access point neither know their physical location. Finally, we focused in using sensors that are commonly available in most smartphones so the user is freed from having to wear several sensors attached to his/her body.

3.3.1 Data collection

An Android 2.2 application running on an LG Optimus Me cell phone was used to collect the data from the accelerometer and Wi-Fi (Figure 3.5). The user specifies an Id and a label for the activity and starts collecting data by pressing the 'Start' button. The cellphone was placed in the user's belt and the data collection consisted of 8 activities: 1)reading in bedroom A, 2)watching television, 3)reading in bedroom B, 4)sitting in the lobby, 5)reading in the library (first floor), 6)looking for a book in the library (first floor), 7)reading in the library (second floor), 8)looking for a book in the library (second floor). Activities 1-4 were performed in an apartment building while activities 5-8 were performed in a library. The data collection process was performed by two participants under supervision. The test set was collected independently in a different day from the training set.

From the tri-axial accelerometer sensor, we read the acceleration values from each of the *x,y,z* axes and classify the *physical activity* being performed as one of *walking, running or resting*. The approach for recognizing the physical activity was a KNN algorithm as described in the previous section 'Physical Activity Recognition in Real Time'. From the Wi-Fi sensor, data from the in range Wireless access points was collected. Specifically, their BSSID (Basic Service Set Identifier) and signal strength was collected. Activities that are very close to each other were selected. Figure 3.6 shows the layout of the 3^{rd} floor of the apartments building. The lobby is located below the room marked with *Reading in bedroom A* but in the 1^{st} floor.

Figure 3.5: Application for the data collection process

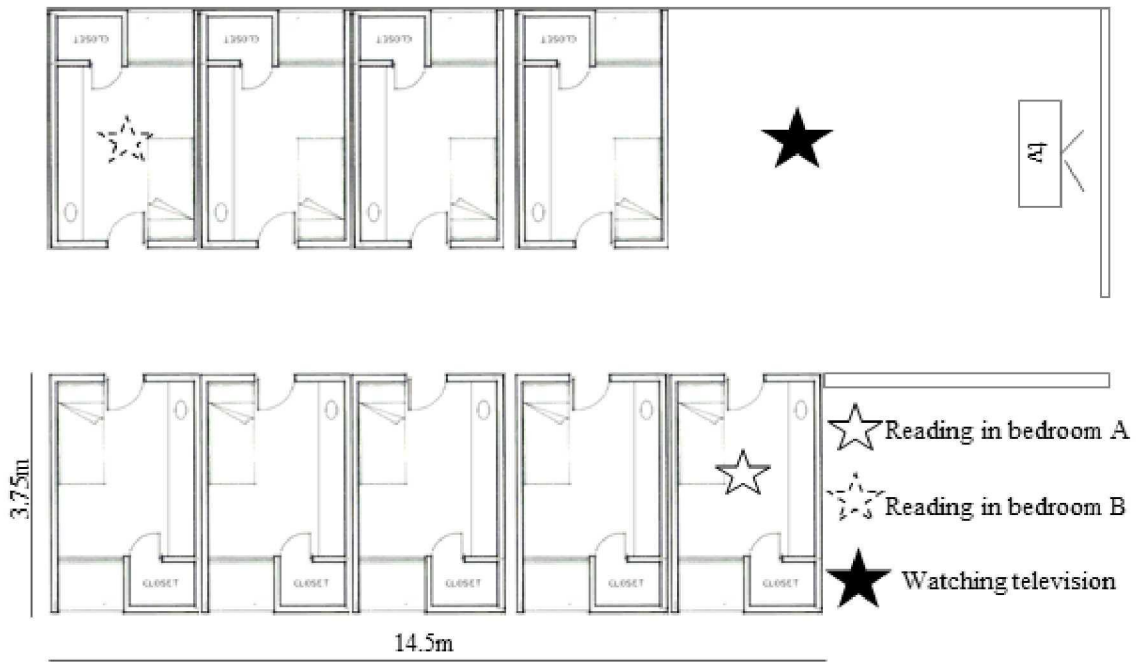

Figure 3.6: Layout of the apartments building 3rd floor.

3.3.2 Training

In the training phase the instances that will be used as the training set are generated. An instance based learning algorithm (K-nearest neighbors) was used for the contextualized activities recognition. Each activity instance has 3 attributes, as shown in Table 3.1. Figure 3.7 shows the process of generating one training instance. First, the application performs two scans to collect the data of the in range access points. A delay of 500ms is set between the scans. Then, every second the physical activity being performed by the user is recognized and stored in a vector V. This is done during 5 seconds, i.e., at the end of the 5 seconds, the vector V will contain 5 ids' (one for each detected physical activity). Finally the application performs two more scans to gather information from the access points. The reason of doing several scans is because in [37] they observed that sometimes one or more access points may not be detected because limited sensitivity of the hardware and/or long beacon interval of some access points. Now the instance is created and its physical activity is set to $Mode(V)$, i.e., the physical activity that was dominant across the 5 seconds period. For every access point found during the scans a pair <*bssid,strength*> is added to L, where *bssid* is the access point identifier and *strength* is the mean of the signal strength from the 4 scans. Figure 3.8 shows an example of the collected training instances for the *reading in bedroom A* activity. First column shows the contextualized activity's Id. The second column shows the unique identifier for a particular instance. Note that the same identifier may be in several rows since an instance may have several access points. The Third column shows the access point's MAC address. The fourth column the signal strength as received by the device. The fifth column shows the recognized physical activity's Id.

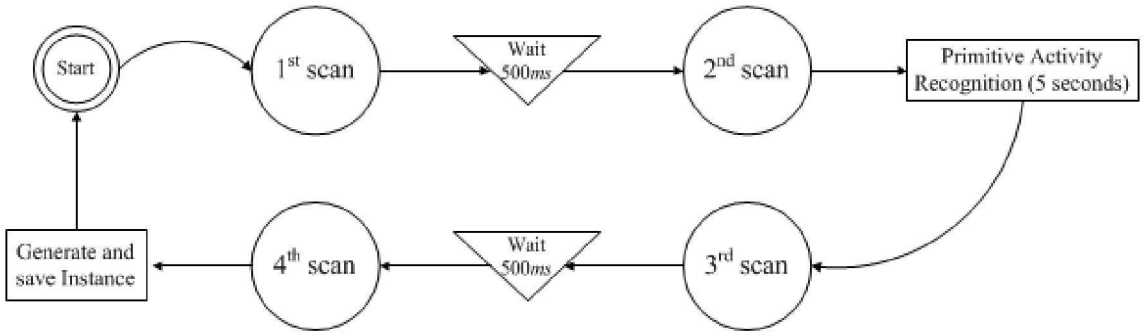

Figure 3.7: Steps to generate one training instance

Table 3.1: Instance's attributes

Name	Description
Id	Unique identifier of the instance (for debugging purposes)
Class	A number from 1 to 8 to identify which *contextualized activity* the instance belongs to.
Physical activity	Identifies the physical activity associated with this *contextualized activity*. 0 for walking, 1 for running and 2 for resting
List of access points	A list L in which each element is a pair <*bssid,strength*>

```
1,1328161246746,00:02:2d:61:ba:07,-88.75,2
1,1328161256783,1c:af:f7:98:d1:88,-86.75,2
1,1328161256783,f4:ec:38:a9:80:0a,-91.0,2
1,1328161256783,00:02:2d:61:ba:07,-91.25,2
1,1328161266827,1c:af:f7:98:d1:88,-89.875,2
1,1328161266827,00:02:2d:61:ba:07,-90.375,2
1,1328161276883,00:14:d1:ba:3c:ce,-91.0,2
1,1328161276883,1c:af:f7:98:d1:88,-91.75,2
1,1328161276883,00:02:2d:61:ba:07,-89.0,2
1,1328161287243,1c:af:f7:98:d1:88,-87.75,0
1,1328161297274,00:02:2d:61:ba:07,-89.75,2
1,1328161297274,00:14:d1:ba:3c:ce,-91.0,2
1,1328161307366,00:02:2d:61:ba:07,-90.25,2
1,1328161307366,00:26:44:68:ca:3d,-93.0,2
1,1328161317388,00:02:2d:61:ba:07,-89.75,2
1,1328161327395,00:02:2d:61:ba:07,-90.375,2
1,1328161327395,14:d6:4d:e4:82:c8,-90.5,2
1,1328161337403,00:02:2d:61:ba:07,-90.5,2
1,1328161337403,00:26:44:68:ca:3d,-88.0,2
```

Figure 3.8: Example of collected instances

3.3.3 Classification

Since the Wi-Fi access point information is gathered in real time it is not possible to construct a model beforehand and this is the reason a *lazy* classifier was chosen (KNN). For the Euclidean distance the difference between attribute values $dif(b_i - c_i)$ was computed as follows for $i = 1..3$ where b is the query instance and c an instance from the training set:

- Primitive activity Id. Set to 0 if both b and c have the same primitive activity.

$$dif(b,c) = \begin{cases} 0 & \text{if } P(b) = P(c) \\ 1 & \text{otherwise} \end{cases} \tag{3.1}$$

  where the function $P(a)$ returns the primitive activity associated with the specified instance.

- Ratio of same access points. The extreme cases are when both instances share the same access points (in this case the distance is 0) and when they do not have any common access point (in this case the distance is 1).

$$dif(b,c) = 1 - \frac{|L(b) \cap L(c)|}{|L(b) \cup L(c)|} \tag{3.2}$$

  where the function L returns the list of access points of the specified instance. Eq.(3.2) is known as the Jaccard distance[4].

- Difference of the signal strength's standard deviation. This is defined as:

$$dif(b,c) = 1 - (1/1 + \alpha) \tag{3.3}$$

  where $\alpha = abs(SD(a,b) - SD(b,a))$ and $SD(p1,p2)$ is a function that returns the standard deviation of the signal strength of all access points of *p1* that are also in *p2*.

For the real time classification, a query instance is crated in the same way a training instance is created (see Figure 3.7) and then it is classified.

[4]Jaccard distance https://en.wikipedia.org/wiki/Jaccard_index

3.4 Complex Activity Recognition

In this section the focus is on recognizing *complex activities* which can last from a few minutes to several hours and are composed of a set of physical activities. Specifically we recognize the following five activities: 1) *Commuting*, which can include a combination of physical activities like walking, running, driving, traveling by bus, stand still, etc. 2) *Working*, which can include physical activities like reading, walking, writing, etc. 3) *At home*, which can include physical activities like walking, resting, brushing teeth, sitting, etc. 4) *Shopping*, which is composed of physical activities like walking, picking groceries, paying, etc. 5) *Exercising*, that may include physical activities like walking, running, squats, crunches, resting, etc. These activities were chosen because they are very common to every person and they may greatly vary in duration so we can test whether the algorithms can deal with these time variation. For example commuting may take just 10 minutes while working may take several hours.

The *complex activity* recognition is based on the framework proposed by Zhang M. and Sawchuk A. [28] which is based on a Bag-Of-Features (BoF) approach and consists of building activity models using histograms of primitive symbols. In their work, Zhang M. and Sawchuk A. recognized nine *physical activities* and they achieved an overall accuracy of 92.7%. In this work we perform experiments based on that approach (BoF) in order to determine if it can be used to recognize higher level activities. We also propose a set of features over the histogram in order to increase the accuracy.

In this work, we focused in using sensors that are commonly available in most smartphones so the user is freed from having to wear several sensors attached to his/her body. In this case we used a cellphone's triaxial accelerometer. Surely, using just one sensor of this type imposes some limitations, e.g., we can detect if a person is sitting, but we cannot tell whether he/she is also reading, eating, watching TV, etc. A *complex activity* will be represented as a set of physical activities. Since our focus is on the former we are not going to label the physical activities (we will just identify them by an integer id).

In the following subsections the process shown in Figure 3.9 is described which includes: data collection, physical activity extraction, primitives generation, histogram generation, feature extraction/noise reduction, and finally the training and classification phase.

3.4.1 Data Collection

An Android 2.2 [5] application running on a LG Optimus Me cellphone was used to collect the accelerometer data from each of the x,y,z axes. The sample rate was set at 50Hz. The cellphone was placed in the user's belt and the data collection consisted

[5] Android 2.2 Platform. http://developer.android.com/sdk/android-2.2-highlights.html

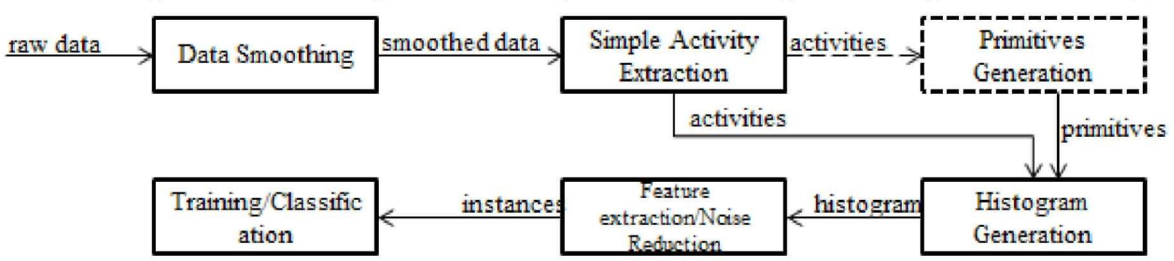

Figure 3.9: Overall process for complex activity recognition. Dotted lines mean that the process is performed just in the training phase.

of 5 complex activities: *commuting, working, at home, shopping and exercising.* A training and a test set were collected in different days. The duration of the activities varies from about 5 minutes to a couple of hours. The total recorded data consists of approximately 41 hours. The data was collected by one user. This is because some activities are user-dependent, e.g., the work of some person may involve being at the office most of the time while the work of someone else may be more physical demanding.

3.4.2 Physical Activity Extraction

Before the physical activity extraction, the raw data was smoothed with the moving average filter (Eq. 3.4) using a window length of 15 samples (the acceleration of each axis).

$$v_s(t) = \frac{1}{n} \sum_{i=t-n}^{t-1} v(i) \tag{3.4}$$

where v is the original vector, v_s is the smoothed vector and n is the window length.

Complex activities are composed of physical activities, so the next step is to extract the latter. We considered physical activities to have a window length of w seconds (i.e, $50 \times w$ samples since we are sampling at 50Hz) with an overlap of 33%.

Each physical activity is represented as a feature vector. To construct the feature vector we computed 14 features from each w seconds window. Table 3.2 shows the statistical features. These features were chosen because they are computationally efficient to calculate and gave us good results when recognizing physical activities. Additionally we used two of the physical features described in [28], the *average of the movement intensity* and the *variance of the movement intensity*:

AI. Average of movement intensity.

$$AI = \frac{1}{T}\left(\sum_{t=1}^{T} MI(t)\right) \tag{3.5}$$

VI. Variance of movement intensity.

$$VI = \frac{1}{T}\left(\sum_{t=1}^{T} (MI(t) - AI)^2\right) \tag{3.6}$$

MI. The movement intensity is computed as follows:

$$MI(t) = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2}, \tag{3.7}$$

where $a_x(t)^2$, $a_y(t)^2$ and $a_z(t)^2$ are the accelerations at time t from the w second window.

Table 3.2: Physical Activity Features

Feature	**Description**
mean	The mean value of the values over the window for each of the axes
standard deviation	The standard deviation of the values over the window for each of the axes
correlation	Pearson correlation for each pair of axes xy, xz, yz
derivatives mean	The mean value of the derivatives over the window for each of the axes

At the end of this process we have a set of physical activities that arose from each of the *complex activities*. At this moment the physical activities do not have a *type*. In the next section *Primitives generation* we describe how *primitives* are constructed from these *"un-typed"* activities.

3.4.3 Primitives Generation

This step is just performed in the training phase. *Primitives* are the building blocks of *complex activities*. To generate these *primitives*, we performed clustering on *all* physical activities from the training set generated in the previous step. We used the k-means algorithm, where k is the number of clusters in which the physical activities are grouped in, i.e, the number of primitives will be k. Once the clustering is finished each *primitive* corresponds to the centroid of each cluster.

3.4.4 Histogram Generation

The histogram represents the distribution of *primitives* for each *complex activity*, Procedure 2 shows how to construct it.

Procedure 2 Generate histogram of primitives

1: initialize array $H[1..nc]$ to zeros. nc is the number of centroids
2: $count \leftarrow 0$
3: **while** complex activity has more physical activities **do**
4: $\quad id \leftarrow \text{NearestCentroid}(physicalActivity)$
5: $\quad H[id] \leftarrow H[id] + 1$
6: $\quad count \leftarrow count + 1$
7: **end while**
8: divide each element of H by $count$ ▷ Normalize H

The procedure NearestCentroid(*physicalActivity*) uses the Euclidean distance to find the closest centroid to *physicalActivity* and returns its *id*. As shown in Figure 3.9, once we have the normalized histograms of primitives, instead of using them as the final instances to train a classifier we first extracted a set of features from them and applied a threshold in order to reduce noise. Each histogram corresponds to one feature vector with the following features:

1. Number of primitives with probability > 0 normalized by the number of centroids. (This is a numeric feature)

2. The *id's* of the k_{th} primitives with highest probability. We set $k = \lceil \#primitives/30 \rceil * 3$. (This is a categorical feature)

3. For each *primitive* we add a binary feature f_i (Eq. 3.8). This is done to reduce some noise due to isolated primitives that may appear in the histogram. For the experiments we set the threshold to 0.005. (We treat this as a numeric feature)

$$f_i = \begin{cases} 1 & \text{if } H[i] > threshold, \\ 0 & otherwise. \end{cases} \tag{3.8}$$

3.4.5 Training and Classification

Once the feature vectors are extracted from the histograms, they can be used to train and test a classifier. The classification is performed using the total information from each of the instances, i.e, without using fixed length windows.

3.5 Summary

In this Chapter the methodology for recognizing different types of human activities was presented. First, a sensing platform was chosen. In this case it was a smartphone with Android operating system which has a triaxial accelerometer. Then the process of physical activity recognition was presented which consists of extracting simple statistical features from fixed size time windows. Once the feature vectors are extracted, they can be used to train and test a classifier. Next, Wi-Fi access points were used to contextualize the physical activities by adding location information. Finally complex activity recognition process was presented using a bag of features approach. Complex activities are modeled as a distribution of physical activities and then a histogram is created from which feature vectors are extracted. The feature vectors are then used to train and test a classifier.

Chapter 4

Experiments and Results

In this chapter the experiments and results for the following phases are presented:

1. Physical activity recognition.
2. Contextualizing physical activities using information gathered from Wi-Fi access points.
3. Complex activity recognition.

For each phase, the experiment setup is described which includes information about the amount of the collected data, how it was collected, the type of activities, etc. Then the results of the recognition are presented which may include a comparison between several methods. Finally, it concludes with an overall summary about the three phases.

4.1 Physical activity recognition

In this section, we present how the experiments for physical activity recognition were carried out, i.e., the sampling rate, cellphone location, collected instances, etc. Then the results of three classifiers (KNN, C4.5 and Naive Bayes) are shown.

4.1.1 Experiments Description

For the data collection process a smartphone was sued as described in Chapter 3 (Sensing Platform). The sampling rate was set to 40Hz (one sample every 25 milliseconds). This sensor is capable of sensing at approximately 90Hz but it was decided to set it at a lower rate to save battery. The cellphone was placed in the user's belt in a vertical position. The activities to be recognized were five: *walk, run, climbing-up stairs, climbing-down stairs and resting.*

The first experiment consisted of collecting training data for about 3 to 4 minutes for each of the activities. In total, 480 training instances were collected among the five

activities. The test set consisted of collecting data for 1 minute for each activity which is equivalent to 150 instances among the five activities.

Then a second experiment was performed which consisted of adding one more minute of training, i.e, training the algorithms with data collected during 4 to 5 minutes. From this data 630 training instances were generated. The test set consisted of 1 minute of data.

4.1.2 Results

The overall accuracy(correctly classified instances/total number of instances) for the first experiment using KNN algorithm was 82.66%. Table 4.1 shows the confusion matrix. In this table one can see that from the 30 instances of *Walk*, 20 were correctly classified and the other 10 were misclassified as *Up-stairs*. The diagonal of this table shows the number of instances that were classified correctly. The overall accuracy using C4.5 and Naive Bayes was 86.66% and 90% respectively. Tables 4.2 and 4.3 show their confusion matrix. The greatest source of error was between *Down-stairs* and *Walk*. With KNN 11 instances of type *Down-stairs* were misclassified as *Walk* and 4 of them as *Up-stairs*. *Resting* and *Run* instances were all correctly classified by the three algorithms. Naive Bayes was able to classify all *Up-staris* instances correctly.

For the second experiment (adding one more minute for training) the overall accuracy with KNN was 89.33%. Table 4.4 shows the results. One can see that with an extra minute for training the accuracy of *Down-stairs* increased. The total accuracy with C4.5 and Naive Bayes was 87.33% and 93.33% respectively. Tables 4.5 and 4.6 show their confusion matrix. Table 4.7 shows the accuracy for each activity and each of the 3 methods. The accuracies of *Walk* and *Down-stairs* were the only ones that changed when adding the extra minute. KNN was the most benefited with the extra minute. KNN had an increased in its accuracy of up to 66% (for *Down-staris*) and 15% on average. C4.5 had an average increase of 1% and Naive Bayes a 6%.

Based on these results, real time recognition seems feasible with simple algorithms. KNN and Naive Bayes got the best results. The advantage of KNN is that it does not require a model to be created. This allows the classification of instances where some of the attributes may not be known until running time, e.g., when scanning for nearby Wi-Fi access points.

Table 4.1: Experiment 1: Confusion matrix using KNN

Activity	Classified as				
	Walk	Run	Up-stairs	Down-stairs	Resting
Walk	**20**	0	10	0	0
Run	0	**30**	0	0	0
Up-stairs	1	0	**29**	0	0
Down-stairs	11	0	4	**15**	0
Resting	0	0	0	0	**30**

Table 4.2: Experiment 1: Confusion matrix using C4.5

Activity	Classified as				
	Walk	Run	Up-stairs	Down-stairs	Resting
Walk	**21**	0	7	2	0
Run	0	**30**	0	0	0
Up-stairs	4	0	**26**	0	0
Down-stairs	5	0	2	**23**	0
Resting	0	0	0	0	**30**

Table 4.3: Experiment 1: Confusion matrix using Naive Bayes

Activity	Classified as				
	Walk	Run	Up-stairs	Down-stairs	Resting
Walk	**25**	0	5	0	0
Run	0	**30**	0	0	0
Up-stairs	0	0	**30**	0	0
Down-stairs	9	0	1	**20**	0
Resting	0	0	0	0	**30**

Table 4.4: Experiment 2: Confusion matrix using KNN

Activity	Classified as				
	Walk	Run	Up-stairs	Down-stairs	Resting
Walk	**20**	0	8	2	0
Run	0	**30**	0	0	0
Up-stairs	1	0	**29**	0	0
Down-stairs	3	0	2	**25**	0
Resting	0	0	0	0	**30**

Table 4.5: Experiment 2: Confusion matrix using C4.5

Activity	Classified as				
	Walk	Run	Up-stairs	Down-stairs	Resting
Walk	**21**	0	3	6	0
Run	0	**30**	0	0	0
Up-stairs	3	0	**26**	1	0
Down-stairs	4	0	2	**24**	0
Resting	0	0	0	0	**30**

Table 4.6: Experiment 2: Confusion matrix using Naive Bayes

Activity	Classified as				
	Walk	Run	Up-stairs	Down-stairs	Resting
Walk	**23**	0	3	4	0
Run	0	**30**	0	0	0
Up-stairs	0	0	**30**	0	0
Down-stairs	2	0	1	**27**	0
Resting	0	0	0	0	**30**

Table 4.7: Accuracy for each activity in both experiments.

Activity/Accuracy	First Experiment			Second Experiment		
	KNN	C4.5	NB	KNN	C4.5	NB
Walk	66.66%	70%	83.33%	66.66%	70%	76.66%
Run	100%	100%	100%	100%	100%	100%
Up-stairs	96.66%	86.66%	100%	96.66%	86.66%	100%
Down-stairs	50%	76.66%	66.66%	83.33%	80%	90%
Resting	100%	100%	100%	100%	100%	100%
Overall accuracy	82.66%	86.66%	90%	89.33%	87.33%	93.33%

4.2 Contextualizing physical activities using information gathered from Wi-Fi access points

In this section, we present how the experiments for contextualizing physical activities were carried out. Then the results of the classification of 8 activities performed in an academic environment are shown.

4.2.1 Experiments Description

The data collection consisted of 8 activities: 1)reading in bedroom A, 2)watching television, 3)reading in bedroom B, 4)sitting in the lobby, 5)reading in the library (first floor), 6)looking for a book in the library (first floor), 7)reading in the library (second floor), 8)looking for a book in the library (second floor). Activities 1-4 were performed in an apartment building while activities 5-8 were performed in a library. The data collection process was performed by two participants under supervision. The test set was collected independently in a different day from the training set.

For this experiment we collected a total of 741 instances for the training set and 243 instances for the test set which correspond to approximately 170 minutes of data. Tables 2 and 3 show the number of instances per activity and the average number of detected Access Points. The average number of detected Access Points is important because in [37] they observed that the number of received Access Points strongly affects accuracy of proximity classification. They improved their results by performing three scans and feeding the algorithm with the scan that has the highest number of detected Access Points.

Table 4.8: Number of training instances and their respective average number of detected Access Points

Contextualized Activity	**Number of training instances**	**Average number of detected Access Points**
1) Reading in bedroom A	102	2.2
2) Watching television	104	4.0
3) Reading in bedroom B	68	1.5
4) Sitting in the lobby	91	5.1
5) Reading in the library (first floor)	107	4.2
6) Looking for a book in library (first floor)	78	8.5
7) Reading in the library (second floor)	103	4.9
8) Looking for a book in library (second floor)	88	10.8

Table 4.9: Number of test instances and their respective average number of detected Access Points

Contextualized Activity	**Number of test instances**	**Average number of detected Access Points**
1) Reading in bedroom A	29	2.3
2) Watching television	28	3.7
3) Reading in bedroom B	27	1.5
4) Sitting in the lobby	32	4.2
5) Reading in the library (first floor)	36	6.0
6) Looking for a book in library (first floor)	30	7.0
7) Reading in the library (second floor)	31	9.9
8) Looking for a book in library (second floor)	30	8.4

4.2.2 Results

The overall classification accuracy (number of correct classifications/number of total instances) using holdout validation was 89.7%. Table 4.10 shows the confusion matrix. From this table we can see, e.g., that 1 of the 28 instances of watching television activity was misclassified as reading in bedroom B and the remaining 27 instances were correctly classified. The main diagonal shows the number of correctly classified instances. 10-fold cross validation was also performed over the entire data set (984 instances = training + test instances) and the resulting accuracy was 90.3%.

It can be seen that there is a relation between the average number of detected Access Points per activity (Tables 4.8 and 4.9) and the accuracy of the classification. For example, reading in bedroom B has the lowest average of detected Access Points (just 1), and in the confusion matrix it can be seen that 8 of its instances were misclassified. In contrast, the activities with high number of detected Access Points had fewer misclassifications. Then, based on these results it appears that in order to achieve good classification accuracies for this experiment's configuration, the activity must have at least an average of 3 detected Access Points.

The major source of error was between activities that are physically close to each other. Given that the library is very far from the departments building there were no misclassifications between these activities.

Table 4.10: Confusion matrix

Activity	**Classified as**							
	Reading in bedroom A	Watching television	Reading in bedroom B	Sitting in the lobby	Reading in library 1f	Looking for book in library 1f	Reading in library 2f	Looking for book in library 2f
Reading in bedroom A	**22**	5	0	2	0	0	0	0
Watching television	0	**27**	1	0	0	0	0	0
Reading in bedroom B	5	3	**19**	0	0	0	0	0
Sitting in the lobby	0	0	0	**32**	0	0	0	0
Reading in library 1f	0	0	0	0	**32**	0	4	0
Looking for book in 1f	0	0	0	0	0	**26**	0	4
Reading in library 2f	0	0	0	0	0	0	**31**	0
Looking for book in 2f	0	0	0	0	0	1	0	**29**

4.3 Complex activity recognition

In this section, we present how the experiments for complex activity recognition were carried out, i.e., number of collected instances, how the validation was done, etc. Then the results for directly classifying the histograms and the results after feature extraction over the histograms are shown.

4.3.1 Experiments Description

For the experiment, 41 hours of data were collected consisting of 13 training instances and 67 for the test set. Table 4.11 shows the number of instances per activity. The duration of each activity varies from a couple of minutes to several hours. The number of training instances was chosen so that each type of instance encompasses the most common physical activities that may be present in it.

Table 4.11: Distribution of instances

Activity	No. Training Instances	No. Test Instances
1)Commuting	4	14
2)Working	3	19
3)At home	3	16
4)Shopping	1	13
5)Exercising	2	5

The training instances were just used to generate the *primitives* and the classification accuracy was evaluated using 10-fold cross validation over the entire data set (train + test set). First, experiments over the histogram of primitives (before feature extraction and noise reduction) were performed. Then, experiments after feature extraction and noise reduction over the histogram were done. We evaluated the accuracy for different values of k which represent the number of centroids to use, i.e, the number of *primitives*. We also set the physical activity window length to 2, 4 and 10 seconds respectively.

4.3.2 Results

Figures 4.1, 4.2 and 4.3 show the classification results before and after feature extraction for different number of primitives and different physical activity window lengths. The K-Nearest Neighbors method was used for the classification with a $K = 1$. The instances include the information of the entire activity, i.e, no time windows were used.

The maximum achieved accuracy for a 2 second window length was 92.5% with a k of 25 and 40. For a window length of 4 seconds the maximum accuracy was 92.5% for

$k = 14$ and with a window length of 10 seconds the maximum accuracy was 91.2% for $k = 45$ and $k = 55$. Table 4.12 shows the average accuracies for k between 15 and 100; from this Table it seems that using a window length of 2 seconds gives better results on average.

Figure 4.4 shows how an *exercising* instance is classified over time, i.e, as more primitives are added. At first, it is correctly classified since the closest instance is of type *exercising*. Then, suddenly the distance to the correct class increases and the distance to *commuting* becomes the shortest. After some time it is then incorrectly classified as *shopping* and at the end it is again correctly classified as *exercising*.

Table 4.12: Average accuracies for k between 15 and 100 for window lengths of 2, 4 and 10 seconds

	2 seconds	**4 seconds**	**10 seconds**
Before Feature Extraction	85.39%	83.59%	82.25%
After Feature Extraction	87.61%	87.12%	86.42%

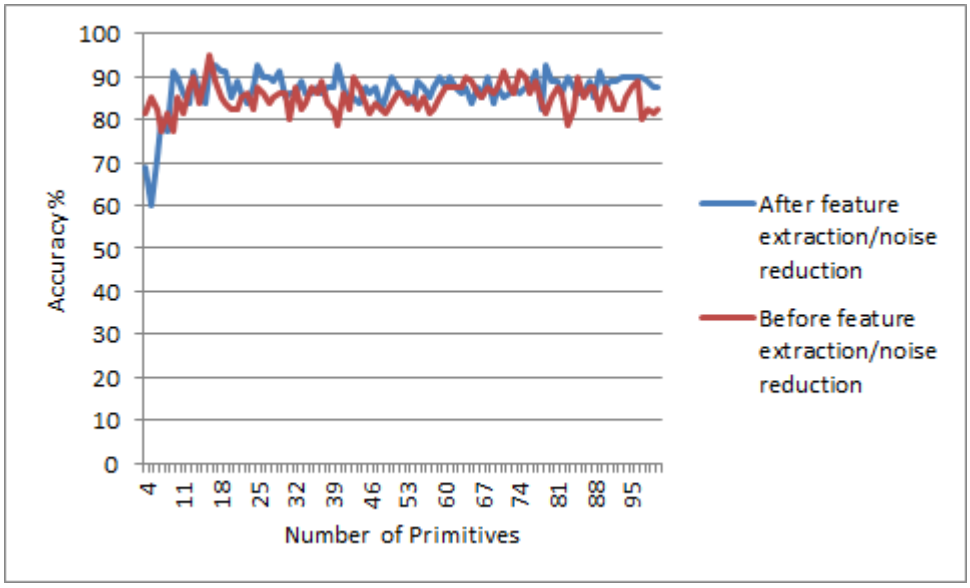

Figure 4.1: Accuracies for a window length of 2 seconds

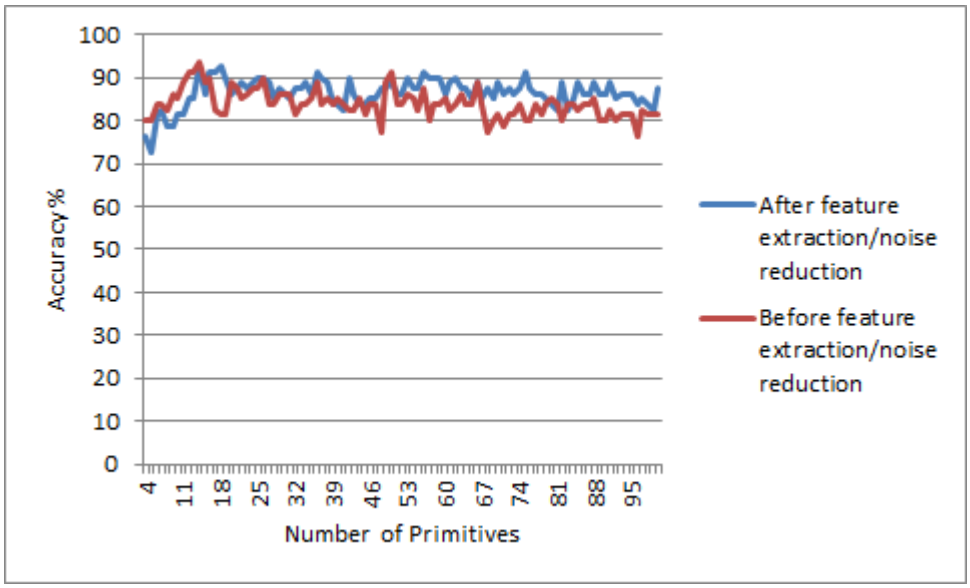

Figure 4.2: Accuracies for a window length of 4 seconds

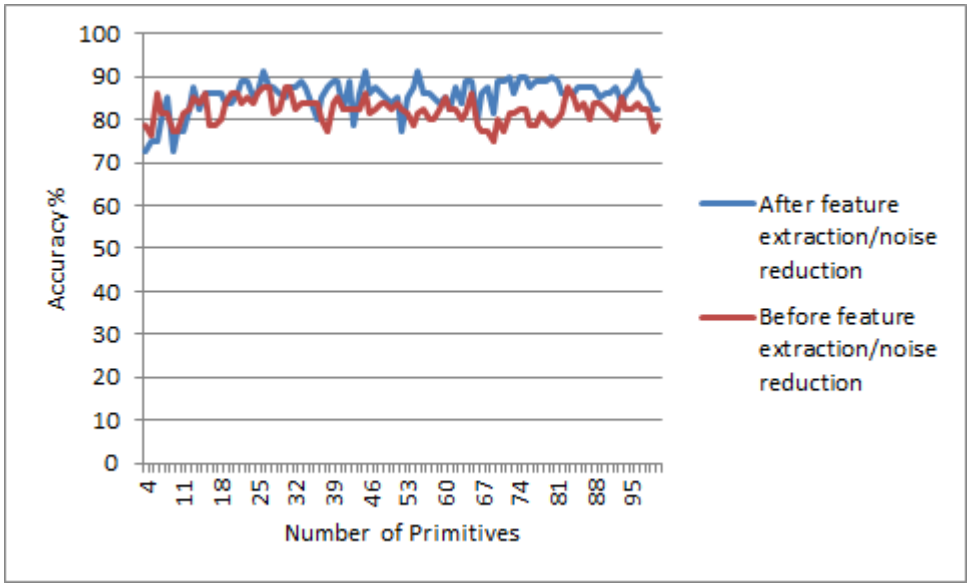

Figure 4.3: Accuracies for a window length of 10 seconds

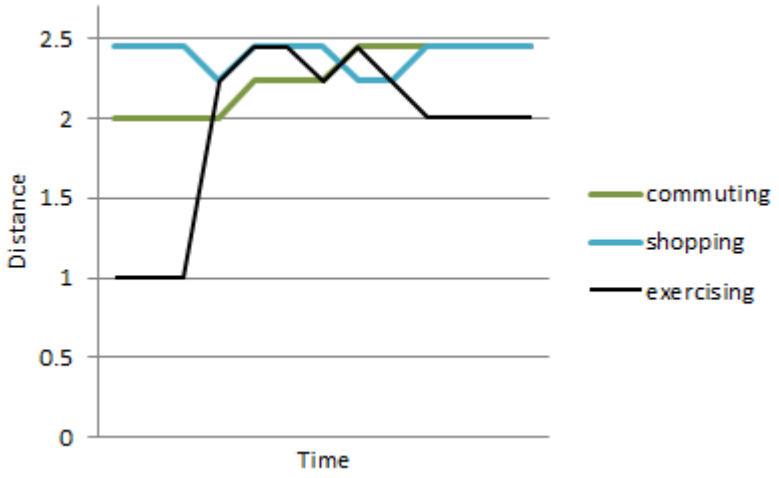

Figure 4.4: Classification distances for exercising as more primitives are added over time

4.4 Summary

In this chapter the experiments setup and results were presented for each of the three phases of this work. The following is a summary of the final results:

- The results showed that physical activity recognition is feasible to be performed in real time using base level classifiers. In this case Naive Bayes had the best results.
- The results for physical activity recognition showed that adding one more minute of training led to a significant increase of the overall accuracy.
- When adding information from Wi-Fi Access Points it was possible to contextualize the physical activities in order to have a better understanding of the user's state.
- It was possible to discriminate between two activities even when they are very close to each other.
- The results for complex activity recognition showed that the Bag-Of-Features approach can be used to model high level activities that are composed of simpler ones.
- It was shown that extracting features from the histograms helped to increase the accuracy.
- The achieved recognition accuracies for complex activities were good when using the information from the entire activity. However more research needs to be done in order to come up with a working implementation in real time.

Chapter 5

Conclusions

In this thesis, we presented how sensors that are commonly available in smartphones can be used to recognize the *state* of a person. By *state* we mean the physical activity being performed by the user. This work was divided into three phases: *1) Physical activity recognition.* This consists of recognizing activities that do not depend on the context and that last just a few seconds. Examples of these activities are: walking, resting, running, climbing-up stairs, etc. *2) Contextualizing physical activities.* This involves adding information gathered from nearby Wi-Fi Access Points to have a better understanding of the user's situation. For example, if *walking* is recognized as a physical activity and using Wi-Fi signals we know that the user is in a library we may infer that he/she is looking for a book. *3) Complex activities.* These activities consist of a collection of physical activities and they can last from some minutes to several hours. Examples of these activities are: shopping, exercising, working, etc.

It was shown how the recognition task can be stated as a classification problem. In Chapter 2, we presented an introduction to basic Machine Learning methods which are the basis of this work and made a survey of the state of the art in human activity recognition using wearable sensors. In Chapter 3, the methodology for each of the three phases was presented which included the sensing platform, the data collection process, training and classification. In Chapter 4, experiments for each of the three phases were described and results were presented.

5.1 Conclusions

From the experiments results we conclude that the use of smartphones for activity recognition in real time is feasible and good accuracies can be achieved. The following is a summary of the final results:

- The results showed that physical activity recognition is feasible to be performed in real time using base level classifiers. In this case Naive Bayes had the best results with an accuracy of 93.33%.

- We concluded that simple statistical features can be used for the recognition which yielded good results.
- When adding information from Wi-Fi Access Points it was possible to contextualize the physical activities in order to have a better understanding of the user's state.
- It was possible to discriminate between two activities even when they were very close to each other by using Wi-Fi information.
- The results from complex activity recognition showed that the Bag-Of-Features approach can be used to model high level activities that are composed of simpler ones.
- It was shown that extracting features from the histograms helped to increase the accuracy.
- The achieved recognition accuracies for complex activities were good when using the information from the entire activity. However more research needs to be done in order to come up with a working implementation in real time.

5.2 Contributions

The following is a list of contributions of the present work:

- We showed how physical activity recognition can be performed using smartphones. The main difference with other works is that just one sensor was used (trixaial accelerometer) which is available in most smartphones and a working prototype was actually implemented on Android which is capable of detecting the activities in real time.
- Using information from Wi-Fi Access Points we were able to contextualize the physical activities. A real time prototype was implemented on an Android device which is capable of detecting activities even when they are physically very close to each other.
- Complex activities were modeled as a collection of physical activities represented as a histogram. We proposed a set of features that can be extracted from the histogram. This additional step helped to increase the recognition accuracy.

5.3 Future Work

Some of the future directions of the present work are listed below:

- Some high level activities may not have sense if seen from a single user perspective. For example, in a meeting a single person may be talking and the others could be listening. If we focus on one person it may seem that he/she is not involved in any activity but if we take the context and the other users into account we may conclude that he/she is paying attention to the speaker, so we plan to include more sensors like Bluetooth to detect activities that involve interactions between several users. The interactions could be represented with a probabilistic graphical model such as Bayesian networks. The time of the day and proximity to other users could also be used to add more information from the context.

- Devise a way to implement complex activity recognition in real time. The challenge is to decide how much information from the past to use and to detect activity transitions. One way would be to represent primitives as nodes and transitions between primitives as edges. Then, use social network analysis algorithms over the graph to see how it evolves over time and use this to devise a metric that indicates when a transition is made.

- To extract patterns from the activities performed during several days and detect abnormal behavior. This has applications in areas like health care and security. One approach would be to model the activities as a graph and use subgraph matching algorithms to extract meaningful patterns.

Bibliography

[1] Diane J. Cook, Juan C. Augusto, and Vikramaditya R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277 – 298, 2009.

[2] Emile Aarts and Reiner Wichert. Ambient intelligence. In Hans-Jörg Bullinger, editor, *Technology Guide*, pages 244–249. Springer Berlin Heidelberg, 2009.

[3] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5:4–7, 2001. 10.1007/s007790170019.

[4] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3*, IAAI'05, pages 1541–1546. AAAI Press, 2005.

[5] Tomas Brezmes, Juan-Luis Gorricho, and Josep Cotrina. Activity recognition from accelerometer data on a mobile phone. In Sigeru Omatu, Miguel Rocha, José Bravo, Florentino Fernández, Emilio Corchado, Andrés Bustillo, and Juan Corchado, editors, *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, volume 5518 of *Lecture Notes in Computer Science*, pages 796–799. Springer Berlin / Heidelberg, 2009.

[6] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, March 2011.

[7] MP Lawton and EM Brody. Instrumental activities of daily living scale (iadl).

[8] H. Fujiyoshi and A.J. Lipton. Realtime human motion analysis by image skeletonization. In *Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on*, pages 15 –21, oct 1998.

[9] Pedro Canotilho Ribeiro and José Santos-victor. Human activity recognition from video: modeling, feature selection and classification architecture. In *International Workshop on Human Activity Recognition and Modeling (HAREM)*, 2005.

[10] Neil Robertson and Ian Reid. A general method for human activity recognition in video. *Computer Vision and Image Understanding*, 104(2):232 – 248, 2006.

[11] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human activity detection from rgbd images. *CoRR*, abs/1107.0169, 2011.

[12] Seon-Woo Lee and K. Mase. Activity and location recognition using wearable sensors. *Pervasive Computing, IEEE*, 1(3):24 – 32, 2002.

[13] Tao Gu, Zhanqing Wu, Xianping Tao, Hung Keng Pung, and Jian Lu. epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1 –9, march 2009.

[14] H. Nakashima, H. Aghajan, and J.C. Augusto. *Handbook of Ambient Intelligence and Smart Environments*. Springer, 2009.

[15] Emile H.L. Aarts and José Luis Encarnação. *True visions: The Emergence of Ambient Intelligence*. Springer, 2006.

[16] I. A. Group. Scenarios for ambient intelligence in 2010. 2011.

[17] Augusto J.C. and P. Mccullagh. Ambient intelligence: Concepts and applications. *Computer Science and Information Systems*, 4(1):1–28, 2007.

[18] I. Kononenko and M. Kukar. *Machine Learning and Data Mining*. Horwood Publishing, 2007.

[19] T. Segaran. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. Oreilly Series. O'Reilly Media, 2007.

[20] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2010.

[21] I.H. Witten, E. Frank, and M.A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011.

[22] J. R. Quinlan. Improved Use of Continuous Attributes in C4.5. *eprint arXiv:cs/9603103*, February 1996.

[23] D. Randall Wilson and Tony R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38:257–286, 2000. 10.1023/A:1007626913721.

[24] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 1st edition, September 2000.

[25] Eunju Kim, Sumi Helal, and D. Cook. Human activity recognition and pattern discovery. *Pervasive Computing, IEEE*, 9(1):48 –53, jan.-march 2010.

[26] Andrea Mannini and Angelo Maria Sabatini. Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2):1154–1175, 2010.

[27] DM Karantonis, MR Narayanan, M Mathie, NH Lovell, and BG Celler. Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE*, 10(1):156 – 167, 2006.

[28] Mi Zhang and Alexander A. Sawchuk. Motion primitive-based human activity recognition using a bag-of-features approach. In *ACM SIGHIT International Health Informatics Symposium (IHI)*, pages 631–640, Miami, Florida, USA, January 2012.

[29] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, pages 10–19, New York, NY, USA, 2008. ACM.

[30] David M. Blei. Surveying a suite of algorithms that offer a solution to managing large document archives. *Commun. ACM*, 55(4), 2012.

[31] Tâm Huynh, Ulf Blanke, and Bernt Schiele. Scalable recognition of daily activities with wearable sensors. In Jeffrey Hightower, Bernt Schiele, and Thomas Strang, editors, *Location- and Context-Awareness*, volume 4718 of *Lecture Notes in Computer Science*, pages 50–67. Springer Berlin / Heidelberg, 2007.

[32] Hao Tian, Pang Lei, Li Xingjuan, and Xing Shusong. Wearable activity recognition for automatic microblog updates. In *Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on*, pages 1720 –1723, july 2009.

[33] Stefan Dernbach, Barnan Das, Narayanan C. Krishnan, Brian L. Thomas, and Diane J. Cook. Simple and complex activity recognition through smart phones. In *Intelligent Environments (IE), 2012 8th International Conference on*, pages 214 –221, june 2012.

[34] John Krumm and Eric Horvitz. Locadio: Inferring motion and location from wi-fi signal strengths. In *in First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous*, 2004.

[35] J. Correa, E. Katz, P. Collins, and M. Griss. Room-level wifi locationntracking. *Carnegie Mellon Silicon Valley, CyLab Mobility Research Center technical report MRC-TR-2008-02*, 2008.

[36] G.V. Zdruba, M. Huber, F.A. Karnangar, and I. Chlarntac. Monte carlo sampling based in-home location tracking with minimal rf infrastructure requirements. In *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, volume 6, pages 3624 – 3629 Vol.6, nov.-3 dec. 2004.

[37] Alessandro Carlotto, Matteo Parodi, Carlo Bonamico, Fabio Lavagetto, and Massimo Valla. Proximity classification for mobile devices using wi-fi environment similarity. In *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, MELT '08, pages 43–48, New York, NY, USA, 2008. ACM.