FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Activity recognition from smartphone sensing data

Alexandre de Oliveira Lopes

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Doutor João Pedro Carvalho Leal Mendes Moreira, Prof. Auxiliar da FEUP

Co-orientador: Doutor João Manuel Portela da Gama, Prof. Associado com Agregação, FEP, UP

July of 2012

Activity recognition from smartphone sensing data

Alexandre de Oliveira Lopes

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente de Júri: Doutor Henrique Daniel de Avelar Lopes Cardoso, Prof. Auxiliar da FEUP

Vogal Externo: Doutor Paulo Jorge Freitas Oliveira Novais, Prof. Auxiliar da Universidade do Minho

Orientador: Doutor João Pedro Carvalho Leal Mendes Moreira, Prof. Auxiliar da FEUP

_____

12th of July 2012

Resumo

Reconhecer atividades humanas com sensores junto ao corpo tem-se tornado bastante importante ao longo dos anos, com o objetivo de criar/melhorar sistemas de apoio à 3ª idade, fitness e saúde e ajudar aqueles com deficiências cognitivas. Esta tarefa de reconhecimento de atividades que ocorrem num determinado momento com um indivíduo tem o nome de Activity Recognition.

Com o desenvolvimento de novas tecnologias, como os smartphones, foi possível ultrapassar a barreira do utilizador ter de usar vários sensores junto ao corpo e passar a usar apenas um, no bolso da frente das calças. Ser possível dar uma nova solução para este problema é motivante, além disso, trabalhar com Android que é a tecnologia líder no mercado e ajudar a melhorar a experiência do utilizador com o telefone dão um grande alento para a conclusão deste projeto.

Tendo os smartphones um acelerómetro tri-axial embutido é possível criar aplicações que são capazes de prever as atividades do utilizador com grande exatidão. Esta tese vai de encontro a uma nova solução, que é criar uma aplicação Android para o problema de reconhecer atividades tratando-o como um problema de classificação. Para chegar à solução foi necessário o estudo de trabalhos realizados previamente. O objetivo foi perceber quais as atividades humanas consideradas comuns e que abordagens podiam ser feitas ao problema de classificação, como supervised ou semi-supervised learning, e quais os classificadores mais usados e os melhores métodos para os comparar.

Com este estudo decidiu-se avançar para uma aplicação Android que explorasse as abordagens com supervised e semi-supervised learning com one-step classification (classificar os dados em Standing Idle, Sitting, Running e Walking) e hierarchical classification (tendo esta abordagem duas classificações, primeiro em Dynamic e Static activities e depois dentro deDynamic em Running e Walking e dentro de Static em Standing Idle e Sitting).

Usa-se uma abordagem supervised e semi-supervised learning, em que no primeiro caso se cria um modelo que se mantém estático ao longo do tempo e no segundo caso alguns dos dados classificados pelo modelo são adicionados ao ficheiro de treino do classificador com o intuito de criar um novo modelo e ter um ficheiro actual e que reflecte o comportamento do utilizador. Os principais componentes da aplicação e a sua interação e a sua arquitetura básica são explicadas para uma melhor compreensão de todo o sistema. Para comparar as abordagens e

a performance dos classificadores a sua exatidão na classificação foi escolhida. Curvas precision/recall também foram criadas para perceber e avaliar os sistemas de recolha de informação dos modelos. Como se trabalhou numa aplicação móvel o uso de memória e bateria por parte da aplicação e tempo gasto na classificação foram tidos em conta. Para testar a viabilidade da aplicação estas questões foram monitorizadas. Os classificadores escolhidos foram Naive Bayes e Hoeffding trees.

As principais conclusões deste estudo são: 1) classificação hierárquica tem uma melhor performance que a classificação one-step; 2) o melhor par que se pode criar na classificação hierárquica é o uso de Naive Bayes na primeira classificação e Hoeffding trees na segunda; 3) Semi-supervised learning é no global melhor que supervised classification; 4) Naive Bayes consome menos bateria que Hoeffding trees.

Abstract

Recognizing human activities with sensors next to the body has become more important over the years, aiming to create or improve systems in elder care support, health/fitness monitoring, and assisting those with cognitive disorders. This task of recognizing activities taking place at a certain moment when considering only one individual user is called *Activity Recognition.*

With the development of new technologies, like smartphones, it was possible to overcome the barrier of the person having to use multiple body worn sensors and passing to use only one, in his trousers' front pocket. Being able to give a new solution for this problem is a huge motivation, besides having the pleasure of working with Android technology that is leading the market and helping the user experience with is phone.

Having the smartphones a triaxial accelerometer built in it is possible to create applications that are capable of recognizing the activities of the user with great accuracy.

This thesis aims to meet this new solution, creating an Android application, for the problem of recognizing the activities performed by the user and treating it as a classification problem. To embark into a path that leads to the solution it was necessary to study previous works in order to trace this path. The main objective was to understand what were considered the common human activities and what approaches could be taken when dealing with this classification problem, like supervised or semi-supervised learning. What were the most usual classifiers, what their differences were and how could we compare them.

With this study we decided to built an Android application that explore supervised and semi-supervised learning with both one-step classification (classifying the data in Standing Idle, Sitting, Running and Walking) and hierarchical classification (having this approach two classifications, first in Dynamic and Static activities and then inside Dynamic into Running and Walking and inside Static into Standing Idle and Sitting). On supervised learning a model is created and it stays static along the time. On semi-supervised learning some instances labeled by the model are added to the training file in order to create a new model and have a training file up to date and with activities from the current user. The main components of the application, how they interact its basic architecture are presented. In order to compare the classifiers' performance their accuracy was chosen. Curves of precision/recall were also created to understand and evaluate the models' information retrieval system. Since we were working on a

mobile application the memory and battery usage and time spent on the classification were also an issue. To check the feasibility of the application these issues had to be monitored. The classifiers used in the experiments were Naive Bayes and Hoeffding trees.

The main conclusions from this study are: 1) hierarchical classification has better performance than one-step classification; 2) the best mix for the hierarchical classification is using Naive Bayes in the first classification and Hoeffding trees in the second one; 3) Semi-supervised learning is globally better than supervised classification; 4) Naive Bayes consumes less battery than Hoeffding trees.

Acknowledgements

Writing the acknowledgments means that our work is pratically done and that we are proud of it.

It has been a long journey that I could not finish without the help of my supervisors. I thank them, João Moreira and João Gama, for the patiente and share of knowledge.

Thanks to my family, especially to my parents and grandparents for all the support and motivation that they gave me in order to do a good work.

Thanks to my friends for keeping me in a good mood and motivating me.

Thanks to the support of the project Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIA/098355/2008)

Alexandre Lopes

Contents

1- Introduction 1
1.1 Problem 2
1.2 Project 4
1.3 Motivation 5
1.4 Dissertation Structure 5

2- Related Work 7
2.1 Single sensor 7
2.2 Data Mining 8
2.3 Supervised and Semi-supervised learning 9
2.4 Classifiers 10
2.4.1 Markov 10
2.4.2 Base-level e meta-level classifiers 12
2.4.3 Naïve Bayes 14
2.4.4 Decision tree – J48 16
2.4.5 Hoeffding Tree 19
2.5 Hierarchical classification 22
2.5.1 SVM 24
2.6 Battery 28
2.7 Data Extraction 29
2.8 Sliding Windows 30
2.9 Duty cycles 31
2.10 Existing prototypes 31

3- Software 33
3.1 Moa - Massive Online Analysis 33
3.2 Android 34
3.3 Power Tutor 35

4- Prototype 36
4.1 Techniques used 36
4.2 Experimental setup 36

4.3 Architecture 37
4.4 Experiments and Results 41

5- Conclusions and Future Work 50

References 52

Appendix A 55
Precision/Recall graphs 55

List of Figures

Figure 1 Axes of motion relative to user 3
Figure 2 Location of the sensors on the user's body 7
Figure 3 Acceleration signals from five biaxial accelerometers for walking, running, and tooth brushing 8
Figure 4 Summary of classifier results (mean ± standard deviation) using user-specific training and leave-one-subject-out training where both training data sets are equivalent to five laboratory data sessions 8
Figure 5 Overall precision/recall for the static and HMM classifiers trained/tested on all locations and on a single location 11
Figure 6 Accuracy of classifiers for four different settings 13
Figure 7 Performance correlation for IDD and non-IDD data 14
Figure 8 Bayes distribution 15
Figure 9 Decisions at each node to get a final classification 16
Figure 10 Accuracies of activity recognition 17
Figure 11 Acceleration plots for Walking and Jogging 18
Figure 12 Confusion matrix for J48 Model (Stairs Combined) 18
Figure 13 Accuracy as a function of the number of training examples 21
Figure 14 Tree size as a function of the number of training examples 21
Figure 15 Accuracy as a function of the noise level 22
Figure 16 Classification framework, which is a structured as binary tree 23
Figure 17 Detection falls. Tests in each node with no ambiguity 23
Figure 18 Flowchart describing the classification process 24
Figure 19 Acceleration signal with period of inactivity 25
Figure 20 Distinguishing short time motion and noise of gentle motion 25
Figure 21 Classification results using only one classifier 26
Figure 22 3D-accleleration signals measured above 6 activities 26
Figure 23 Classification results using two SVM classifiers 27
Figure 24 Motionless confusion matrix using 3MotionlessM 27
Figure 25 Motion confusion matrix using 3MotionM 27

Figure 26 Sensor duty cycles, device computation time and sensor energy cost per sample 28
Figure 27 Power usage of sensors 28
Figure 28 X-axis readings for different activities 29
Figure 29 Sensor duty cycles, device computation time and sensor energy cost per sample 31
Figure 30 DiaTrace application 32
Figure 31 Workflow in MOA 33
Figure 32 Android components' 33
Figure 33 Application components 37
Figure 34 Main functions on prototype behavior also one-step classification with supervised learning 39
Figure 35 Main functions on prototype behavior also one-step classification with supervised learning 40
Figure 36 Accuracy of one step classification using both supervised and semi-supervised learning 42
Figure 37 Final classifiers' accuracy using hierarchical supervised classification 43
Figure 38 Classifiers' accuracy on final step of hierarchical classification with a semi-supervised learning approach 43
Figure 39 Final classifiers' accuracy using hierarchical supervised classification with different classifiers for each step 44
Figure 40 Final classifiers' accuracy using hierarchical semi-supervised classification with different classifiers for each step 45
Figure 41 Precision/recall graph of Naïve Bayes used in one-step classification. Semi-supervised learning (left), supervised learning (right) 46
Figure 42 Precision/recall graph of Hoeffding Tree used in second classification (Hoeffding Tree also used in first classification) of the hierarchical approach (Running and Walking). Semi-supervised learning (left), supervised learning (right) 46
Figure 43 Precision/recall graph of Naïve Bayes used in second classification (Naïve Bayes also used in first classification) of the hierarchical approach (Running and Walking). Semi-supervised learning (left), supervised learning (right) 47
Figure 44 Power used by Hierarchical approach 48
Figure 45 Power used by One step approach 48

Listo f Tables

Table 1: Classifiers' accuracy ……..……41
Table 2: accuracy in the first level of the hierarchical approach……..…41
Table 3: Classifiers' accuracy in the second classification of the hierarchical approach……....41
Table 4: Classifiers' Accuracy for the walking activity using as test set data from a person without data on the training sets……...42

Abbreviations

MOA Massive Online Analysis

Chapter 1

Introduction

Studying and understanding the human body have always been an area of great interest. With the creation of sensors a new opportunity for further studying the body was created. The sensors could provide new data about the movements that could be used to recognize physical activities.

Since the 80's this field of recognizing activities from data collected from body worn sensors has been capturing the attention of more and more people of the computer science community due to its interdisciplinary with areas like medicine, human computer interaction or, even, sociology. Moreover it gives the possibility to create systems that can adapt to the users enabling the possibility of creating or improving systems in elder care support, health/fitness monitoring, and assisting those with cognitive disorders

The appearing of smartphones was a big breakthrough in this area because they had computation power to process in real time the data gathered by their sensors and they were not a strange object to the users, which was one of the major setbacks of the first systems that were multi-sensor.

The smartphone is more advanced than a normal mobile phone. The first smartphones were a hybrid of PDA and normal mobile. Now they combine multiple functions of several devices like GPS, media playback device, digital cameras. They also give to the user the possibility to be always connected to the internet.

This permits that data can be processed in real time or stored and then processed on the computer or simply being sent over the internet or bluetooth without bothering the user.

In physics, movement is the variation of the spatial position of an object over time.

The accelerometer operates in a simple way. It obeys to Newton's second law (the acceleration of an object as produced by a net force is directly proportional to the magnitude of

the net force, in the same direction as the net force, and inversely proportional to the mass of the object).

A simpler explanation in order to visualize the accelerometer behavior is to imagine it like a glass of water half full. If the glass is on a flat surface and it is pushed the water moves. The stronger we push the glass the more the water oscillates. What the accelerometer does is to measure these movements. It does it measuring the angle between the water when the glass is being pushed and when the glass was still, with this angle it can return the acceleration that was applied on the glass.

The research in accelerometer data is so important because it can be used for classifying activities that will help to develop programs that aim to aid in health care support or even to be used in a recreational way practicing sports, some of the objectives outlined in the beginning of the research in this field.

1.1 Problem

The introduction sets three main ideas: (1) the relevance of doing research form sensors' data for the improvement of humans' life; (2) the contribution given by smartphones for the reseach described in (1); and (3) the how and why this data is important for activity recognition.

Having these main ideas we can advance to understanding the problem and how these three things are connected.

Like it was stated studying the human body with sensors is not a new problem. The first works in this field used multiple sensors the collect as much data as possible. However the need to use several sensors and because of their size and the need of using cables to transfer the data to the computers was not attractive to the user.

This was a major setback because, due to these limitations, the tests made didn't reflect the normal behavior of the user. With these unappealing systems an effort had to be done in this field to make it viable. This can be done by using only the sensors available in smartphones. Like the accelerometer, for instance.

The smartphones had a big contribution for this problem of recognizing activities because of its computational power and sensors. In order to better understanding why the smartphones were such a good aid it is unavoidable to give a better explanation of the problem that we are dealing with.

This problem of trying to predict activities from accelerometer data has been treated as a classification problem. Like it will be shown almost every author had approach. The main difference between the existing approaches is the classification algorithm used.

Recognizing activities has the objective of recognizing actions of one or several agents by observing their actions and environmental conditions. With previous recorded data system creates a model that tries to infer from what activity new unlabelled data comes from, classifying it.

The classification problem can, also, be explained as a mathematical problem. A set of N training examples of the form (x; y) is given, where y is a discrete class label and **x** is a vector of d attributes, each of which may be symbolic or numeric. The goal is to produce from these examples a model $\hat{y}$=f(x) that will predict the classes y of future examples x with high accuracy. [1]

One of the biggest problems in the beginning, like it was mentioned, was the necessity of the user wearing several sensors on different parts of the body, which conditioned him and influenced the data collected making the results of the tests obsolete.

The smartphone tries to solve these problems of having a strange device next to the body and the need to have multiple sensors, in this case only one sensor that is built-in the smartphone is used. Like it will be stated later a single motion sensor placed in a determinate position on the body, given its good accuracy results, will be sufficient to give good results regarding the acceleration on the body. So using others sensors would just give marginal gains.

Being the smartphone a device that the user usually carries the data collected by the accelerometer will reflect how the user normally acts. So, no longer we have to work with data that does not reflect the normal behavior of the person being tested, tests will give truthful results.

The accelerometer measures the acceleration in three different dimensions (x, y, z) - Figure 1.

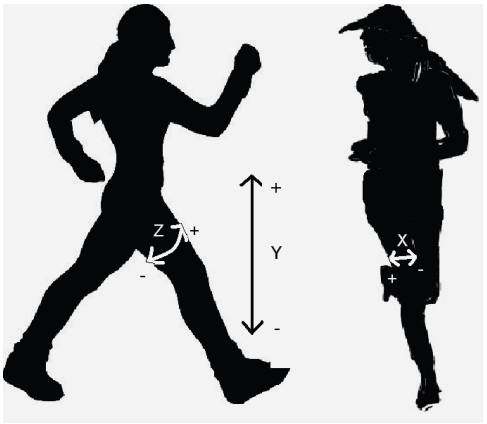


Figure 1 Axes of motion relative to user [2]

The accelerometer was built-in the smartphones to enhance gameplay and to rotate the screen according the phone position, but soon everybody saw the advantages of this sensor and which new applications could be created using it. In the case of this problem it made it possible to predict activities using the phone.

To treat this as a classification problem, a knowledge discovery system needs to be built. Knowledge discovery systems are constrained by three main limited resources: time, memory

and sample size. In traditional applications of machine learning and statistics, sample size tends to be the dominant limitation.

Since the work was developed on a smartphone, the memory space was limited, and the sample size had to be decided testing the accuracy of the system with different sizes, like it will be explained later.

Another specificity of the problem which is closely linked with the sample size was the need to work with data streams that have the problem of rate of the arrival of examples. When new examples arrive at a higher rate than they can be mined, the quantity of unused data grows without bounds as time progresses.

The problem becomes more complex when besides the speed the data has to be processed to do the classification in real-time without consuming too much memory. Additionally, the battery lifetimehas to be taken into account. The application created cannot have much impact on the user's smartphone performance and, at the same time, it has to collect and classify the data accurately.

Now it is possible to understand the link between the three main ideas of the introduction. The user is used to the smart phone, so he will act normally and carry the phone every day. The smartphone containing the accelerometer and the data collected by it can be used in the classification problem in order to recognize the activities of the user. The data collected will help studying the human body.

1.2 Project

The main objective of this prject is to give an answer to the problem described above by creating an Android application that can classify the user activities taking into account all the specific restrictions aforementioned.

Before explaining the decisions of which classifiers were used, how the data was recorded among other specificities of the system, it is important to understand what we wanted before building the application. So, in the end, we could judge if the main objective was reached.

Like it was said since the beginning we wanted an application that treated this problem as a classification problem. It had to work in real time and classify the four main human activities, sitting, standing idle, walking and running. The main idea was to have the accelerometer gathering unlabeled data and sending it to the application that was also, always, running.

The application would have a model to classify the data. All the new labeled data could be saved or just presented on the screen to ensure the application's good performance. The model could also be generated on the phone.

We wanted to do a semi-supervised learning so we decided new labels predicted with a given certainty woul be added to the training set. This approach is compared against supervised methods.

1.3 Motivation

The use of sensors has become mainstream, everybody carries at least one sensor on everyday life without realizing. Being the smartphones a device where the innovation in sensors is present, it is thrilling to learn how to work with these devices and try to create an application that can have a good impact on the life of the users.

The researchers and the mobile phone producers seek how to create mobile phones and applications that look like they were designed on purpose for the each user. This project gives that opportunity like we will see later. It tries to adapt the application to the user in order to achieve better results.

By the conclusion of this project we expect to make the labeled data ready for use by other applications. This will be of the most importance to make an application that can be used in the future.

1.4 Dissertation Structure

The introduction, where this subchapter is contained, is the first chapter of this thesis and it has the objective of presenting the problem that we are dealing starting by explaining how it emerged and its importance. Also in the **1- Introduction** chapter we can find a detailed explanation of the problem and the final objective of this work. Before starting on a more theoretical explanation of all the process and to finish this first chapter makes sense that we show what motivates us into doing this work and the challenges that rise by this approach to the problem.

Understanding the problem is the first step to solve it. With this in mind we move towards the solution and explore what other authors have done in order to solve it. The **2- Related Work** chapter is the chapter with more subchapters. The objective of this chapter is to provide an overview of the work done in this area to deal with this specific problem. In it concepts like data mining, supervised and semi-supervised learning are explained as well as possible approaches like one-step classification and hierarchical classification and what are the common activities that humans perform on everyday life that worth classifying. Several classifiers are presented and compared by other authors being possible to take some conclusions about each of them and which can be helpful for my own solution. The characteristics of the data that enable to differentiate the instances are studied as well as the prototypes already built.

Following this second chapter we have **3- Software** that presents the programs used to build and test the prototype. Open source software was chosen to use in the programming part of the prototype, to test the application we chose to use open source software and free Android applications.

4- Prototype is the next chapter. Here we present what techniques we decided to use with a brief explanation of the choice. After the experimental setup subchapter is set to inform the reader how the experiments were set in order to get a better evaluation when reading the experiments and results subchapter. But, before, the explanation of the basic architecture of the application can be found. This explanation aims to give to the reader a basic notion of all the components used by the application and how MOA is embed on the application with Android and what a critical role it plays. Realizing this we hope the tests make more sense to the reader. Here all the experiments are presented as well as the results that lead to the final subchapter, conclusions and future work.

Taking into account all the previous chapters an overview of what was accomplished with this new solution is written. Conclusions are taken from the results of the experiments and ideas are for improving the prototype and/or the research in this field are presented. An evaluation of the initial expectation is, also, made.

Capítulo 2

Related Work

To understand what is possible to make in order to innovate we need to know what have been done until now.

That is why in this chapter we analyze the state of art in recognizing activities wearing sensors.

To a better understanding we try to present the works like the application was being built, first understanding the concepts and advancing to the algorithms and techniques.

The starting point is the work that was a breakthrough in this field and we move towards the recent works that contributed for this thesis. During this approach some notes about their conclusions are taken, besides the conclusions the techniques used are briefly explained to understand how they fit in this thesis.

2.1 Single sensor

Activity recognition is not a new thing. Bao & Intille [3] created a system capable of recognizing twenty activities with bi-axial accelerometers positioned in five different locations of the user's person {hip, wrist, arm, thigh, and ankle} – Figure 2.

Figure 2 Location of the sensors on the user's body [3]

This work led to an important discovery, which was possible to get accurate results recognizing activities just using acceleration values gathered by a sensor placed on the thigh or dominant wrist – Figure 3.

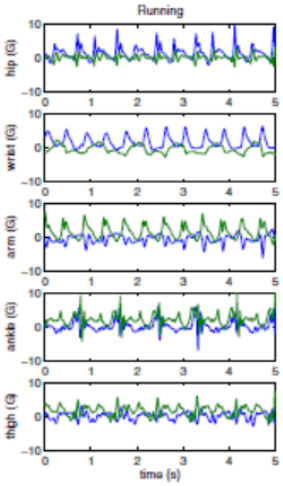


Figure 3 Acceleration signals from five biaxial accelerometers for walking, running, and tooth brushing [3]

Besides these conclusions they treated this problem as a classification problem and compared four classifiers decision table, instance-based learning (IBL or nearest neighbor), C4.5 decision tree, and naive Bayes. Having the decision tree good results – Figure 4 – shows a starting point when begin testing classifiers.

Classifier	User-specific Training	Leave-one-subject-out Training
C4.5	77.31 ± 4.328	72.99 ± 8.482

Figure 4 Summary of classifier results (mean ± standard deviation) using user-specific training and leave-one-subject-out training where both training data sets are equivalent to five laboratory data sessions.

2.2 Data Mining

Like we saw Bao & Intille treated this problem as a classification problem. But to understand what a classification problem is, first we need to understand what data mining is.

Data mining is the process that results in the discovery of new patterns in large data sets. It utilizes methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract knowledge from an existing data set and transform it into a human-understandable structure for further use.

In this thesis case we have to deal with data streams and our interest lies on four main fields:

- Accuracy
- Memory space needed
- Necessary time for learning from a training set in order to be able to predict activities
- Power consumption

In matters of accuracy we have just seen how careful Bao & Intille when comparing the classifiers were. But we will see, on next sections, how this is a matter of high consideration in all the works.

In data mining applications, the bottleneck is time and memory [1].

Before presenting other works and techniques some considerations have to be taken into account.

An algorithm can give faster results if a small amount of information is processed, however the results can be less accurate. When working with data streams the amount of data processed has to be carefully considered because in a mobile phone we have limited memory space, we want to give results in real time so the processor of the mobile has to be taken into account, but nevertheless, we want accurate results.

Later I present some algorithms and techniques used on several experiments to test the four main fields and to address the aforementioned issues.

2.3 Supervised and Semi-supervised learning

Gu et al [4] tried to solve the activity recognition problem with techniques of semi-supervised learning.

Both Masud et al. and Guan et al. use ensemble methods to increase accuracy in partially labeled data (semi-supervised problems).

Traditional classifiers use only labeled data (feature / label pairs) to train. Labeled instances however are often difficult, expensive, or time consuming to obtain, as they require the efforts of experienced human annotators. Meanwhile unlabeled data may be relatively easy to collect, but there has been few ways to use them.

Supervised learning has the problem of just using labeled data, which makes it impossible to improve classifiers because they cannot use new labeled data to create a more accurate model. With supervised learning it is impossible to create an application for everybody that tends to adapt to the users in order to get better results.

Semi-supervised learning addresses this problem by using large amount of unlabeled data, together with the labeled data, to build better classifiers. Because semi-supervised learning requires less human effort and gives higher accuracy, it is of great interest both in theory and in practice [5].

The first model created with the training data will be improved /replaced as more and more recent labeled data is used as training data. The objective of this is to have a model that adapts to the user that provides unlabeled data and the classification becomes more accurate.

An example of semi-supervised learning is **co-training** and **en-co-training** [6].

Co-training (Blum & Mitchell, 1998) (Mitchell, 1999) assumes that features can be split into two sets; each sub-feature set is sufficient to train a good classifier; the two sets are conditionally independent given the class. Initially two separate classifiers are trained with the labeled data, on the two sub-feature sets respectively. Each classifier then classfies the unlabeled data, and 'teaches' the other classifier with the few unlabeled examples (and the predicted labels) they feel most confident. Each classifier is retrained with the additional training examples given by the other classifier, and the process repeats [5].

The difference between en-co-training and co-training is that it uses three classifiers instead of two, which are trained used all the labeled data. All the classifiers are different and they label data after the majority of their predictions.

2.4 Classifiers

The concept of classifier has just been addressed in the explanation of some semi-supervised learning techniques, now we explain it.

2.4.1 Markov

Lester et al. [7] believe that the more common human activities are sitting, walking, walking up/down stairs, riding elevator down/up and brushing teeth. In order to create a model that could predict with great accuracy these activities they compared a static classifier with a HMM (hidden Markov model) classifier. Their training was made using a single stream of sensor data from three locations on the body (shoulder, waist and wrist). To discover the best place to carry the sensor the classification algorithm used joins static classifiers with the objective of gather the more important features, the ones that can be used to distinguish

activities. Each classifier works with a feature making the system flexible. The number of features that we want to work with only depends on the number of classifiers used. A second layer of HMM combines all the classifiers' outputs and estimates the most probable activity while it gives temporal smoothing.

Their objective was to be able to pre-train a set of classifiers so the device would work well for most users right away.

The data is separated in training and test data. All the data is gathered randomly in folds with the same number of segments. Three or four folds are used to train the static and HMM classifiers, the remaining folds are used to run tests.

The system chooses the most important features, other features are chosen on an incremental way, while the HMM allow a continuous tracking of the activities.

This brings advantages because it's possible to make a historic of information that leads to misclassification in the static classification. With this information it is possible to smooth the errors and understand what happens between different activities.

This leads to the conclusion that is possible to train a general classifier and have other classifiers ready to be used depending the situation, the more data from different people we use the more accurate the model is, it is possible to recognize a range of physical activities with a light-weight and unobtrusive wearable device like is possible to see in Figure 5.

	Static Classifier		HMM Classifier	
	Overall Precision	Overall Recall	Overall Precision	Overall Recall
Trained on Location 1,2,3 *(all locations)* Tested on Location 1,2,3 *(all locations)*	79.18%	71.14%	82.07%	81.55%
Trained on Location 1 *(shoulder)* Tested on Location 1 *(shoulder)*	79.37%	*71.26%*	83.84%	*82.64%*
Trained on Location 2 *(waist)* Tested on Location 2 *(waist)*	81.83%	*77.05%*	85.87%	*84.85%*
Trained on Location 3 *(wrist)* Tested on Location 3 *(wrist)*	81.01%	*68.66%*	87.18%	*87.05%*
Single Location Average:	*80.74%*	*72.32%*	*85.63%*	*84.85%*

Figure 5 Overall precision/recall for the static and HMM classifiers trained/tested on all locations and on a single location [7]

This approach gives lower accuracy results than other solutions like we will see ahead.

HMMs didn't appear to be particularly useful to reach the objective of the thesis that is recognizing activities. HMMs are useful in recognizing a sequence of activities to model human behavior [8].

2.4.2 Base-level e meta-level classifiers

The study of classifiers is important to understand there are alternatives to HMMs. With that in mind a performance evaluation was made on some base-level classifiers [8], like Decision trees, C4.5, K-nearest neighbours, SVM and Naïve Bayes.

Although the overall performance of meta-level classifiers is known to be better than base-level classifiers, base-level classifiers are known to outperform meta-level classifiers on several data sets and that was what the authors [8] were trying to discover.

Meta-level classifiers are used in ensemble learning that is the usage of several prediction models. The combination of these models will lead to the final prediction.

Meta-level classifiers can be into three frameworks: voting (used in bagging and boosting), stacking (Wolpert 1992; Dzeroski & Zenko 2004) and cascading (Gama & Brazdil 2000). In voting, each base-level classifier gives a vote for its prediction. The class receiving the most votes is the final prediction.

In stacking, a learning algorithm is used to learn how to combine the predictions of the base-level classifiers. The induced meta-level classifier is then used to obtain the final prediction from the predictions of the base-level classifiers. The state-of-the-art methods in stacking are stacking with class probability distributions using Meta Decision Trees (MDTs) (Todorovski & Dzeroski 2003), stacking with class probability distributions using Ordinary Decision Trees (ODTs) (Todorovski & Dzeroski 2003) and stacking using multi-response linear regression (Seewald 2002). Cascading is an iterative process of combining classifiers: at each iteration, the training data set is extended with the predictions obtained in the previous iteration. Cascading in general gives sub-optimal results compared to the other two schemes.

A set of classifiers was chosen by the authors and they tested their accuracy. The classifiers were:

- **Boosting** (Robert E. Schapire (1990)) is used to improve the classification accuracy of any given base-level classifier. It applies a single learning algorithm repeatedly and combines the hypothesis learned each time (using voting). Assigns a certain weight to each example in the training set, and then modifies the weight after each iteration depending on whether the example was correctly or incorrectly classified.

- **Bagging** (Breiman 1996) is another simple meta-level classifier that uses just one base-level classifier at a time. It works by training each classifier on a random redistribution of the training set.

- **Plurality Voting** selects the class that has been predicted by a majority of the base-level classifiers as the final predicted class.

- **Stacking with ODTs** (Todorovski & Dzeroski 2003) is a meta-level classifier that uses the results of the base-level classifiers to predict which class the given instance belongs to. The input to the ODTs are the outputs of the base-level classifiers.

- **Stacking with MDTs** (Todorovski & Dzeroski 2003) learns a meta-level decision tree whose leaves consist of each of the base level classifiers. Thus, instead of specifying which class the given test instance belongs to, as in a stacked ODT, an MDT specifies which classifier should be used to optimally classify the instance.

To test their accuracy, data collected from a single subject and from multiple subjects were used. For comparison they runned classifiers on data that were independently and identically distributed (IDD) and not (non-IDD).

Classifier	Accuracy(%)			
	Setting1	Setting2	Setting3	Setting4
Naive Bayes(NB)	98.86	96.69	89.96	64.00
Boosted NB	98.86	98.71	89.96	64.00
Bagged NB	98.58	96.88	90.39	59.33
SVM	98.15	98.16	68.78	63.00
Boosted SVM	99.43	98.16	67.90	**73.33**
Bagged SVM	98.15	98.53	68.78	60.00
kNN	98.15	99.26	72.93	49.67
Boosted kNN	99.15	99.26	72.93	49.67
Bagged kNN	99.15	99.26	70.52	46.67
Decision Table(DT)	92.45	91.91	55.68	46.33
Boosted DT	97.86	98.53	55.68	46.33
Bagged DT	93.30	94.85	55.90	46.67
Decision Tree(DTr)	97.29	98.53	77.95	57.00
Boosted DTr	98.15	98.35	77.95	57.00
Bagged DTr	97.29	95.22	78.82	63.33
Plurality Voting	**99.57**	**99.82**	**90.61**	65.33
Stacking (MDTs)	99.00	99.26	89.96	64.00
Stacking (ODTs)	98.86	98.35	84.50	64.00

Figure 6 Accuracy of classifiers for four different settings [8]

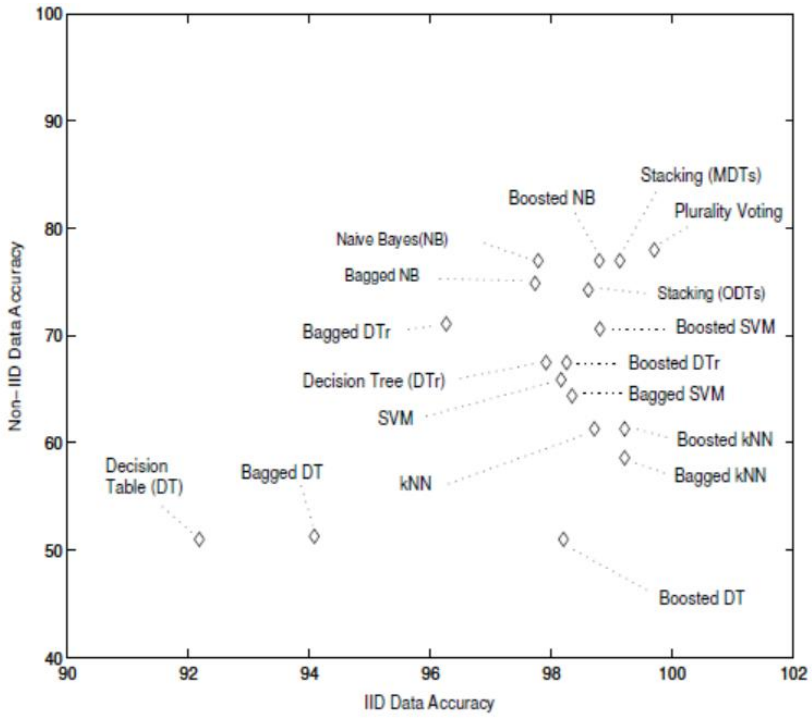


Figure 7 Performance correlations for IDD and non-IDD data [8]

On the Figures 6 and 7 is possible to see one of the conclusions of the authors, that plurality voting has the best performance correlation.

Another conclusion important for this thesis is that we can achieve high accuracy using a single triaxial accelerometer, like the one inside the mobile phone.

Although plurality voting, a meta-level classifier, has a better performance than the others, we think it wouldn't be a good idea to put it in the Android application because it needs more memory space than a base-level classifier, it takes more time to be processed which leads to more CPU usage and, therefore, more battery usage.

The higher precision of at most 4% (according to the tests shown) does not make up for all the disadvantages in the mobile performance.

2.4.3 Naïve Bayes

Naïve Bayes is a base-level classifier, commonly used in the data mining universe like in the aforementioned experiment.

This classifier technique is based on the Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naïve Bayes can often outperform more sophisticated classification methods.

The Naive Bayes algorithm is based on conditional probabilities. It calculates a probability by counting the frequency of values and combinations of values in the historical data. Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred.

The presence or absence of an attribute is not related with the presence or absence of another.

To apply Bayes' theorem as a classifier we need [9]:

- To know the probabilities a priori p(decision$_i$)
- Calculate conditional probabilities p(x|decisão$_i$) – this is simplified by assuming the independence of the attributes

No other classifier can get better results than this one with the same information.

The classifier's error is a theoretical minimum to the generalization capacity of other else classifier. The optimal Bayes' error – Figure 8

- Is proportional to the black area
- Can generate data sets where the error is minimum

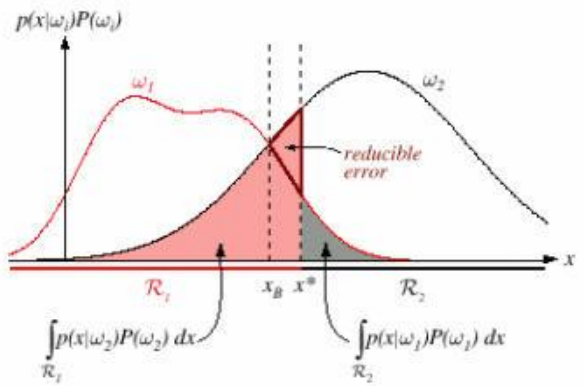


Figure 8 Bayes distribution [9]

In practice all this probabilities are unknown.

To get reliable probabilities from a data set we need endless data.

- O(kp) being p the number of variables and k the number of values from the variables.

Naive Bayes summarizes the variability of a data set in tables of conditional probabilities. The dimension of the model is independent of the number of examples. It is stable to disturbances of the training set, is string to a noise and irrelevant attributes.

All the needed quantities can be calculated passing only once the training set. With this we can conclude that this classifier can be trained on an efficient way, gathering the probabilities of each attribute, respecting its class from the training set. The probability of the unlabeled data can be computed.

Another convenience of this classifier is that it can be trained with a small amount of data so it can predict with good accuracy. This is a great quality when we are dealing with an application in mobile environment, where the memory space is limited as well the processor power.

2.4.4 Decision tree – J48

J48 is the decision tree contained in the WEKA package. Before analyzing experiments results – Figure 9 - it is necessary to understand how a decision tree works, so we can understand the conclusions.

Decision tree uses the strategy divide and conquer, a complex problem is split in smaller simpler problems. This strategy is recursively applied to each problem [10].

It has internal nodes that are a test on an attribute, the branches represent an outcome of the test, the leaf nodes stand for class label or class label distribution and at each node one attribute is chosen to split training examples into distinct classes as much as possible [11]. By its composition it is easy to understand that the discrimination capacity comes from the division of the space into sub-spaces that have a class attached.

To classify new case a matching path is followed until a leaf node [10] – Figure 9.

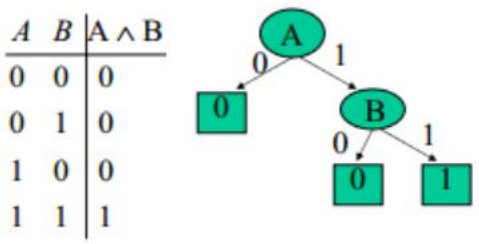


Figure 9 Decisions at each node to get a final classification [10]

At each node available attributes are evaluated on the basis of separating the classes of the training examples – training set has only labeled data. In order to do it goodness (purity) function is used. There are several goodness functions but the one used in C4.5 is the information gain – difference between the information before split and after split [11].

The construction of the tree is driven by the aim of have the minimum entropy, meaning the randomness of the variable.

Then entropy is a function that satisfies three conditions:

- When a node is pure the purity measure should be zero
- When the impurity is maximal (i.e. all classes equally likely), purity measure should be maximal
- Purity measure should obey multistage property (i.e. decisions can be made in several stages)

Having always in mind the creation of a universal model more tests, using classification techniques from WEKA (decision trees (J48), logistic regression and multilayer neural networks), were taken [2].

Regression is usually used when a real-valued output is desired otherwise classification is a natural choice [8].

The use of a straw man [2] serves as control because it, always, indicates the activity that we are trying to predict. Like it was done in other works some activities were chosen because the authors [2] believe that are the more common in everyday life – Figure 10.

	% of Records Correctly Predicted			
	J48	**Logistic Regression**	**Multilayer Perceptron**	**Straw Man**
Walking	89.9	**93.6**	91.7	37.2
Jogging	96.5	98.0	**98.3**	29.2
Upstairs	59.3	27.5	**61.5**	12.2
Downstairs	**55.5**	12.3	44.3	10.0
Sitting	**95.7**	92.2	95.0	6.4
Standing	**93.3**	87.0	91.9	5.0
Overall	85.1	78.1	**91.7**	37.2

Figure 10 Accuracies of activity recognition [2]

The identification of activities with sudden movements it's easier like it can be seen by the accuracy of over 90% in jogging. Jogging seems easier to indentify than walking because it involves more extreme changes in acceleration – Figure 11.

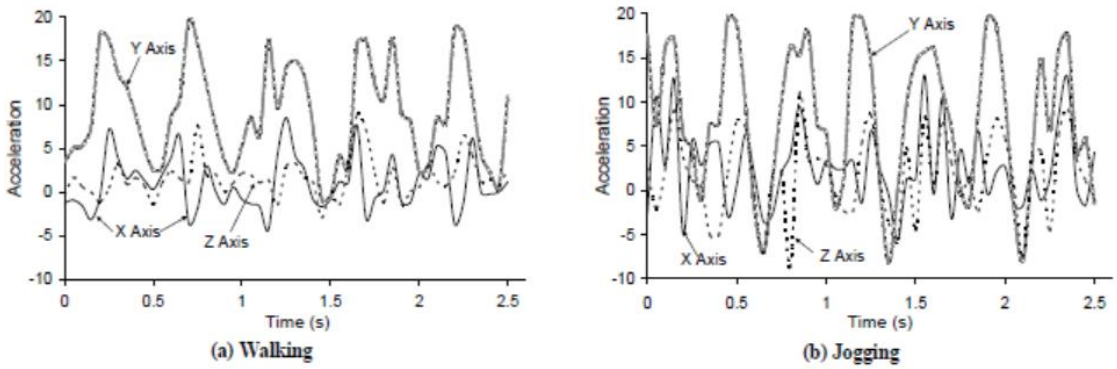


Figure 11 Acceleration plots for Walking and Jogging [2]

But if ascending and descending are joined in just a class named stairs we can observe a better accuracy when the model created using J48 – Figure 12.

		Predicted Class					Accur. (%)
		Walk	Jog	Stairs	Sit	Stand	
Actual Class	Walk	**1524**	7	148	2	2	90.6
	Jog	10	**1280**	31	0	0	96.9
	Stairs	185	33	**784**	4	4	77.6
	Sit	4	0	2	**272**	4	96.5
	Stand	3	1	10	0	**209**	93.7

Figure 12 Confusion matrix for J48 Model (Stairs Combined) [2]

With this model we can predict almost every activity with accuracy over than 90%.

This is a good algorithm for continuous or discrete attributes, it can handle training data with missing values and does pruning (goes back in the tree and tries to remove branches that don't help in the classification and replace them by leaves).

The biggest problem of decision trees is that they assume that all training examples can be stored simultaneously in main memory, and are thus severely limited in the number of examples they can learn from [1].

2.4.5 Hoeffding Tree

Hoeffding trees work by collecting sufficient statistics in the leaves of the tree of the training instances that reach them. Periodically, these leaves are checked to compare the relative merits of each candidate attribute for splitting. The Hoeffding bound or similar metric is used to decide when to be confident that the best candidate is better than the others. At this point the leaf is split on the best attribute, allowing the tree to grow.

Typically, information gain is used to rank the merits of the split candidates, although other metrics could be substituted. In the case of discrete attributes, it is sufficient to collect counts of attribute labels relative to class labels to compute the information gain afforded by a split. [12]

There are incremental learning methods that are reasonably efficient, but do not guarantee that the model learned will be similar to the one obtained by learning on the same data in batch mode. They are highly sensitive to example ordering, potentially never recovering from an unfavorable set of early examples. Others produce the same model as the batch version, but at a high cost in efficiency, often to the point of being slower than the batch algorithm [1].

Advantages of Hoeffding Trees:

- Can be learned in constant time per example
- Do not store any examples (or parts thereof) in main memory, requiring only space proportional to the size of the tree and associated sufficient statistics

It can learn by seeing each example only once and therefore does not require examples from an online stream to ever be stored.

Catlett [13] and others noticed that, in order to find the best attribute to test at a given node, it may be sufficient to consider only a small subset of the training examples that pass through that node.

A statistical result known as the Hoeffding bound is used to decide how many examples are necessary at each node in order to find the best attribute to test. The attribute chosen using n examples (where n is as small as possible) is the same that would be chosen using infinite examples.

The incremental nature of the Hoeffding tree algorithm does not significantly affect the quality of the trees it produces that are asymptotically arbitrarily close to the ones produced by a batch learner.

VFDT allows the use of either information gain or the Gini index as the attribute evaluation measure. It includes a number of refinements to the normal Hoeffding tree algorithm:

- It can solve ties splitting on the current best attribute if $\Delta\bar{G} < \epsilon < \tau$, where τ is a user-specified threshold.

- Recomputing G is the most significant part of the time cost per example. VFDT allows the user to specify a minimum number of new examples n_{min} that must be accumulated at a leaf before G is recomputed. This effectively reduces the global time spent on G computations.

- VFDT processes examples faster than they arrive, which will be the case in all but the most demanding applications, the sole obstacle to learning arbitrarily complex models will be the finite RAM available. To solve this memory issue VFDT deactivates the least promising leaves in order to make room for new ones. Memory usage is also minimized by dropping early on attributes that do not look promising.

- VFDT can be initialized with the tree produced by a conventional RAM-based learner on a small subset of the data. This can give VFDT a head start" that will allow it to reach the same accuracies at smaller numbers of examples throughout the learning curve.

- VFDT can rescan previously-seen examples. This means that VFDT need never grow a smaller (and potentially less accurate) tree than other algorithms because of using each example only once.

Like in other works to verify the better performance of one classifier it was compared with others, in this work VFDT was compared with C4.5.

Regarding the accuracy as a function of number of training examples C4.5 has an early advantage that comes from the fact it reuses examples to make decisions on multiple levels of the tree it is inducing, while VFDT uses each example only once. – Figure 13.

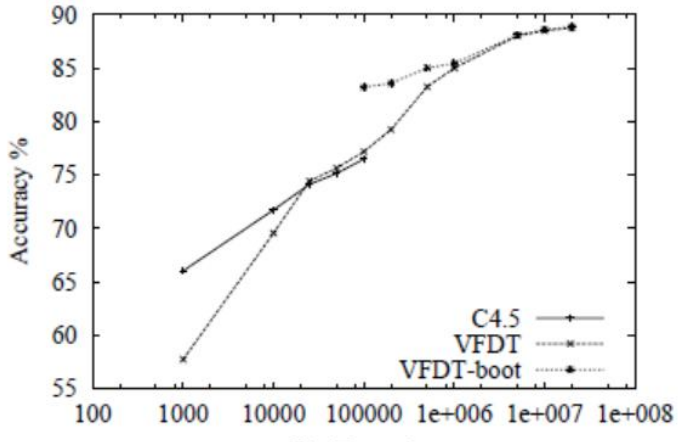


Figure 13 Accuracy as a function of the number of training examples. [1]

The average number of nodes in the trees induced by each of the learners can be seen on Figure 14 and we can notice that VFDT and VFDT-boot induce trees with similar numbers of nodes.

VFDT can substantially increase the comprehensibility of the trees induced relative to C4.5. It also suggests that VFDT is less prone than C4.5 to overfitting noisy data.

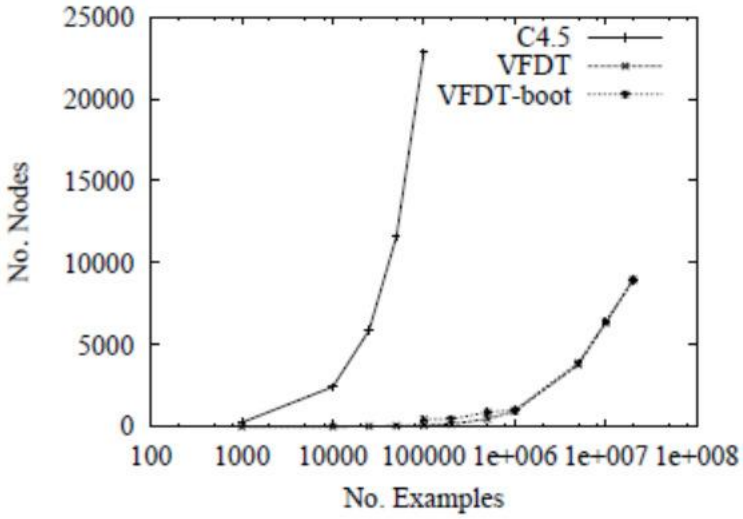


Figure 14 Tree size as a function of the number of training examples. [1]

Also dealing with noise VFDT can get more accurate results, what can be important since we are dealig with a mobile phone- Figure 15.

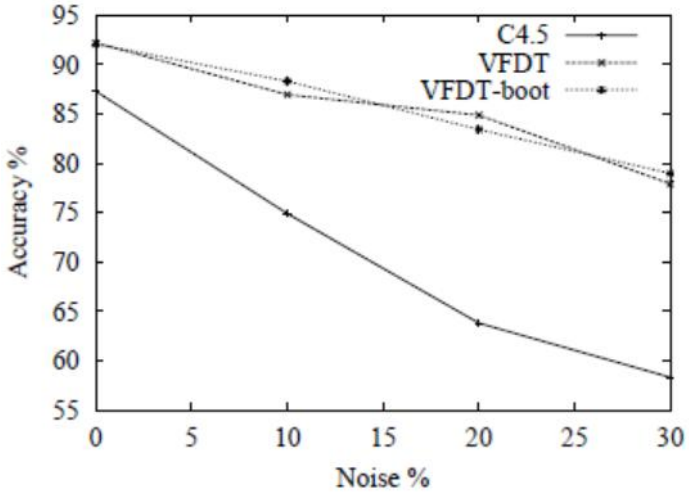


Figure 15 Accuracy as a function of the noise level. [1]

Hoeffding trees allow learning in very small constant time per example, and have strong guarantees of high asymptotic similarity to the corresponding batch trees.

VFDT is a high-performance data mining system based on Hoeffding trees. Empirical studies show its effectiveness in taking advantage of massive numbers of examples. VFDT's application to a high-speed stream of Web log data is under way.

2.5 Hierarchical classification

All the experiments presented until now the authors aim to classify the data only in one step, but it is important to be aware that the classification can be made in multiple steps using hierarchical classification. On hierarchical classification a binary tree is created, meaning that in each node only two classes are possible. In this case, activity prediction, the classification of movements can be made first in motion or motionless as we shall see below. The accuracy is higher the higher up in the tree because first is done a general division, i.e. a bag full of balls and cubes of different colors the first classification will be if it is ball or cube.

The classes on same level on the tree should be independent [14] so there is no possible uncertainty when choosing the path. The use of binary trees gives a flexible structure to the classifier, classes can be added without affect the rest of the tree.

In this thesis, like it was said before, the tree should reflect an approach from a general to a specific activity. The way this is done is of great importance to the processing efficiency. All possibilities of classification must appear on the tree – Figure 16.

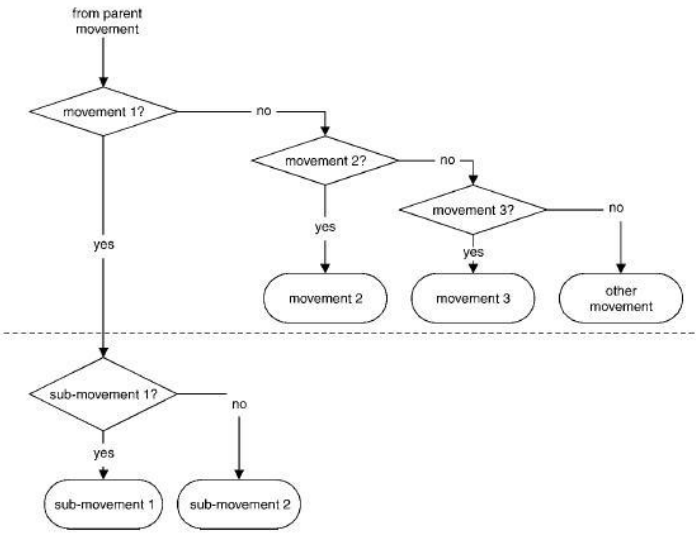


Figure 16 Classification framework, which is a structured as binary tree [14]

The hierarchical approach gives the opportunity to choose a classification algorithm for each node. This opportunity is a great benefit because we can choose the algorithm that gives better results for each specific situation, aiming to have the highest accuracy possible.

To choose the better algorithm, in each node, they have to be tested with the same training set. The algorithm has to do predictions where there is no ambiguity. On the Figure 17 an example of a tree can be observed where the tests done on each node get rid of all the ambiguity.

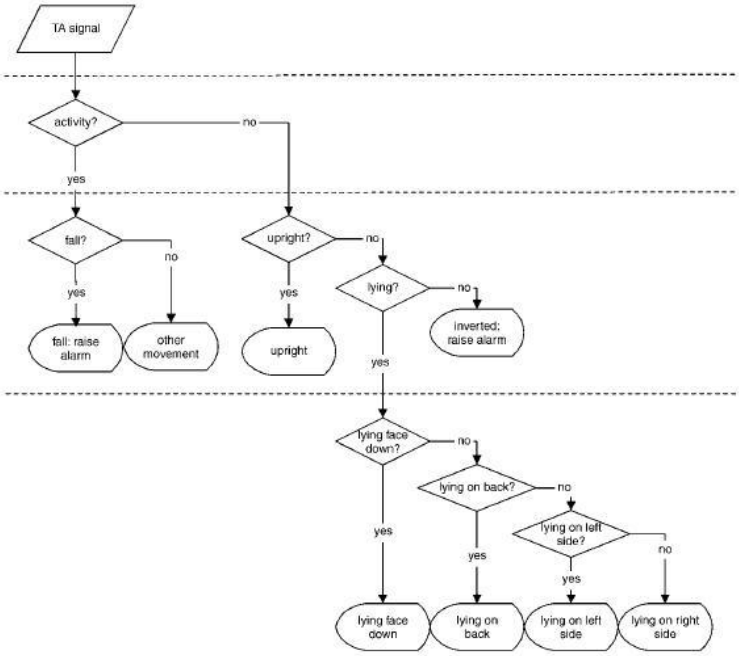


Figure 17 Detection falls. Tests in each node with no ambiguity [14]

2.5.1 SVM

As mentioned, above, this investigation can be very useful in the health care area. With this objective some authors [15] performed some experiments on the movement classification using the hierarchical approach, trying to get more accurate results than doing the classification in one step like the experiments demonstrated until now.

Samples with a frequency of 1Hz were used in a try to classify movements considered common in everyday life like sitting, standing up, lying down, walking, posture transition and gentle motion.

The authors have chosen to implement a two level hierarchical classification.

On the first level the separation between motion and motionless activities was done, as we can see this test takes out all the ambiguity like it was supposed to do. After this first classification has been done SVM classifiers were used to recognize the specific activity inside the two general groups – Figure 18.

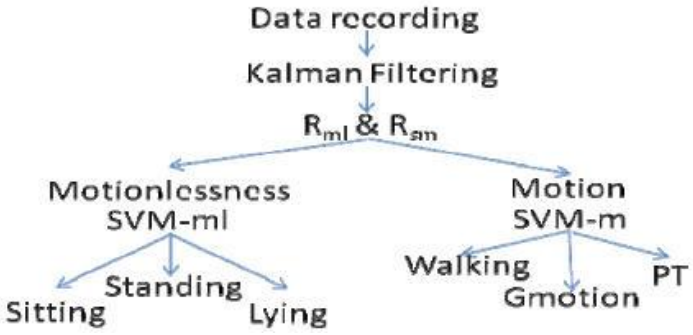


Figure 18 Flowchart describing the classification process [15]

Support vector machines (SVM) are a group of supervised learning methods that can be applied to classification or regression. Support vector machines represent an extension to nonlinear models of the generalized portrait algorithm developed by Vladimir Vapnik. The SVM algorithm is based on the statistical learning theory and the Vapnik-Chervonenkis (VC) dimension introduced by Vladimir Vapnik and Alexey Chervonenkis [16].

A simple way to understand this model is to imagine the examples as points in space, positioned so that a space between the two groups is perceptible, being this space the biggest possible. The data in the test set will be placed as dots in these two spaces.

Rules were defined so this classification moved without problems and ambiguity, an example is the 10s that has to be without movement so an activity can be considered motionless – Figure 19.

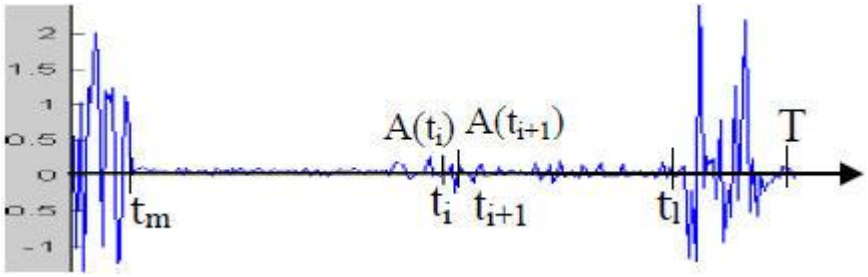


Figure 19 Acceleration signal with period of inactivity [15]

To be motionless cannot be considered absence of any movement because there is always noise present on the data. This noise had to be taken into account when defining the rules so noise could be perceived in relation with little movements, so the first level classification (motion/motionless) could be more accurate- Figure 20.

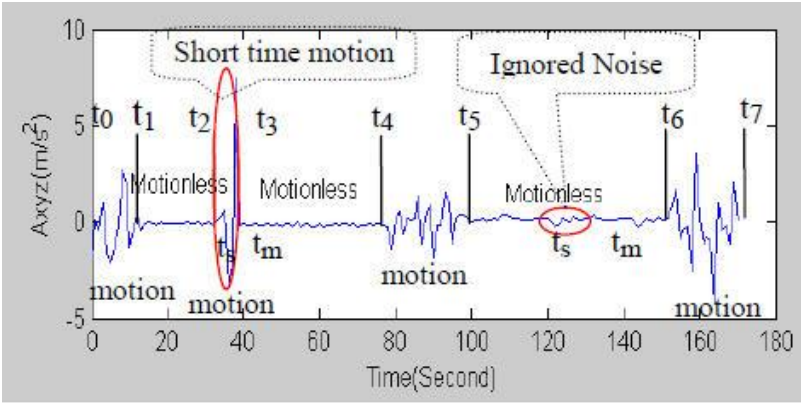


Figure 20 Distinguishing short time motion and noise of gentle motion [15]

Using only one classifier having a training set with all the data as the input, motion and motionless activities from a person, the classifier can only get accurate results when analyzing data from the same person- Figure 21 – not being able to fulfill the main objective that was the creation of an universal model.

Act. / Subj.	Sit.	Sta.	Lying	Walk	PT	Gmotion	Accuracy
Sub1	514	91	116	60	9	20	97.7%
Sub2	133	61	262	40	11	16	70%
Sub3	80	74	39	60	14	20	58.2%
Sub4	190	491	282	40	10	12	35.3%
Sub5	739	254	0	70	10	30	43.6%
Sub6	410	265	113	100	9	8	63.6%
Sub7	738	398	191	22	13	4	72.5%
Sub8	133	61	262	43	8	0	81.9%
Sub9	163	121	186	96	17	0	60%
Sub10	148	64	36	80	17	4	62.7%

Figure 21 Classification results using only one classifier [15]

The average precision was only 63.8% when trying to predict the activities from multiple individuals. The good accuracy results recognizing the activities from subject 1 prove that the model was over fitted to his movements because the classifier had only his data when it was trained.

Using one classifier and data only from one subject leaded to a classification not accurate at all. However using data only from one subject is easy to, in a hierarchical approach, to draw a line between motion and motionless activities as can be seen in Figure 22.

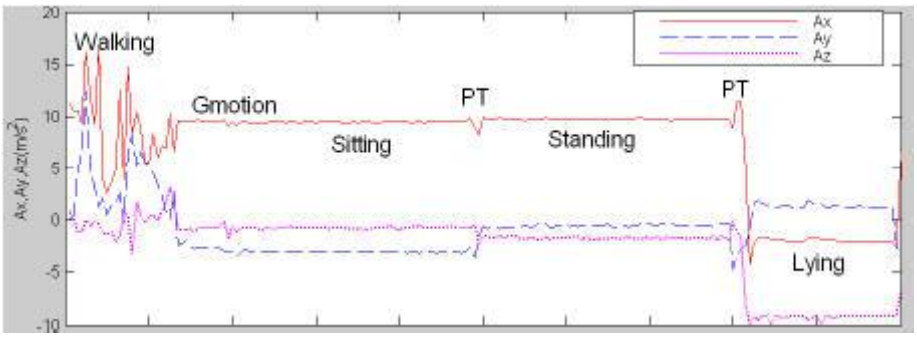


Figure 22 3D-accleleration signals measured above 6 activities [15]

With this conclusion in mind the authors used two SVM classifiers, the first one to differentiate between motion and motionless activities - Figure 23 – and the second one to classify the activities between each group, like it was previously mentioned - Figure 24 and 25.

Subjects	Instance number		Accuracy		
	Inactive	*Motion*	*Inactive*	*Motion*	*Average*
Sub1	721	89	98.4%	92.1%	95.3%
Sub2	456	67	81.6%	88.1%	84.9%
Sub3	193	94	96.5%	85.1%	90.8%
Sub4	963	62	81.6%	75.8%	78.7%
Sub5	993	110	74.4%	63.6%	69%
Sub6	788	117	70.6%	83.8%	77.2%
Sub7	1327	39	91%	71.8%	81.4%
Sub8	456	51	81.6%	92.2%	86.9%
Sub9	470	113	96.4%	86.7%	91.6%
Sub10	248	101	93.5%	88.1%	90.8%

Figure 23 Classification results using two SVM classifiers [15]

Activity	Sit.	Sta.	Lying	Average Accuracy
Sit.	**3070**	207	123	
Sta.	704	**1593**	33	
Lying	27	19	**1069**	
Total	3801	1819	1225	
Accuracy	80.8%	87.6%	87.3%	**83.7%**

Figure 24 Motionless confusion matrix using 3MotionlessM [15]

Activity	Walk	PT	Gmotion	Average Accuracy
Walk	**546**	23	0	
PT	43	**94**	59	
Gmotion	22	6	**55**	
Total	611	123	114	
Accuracy	89.4%	76.4%	48.2%	**81.9%**

Figure 25 Motion confusion matrix using 3MotionM [15]

The lustiness of this method can be further increased using majority voting to reach a prediction. This means that the more voted classification will be the final decision, this way data with noise or misclassified can be avoided.

However, like in other experiments using other approaches, still very hard to differentiate activities with similar acceleration values.

Another conclusion of this work is that using rule-based reasoning and multiclass SVM algorithms the classification can be improved.

The great accuracy achieved in these experiments was highly considered when doing the Android application, although the tests done using decision trees (C4.5) had higher accuracy. However a direct comparison cannot be done because the data used was different.

2.6 Battery

During the conception of this project not only the important results of the classifiers tests were taken into account. The application needs to be appealing to the customer, and for that it has to give good results and have the smallest impact on the phone performance.

With this objective in mind a way to save battery and, at the same time, gather enough data had to be found.

A random use of the accelerometer, turning it on and off, is not an option because the samples need to have the same duration and in order to conduct robust experiments a periodicity needs to be found when retrieving data.

Experiments conducted by Yi Wang et al [17] leaded to the conclusion that what could be done to save battery was the creation of duty-cycles (this concept will be explained later) – Figure 26.

Sensor	Duty Cycles	Computation Time/Sample	Energy(J)/Sample
Accelerometer	6s sensing + 10s sleeping	< 0.1s	0.359

Figure 26 Sensor duty cycles, device computation time and sensor energy cost per sample [17]

The accelerometer is the sensor that spends less energy – Figure 27.

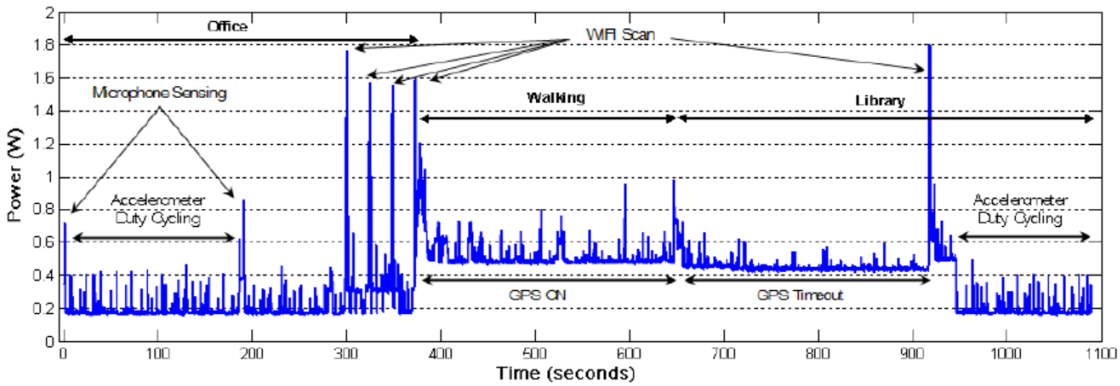


Figure 27 Power usage of sensors [17]

We continue to believe that creating the model is the most expensive action in terms of power consumption.

2.7 Data Extraction

Until now we have been talking about gathering data, testing data but nothing have been said about what is the data, what information can we infer from it.

Ravi et al. [8] showed that several features could be extract from the data:

- Mean
- Standard Deviation
- Energy
- Correlation

The usefulness of these features has been demonstrated in prior work (Bao & Intille 2004).

Standard deviation was used to capture the fact that the range of possible acceleration values differ for different activities.

The periodicity in the data is reflected in the frequency domain. The energy feature is calculated in order to capture data periodicity.

Correlation is calculated between each pair of axes as the ratio of the covariance and the product of the standard deviations. Correlation is especially useful for differentiating among activities that involve translation in just one dimension – Figure 28.

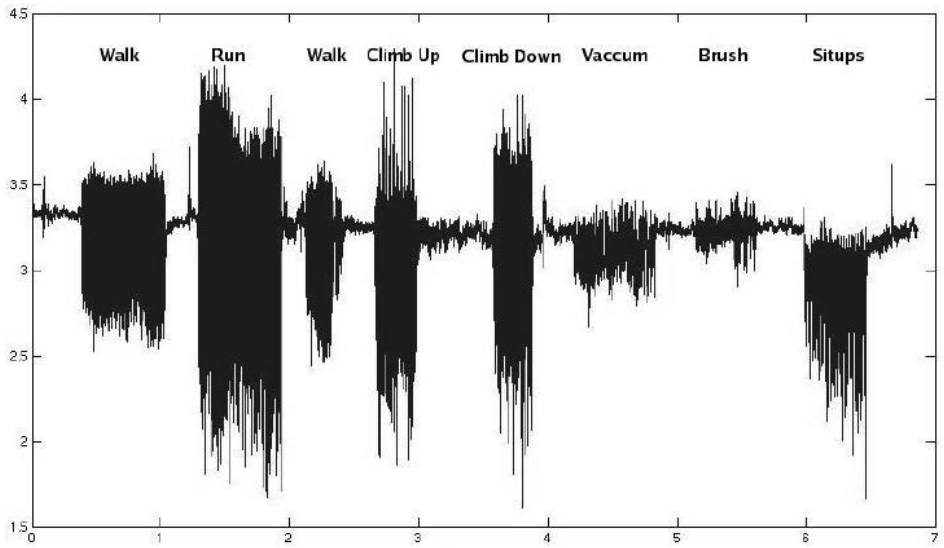


Figure 28 X-axis readings for different activities [17]

For activities like climbing stairs that involve translation in two dimensions cannot be used to distinguish climbing up or down stairs.

2.8 Sliding Windows

The robustness of the classification depends how the data is fetched. A certain order must be maintained so processing data can indicate an activity in a determinate moment.

The objective is to take into account only recent data when making decision. This saves computing power which is very import when working in a mobile platform.

Sliding windows model is useful for sensors network where only the recent events are important. It reduces the memory usage because only saves the window with the data [18].

There are two types of sliding windows [19]:

- Sequence-based windows
- Timestamp-based windows

An algorithm that can be used with a sequence-based window is to maintain a reservoir sample for the first *n* data elements in the stream, and thereafter to stop maintaining the sample except that when the arrival of a new data element causes an element present in the sample to expire, the expired element is replaced with the newly-arrived element.

It requires only memory for storing the predetermined number of data elements.

There are several algorithms that differ in the storage of the data. For example a possible algorithm is to add new elements to a "*backing sample"* and generate the sample of size *k* by down-sampling from that sample, when an element expire it is removed from the backing sample. However another algorithm that needs less memory was created. *Chain-sample* generates a sample of size *l,* to produce a sample of size *k*, maintain *k* independent chain-samples. An element only is chosen to be part of the sample if it has a determinate probability and it will replace another element that expires.

In timestamp-based sliding windows there is an algorithm called *"priority sample"*.

As each data element arrives a priority is assigned randomly between 0 and 1. The element with higher priority and non-expired is included in the sample. In memory only stores elements that can't be replaced with other ones with a later timestamp and higher priority since only these elements can be used in the sample.

Weighing the two types of slideing windows an approach using sequence-based windows seemed more appropriate because defining an interval is not viable because the application can be turned off and not gathering new data. Taking this into account a simpler sequence-based window seemed like a better solution. A number of instances is defined and when the limit is

reached the old element in the file will be replaced by the new one, keeping the file updated and always with the same size.

2.9 Duty cycles

During this thesis the importance of saving battery has been being mentioned. The embedded sensors in the mobile devices are major power consumption [17]. Since it is mandatory the use of the accelerometer to retrieve data so the application can classify it, a way of saving battery was needed.

The accelerometer is a low cost sensor however battery can be saved. Because movements can be changing constantly the accelerometer needs to be sampled periodically. It cannot be turned off and on without criteria since that would mix the data and would be impossible to classify accurately. What can be done is to implement duty cycles. Duty cycles define the interval that the accelerometer is on and off. This ensures that the samples have the same time length which won't be a problem to the sequence-based to the sliding window because we can always know which is the recent data in case we need to replace the old data in the file.

With some experiments authors [17] came to the conclusion that 6s with the accelerometer working enough data could be gathered to indicate an activity and then it could be turned off for 10s – Figure 29.

Sensor	Duty Cycles	Computation Time/Sample	Energy(J)/Sample
Accelerometer	6s sensing + 10s sleeping	< 0.1s	0.359

Figure 29 Sensor duty cycles, device computation time and sensor energy cost per sample [17]

This approach gives enough time for our program to classify the instances gathered and don't waste battery with the accelerometer having it turned on when it is not needed.

2.10 Existing prototypes

Although the market for applications that focus on enhancing the physical activity by classifying the activities is not yet being fully availed there are some prototypes.

To build an application like DiaTrace research was needed [20].

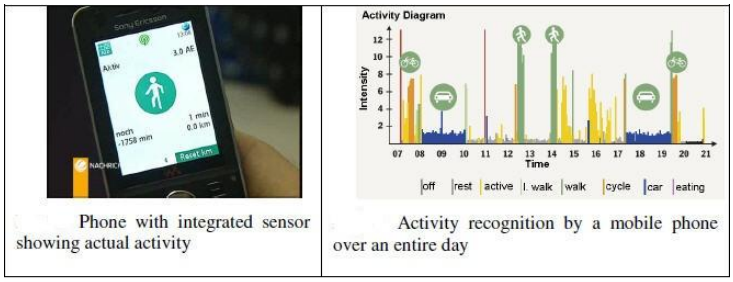
A healthy human being takes, in average, 220msec for an optical response. However a reflex is a direct reaction without processing in the brain which takes approximately 0.06s that is 16 Hertz. According to Shannon theorem a double sampling rate is needed, which gives 32 cycles per second. The sampling rate is relevant for body movements because if the sampling rate of the accelerometer was less than 32 cycles the sample distribution would be abnormal.

DiaTrace uses a reconstruction of the true course of acceleration by interpolation of the scanned acceleration value of each axis. This preprocessing compensates the varying sampling rate as well as the rough quantization. This leads to a new input signal for the pattern recognition. By using relevant features, a long term assessment of daily activities is possible by DiaTrace – Figure 30.

The program measures all daily activities and warns the users of what extra activities they need to do. This application takes into account the huge importance of social networks enabling the users to share their results in order to make the app more appealing and to instill competitiveness in its users.

The authors [20] ensure that wearing the mobile phone on the trousers front pocket they achieve an accuracy of 95% in recognizing activities. For that they claim using techniques of data mining and pre-processing the acceleration of the data.

This experiment has a downside: it lacks explanation in which is the algorithm used, how the data is treated, what amount of data is taken into account and how the classification is done. However it shows how this niche market is not explored yet and the business opportunity it can be.



Phone with integrated sensor showing actual activity

Activity recognition by a mobile phone over an entire day

Activity top ten of the buddies

Electronic medals

Figure 30 DiaTrace application [20]

Capítulo 3

Software

3.1 Moa - Massive Online Analysis

Until now we have seen that multiple authors chose Weka and worked with its classifiers. Since we were working with data streams we choose MOA.

MOA (Massive On-line Analysis) is a framework for data stream mining. It includes tools for evaluation and a collection of machine learning algorithms. Related to the WEKA project, it is also written in Java, while scaling to more demanding problems.

The goal of MOA is a benchmark framework for running experiments in the data stream mining context by proving:

- storable settings for data streams (real and synthetic) for repeatable experiments;
- a set of existing algorithms and measures from the literature for comparison; and
- an easily extendable framework for new streams, algorithms and evaluation methods.

The workflow in MOA follows the simple schema depicted below (Figure 31): firstly a data stream (feed, generator) is chosen and configured, secondly an algorithm (e.g. a classifier) is chosen and its paramters are set, third the evaluation method or measure is chosen and thirdly the results are obtained after running the task [21].

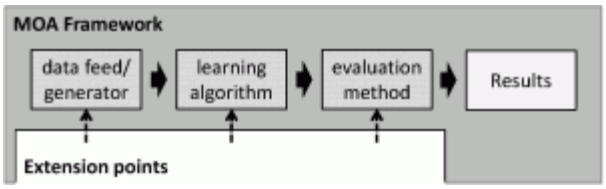


Figure 31 Workflow in MOA [21].

MOA includes Hoeffding Trees that aren't present in Weka and are of most importance when dealing with data streams. This work flow will be explained later integrated with the work developed. It will enable us to understand how the classification is done.

3.2 Android

Android [22] is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language. The following diagram – Figure 32 - shows the major components of the Android operating system.

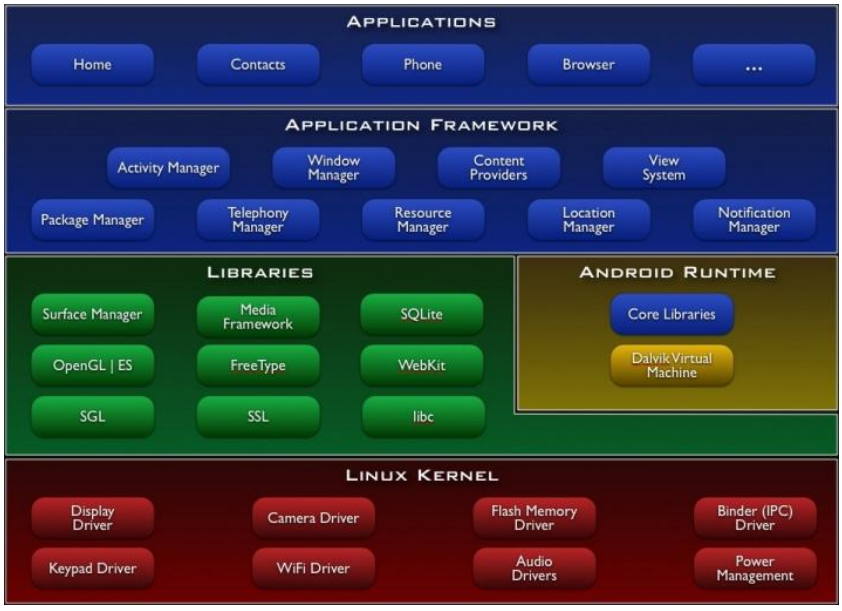


Figure 32 Android components' [22]

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Developers have full access to the same framework APIs used by the core applications.

Using Android and Moa the Java programming language it becomes easier its joint implementation.

3.3 Power Tutor

An application was needed to measure the battery usage.

PowerTutor is an application for Google phones that displays the power consumed by major system components such as CPU, network interface, display, and GPS receiver and different applications. The application allows software developers to see the impact of design changes on power efficiency. Application users can also use it to determine how their actions are impacting battery life. PowerTutor uses a power consumption model built by direct measurements during careful control of device power management states. This model generally provides power consumption estimates within 5% of actual values. A configurable display for power consumption history is provided. It also provides users with a text-file based output containing detailed results. You can use PowerTutor to monitor the power consumption of any application. [23]

Capítulo 4

Prototype

4.1 Techniques used

After an intensive study of previous work done, we decided to use Naïve Bayes and Hoeffding trees that were explained before.

Naïve Bayes was chosen because it treats the attributes as unrelated and it proved to give good results in all the experiments despite its simplicity. In the case of Hoeffding Tree, it was selected because of its advantages (learning in constant time per example and not storing any examples in main memory) and good performance dealing with streams.

These two classifiers seemed adequate for our application that has limited memory space and uses data from a stream. To find the best technique we decided to follow what was done on other works, comparing the accuracy of the classifiers.

4.2 Experimental setup

Before testing the application some decisions had to be made in order to have a controlled environment. This approach allows us to know the expected results for each test beforehand.

The placement of the mobile was an important issue. Without having the option of placing sensors in different parts of the human body we have chosen the trousers' front pocket [24] to conduct all experiments.

To create the models, data from two persons was used. This data contained the average of the values recorded by an accelerometer for several hours doing activities of walking, running, standing idle and sitting. In the total approximately 27 thousand instances were used.

The unlabeled data (files from approximately 16 thousand to 30 thousand instances) was not used to create the model. It belongs to the two people that contributed with data to create the model. There is, also, data from a third person that was not used for learning the models.

The programmer had to choose between timestamp and sequence-based sliding windows depending if window is defined according to a predefined interval or a predefined amount of data. The sequence-based sliding windows were used as well as the 6s long duty cycles.

A value of 70% probability is used to proceed with semi-supervised learning as explained in the next section. This allows creating new models by appending to previous data the recent labeled data with at least 70% of certainty.

4.3 Architecture

We have just seen the conditions on which the prototype will be tested. However before showing the results it is important to understand how it works.

First we present the architecture of the all system to provide a better view of all the components that interact inside the smartphone to make this application work.

The four main components of the application are the accelerometer, SD card and Android with MOA embed – Figure 33. These four components are always connected being the SD card the least used. It is only used to access the files with the training set for creating the models and to save labeled files between classification steps (in hierarchical between 1st and 2nd classification).

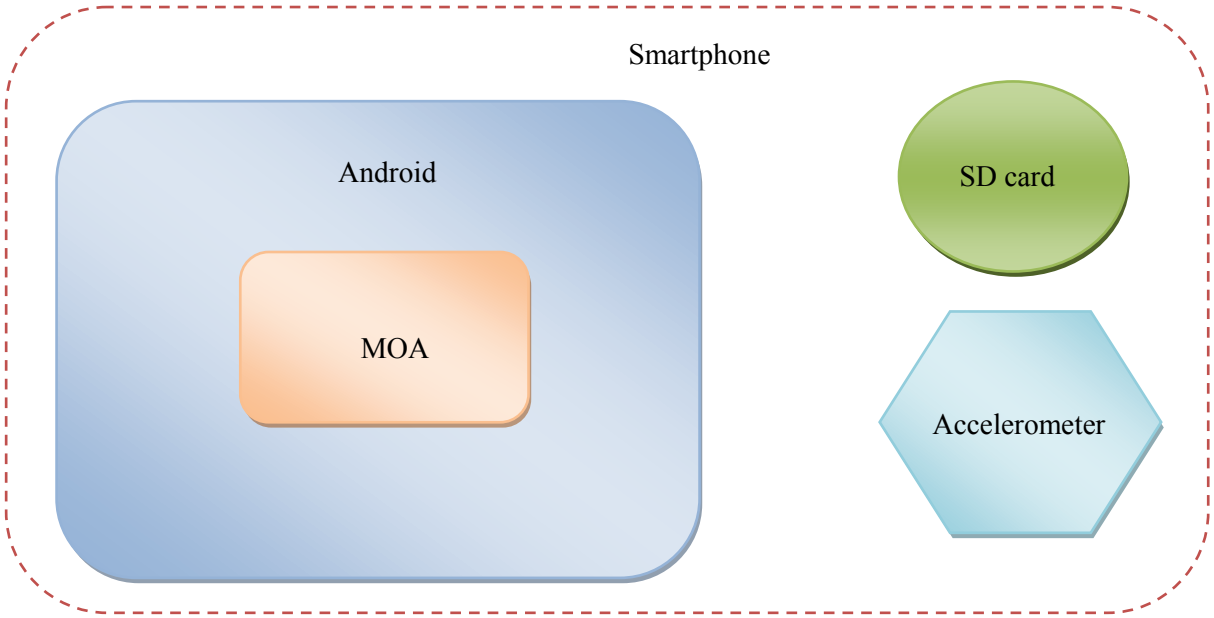


Figure 33 Application components

The accelerometer provides data that is gathered using Android functions and passed to functions that use MOA in order to be classified. Which are these functions and how do they work will be explained next. The usage of MOA is the main aspect of this application. It is this program that handles all the operations with classifiers, models, labeled and unlabeled data; Android just supports thys program and interconnects it with all the other components.

In interface terms is Android straightforward, what really matters is what the user cannot see. The program is divided in some main functions that use MOA aided by Android (Figure 34):

1. Train the model: firstly we need to load a labeled file from the SD card and turn it into a data stream. A classifier is chosen and it tries to classify the next instance of the stream and then learn from it. This attempt to classify an already labeled instance has the porpoise of getting the model's accuracy by counting how many instances would the model label correctly in total.

2. Load unlabeled data: this function actually is not supposed to be used during the normal usage of the application. What it makes is simply to get a file from SD card with unlabeled instances in order to procced to their classification. It will be a very useful function during the tests to create stress situations where not only one instance has to be classified. It is also easier to evaluate the classification results because although the data is unlabeled we know how it should be labeled.

3. Classify unlabeled data: this is the last function related with MOA and it is the one that will give the results that the user will be able to see. It uses the model created by function **1** and it can use the unlabeled data from function **2** or, in normal operation of the application, use the only unlabeled instance that comes from the accelerometer. This function can also save the recent labeled data on a file or just output it to the screen. To get to the labeled instances the function contains a cycle that basically using the model get the probabilities of the instance belonging to each of the possible classes, being the class chosen the one with higher probability, of course.

All these functions can have additional features depending whether we are dealing with a semi-supervised or a supervised learning approach. We defined that an instance classified with 70% or more of certainty (function 3) is added to the training set in the semi-supervised approach.

The hierarchical approach has two functions of each (**1,2,3**) because first we need to create the model to classify the instances in Dynamic or Static movement. When this classification

occurs two files are created where unlabeled data is written, one file only with Dynamic unlabeled data and other with Static unlabeled data. Because of these two files a flag was created whether if both files had data. If that was true (what only happens in stress conditions for the tests) the second part of the hierarchical approach takes turns. First it creates a model for the Dynamic (Running, Walking) or Static (Standing Idle, Sitting), then loads the files created by the first classification, it loads the Dynamic unlabeled file whether we are doing the Dynamic classification or loads the Static unlabeled file whether we are doing the Static classification. In the end we have the instance labeled by function **3**. If we are using semi-supervised learning and doing this hierarchical approach the 70% can be found on both classifications (1st classification and 2nd classification in Dynamic and Static).

The one-step classification is pretty straightforward just implementing function **1,2 and 3** being the only difference between the semi-supervised and the supervised approaches are the adding of unlabeled instances, in the sormer approach, to training set when the classification has a certainty of 70%, at least.

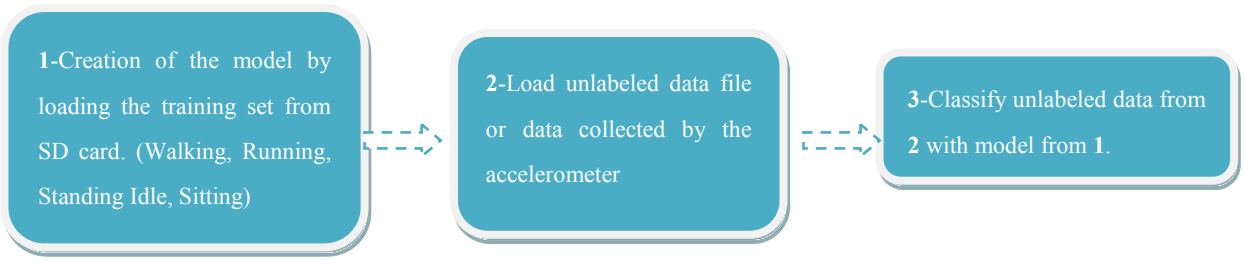


Figure 34 Main functions on prototype behavior using one-step supervised classification.

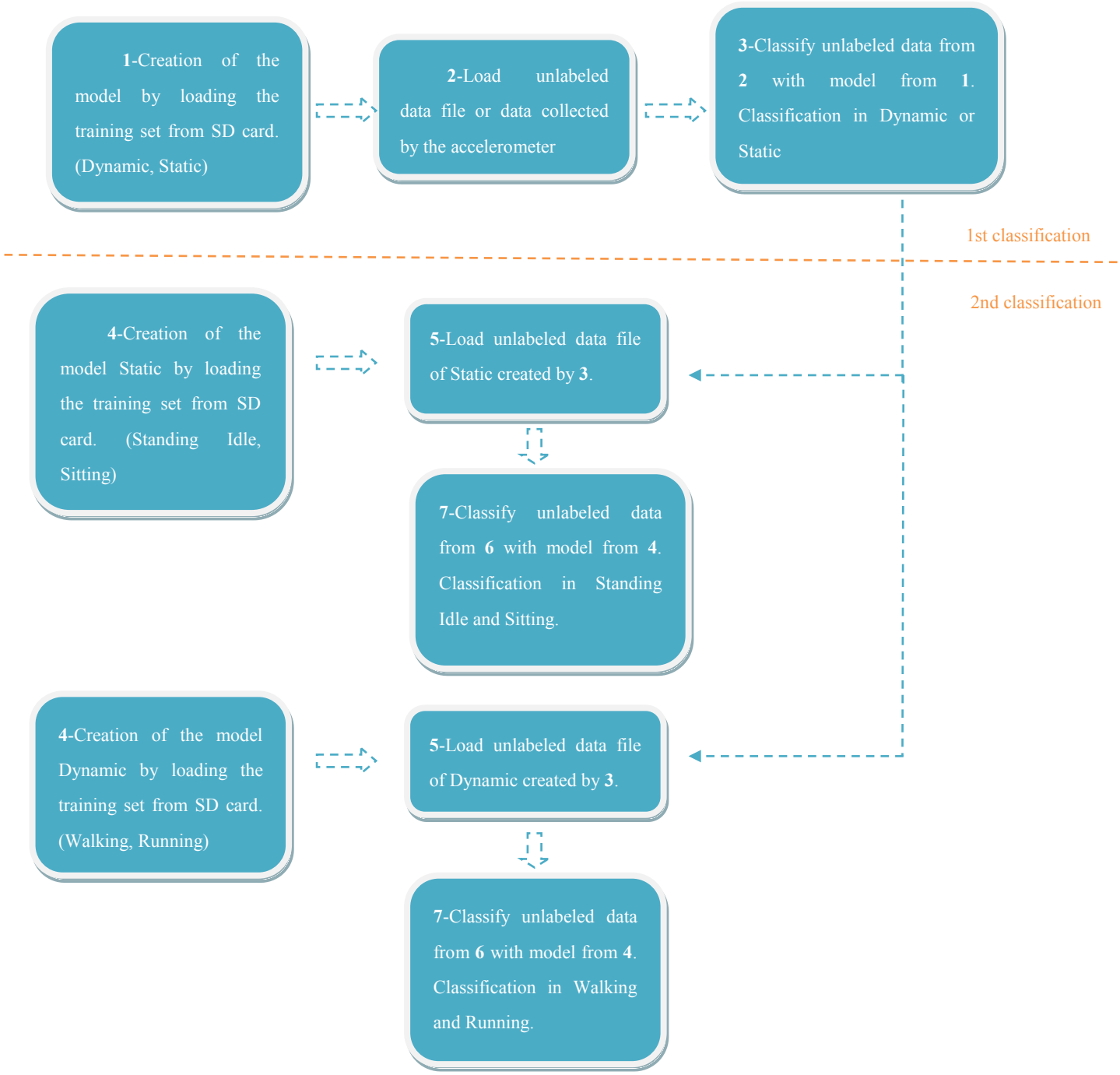


Figure 35 Main functions on prototype behavior using supervised hierarchical classification.

The numbers on second classification (Figure 35) are the same because the aforementioned reason: the hierarchical approach takes turns doing the second classification.

This is the basic architecture of the prototype. These are the main parts the use MOA and handle all the data.

4.4 Experiments and Results

Previously, labeled data from three different persons was recorded. The data contained four activities: walking, running, sitting and standing idle.

Using MOA, two different approaches were taken:

Firstly, models were induced using both Naïve Bayes and Hoeffding Tree. The classifiers were tested on unlabeled data from one person (Table 1).

Table 1: Classifiers' accuracy

	Naïve Bayes	Hoeffding Tree
Accuracy	92.00%	94.78%

Secondly, a hierarchical approach with two levels was also carried out using the same classifiers. In the first level the classification was made between Dynamic and Static activities – table 2. Then, in the second classification, a model was built on each category so we could proceed the classification on Walking and Running on the Dynamic category, and Sitting and Standing Idle on the Static one (Table 3).

Table 2: accuracy in the first level of the hierarchical approach.

Dynamic vs. Static	Naïve Bayes	Hoeffding Tree
Accuracy	82.11 %	99.85%

Table 3: Classifiers' accuracy in the second classification of the hierarchical approach.

	Naïve Bayes	Hoeffding Tree
Running, walking	76.25%	99.05%
Sitting, standing idle	99.83%	99.93%

To test the effectiveness of the classification, unlabeled data of a person, which was not used for training the classifier, was used. Here are the results for the walking activity – table 4.

Table 4: Classifiers' Accuracy for the walking activity using as test set data from a person without data on the training sets

	One-step classification	Hierarchical 1st classification	Hierarchical 2nd classification
Naïve Bayes	86,37%	90,17%	84,27%
Hoeffding Tree	67,65%	94,04%	88,09%

These results only show that Hoeffding Tree is better than Naïve Bayes for the walking activity on a hierarchical approach. However, Naïve Bayes gives better results on the one-step approach (Table 4). Further tests were needed for the remaining activities. Additionally, a semi-supervised approach was also used, besides the supervised one described above, in order to evaluate the usefulness of using unlabeled data from the user that is being tested.

In order to adapt the model to the normal user of the cell phone a threshold of 70% was created. This meant that data labeled with at least 70% of certainty would be recorded on the training file of the classifier, so a new model, more suitable to the user, could be generated. This approach is compared against de supervised approach (Figure 36).

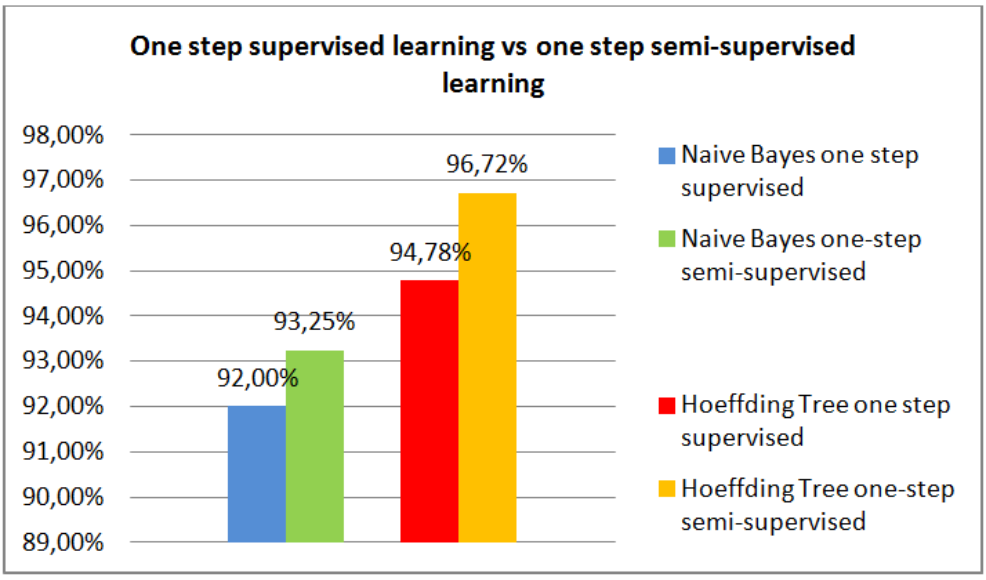


Figure 36 Accuracy of one step classification using both supervised and semi-supervised learning.

It is easier to check the better accuracy when using a semi-supervised approach.

The accuracy using Naïve Bayes in a hierarchical approach on Static classification decreases when we use new labeled data (i.e. the semi-supervised approach) for training the model.

The hierarchical approach has two classifications, first in Dynamic and Static if the activities involve motion or not. Secondly in Dynamic we have walking and running and in Static standing idle and sitting.

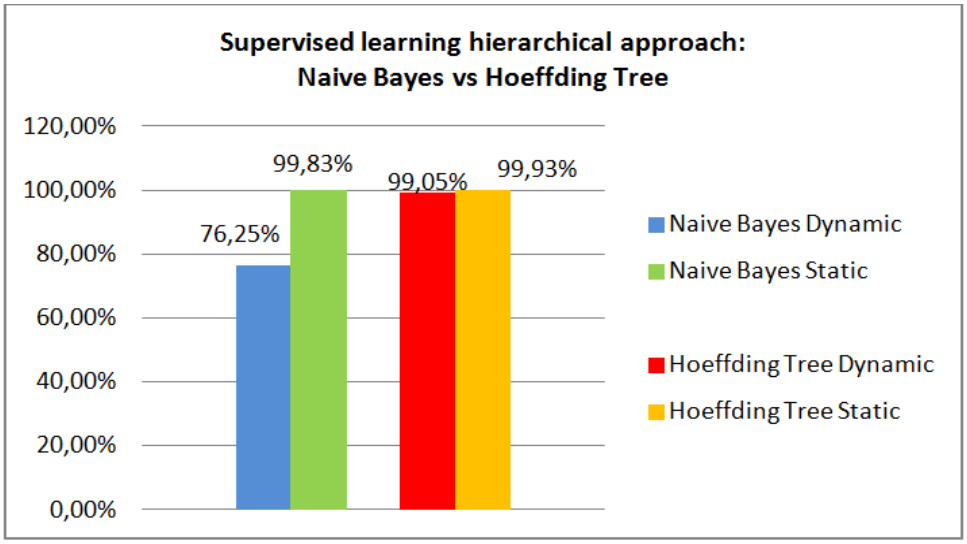


Figure 37 Final classifiers' accuracy using hierarchical supervised classification

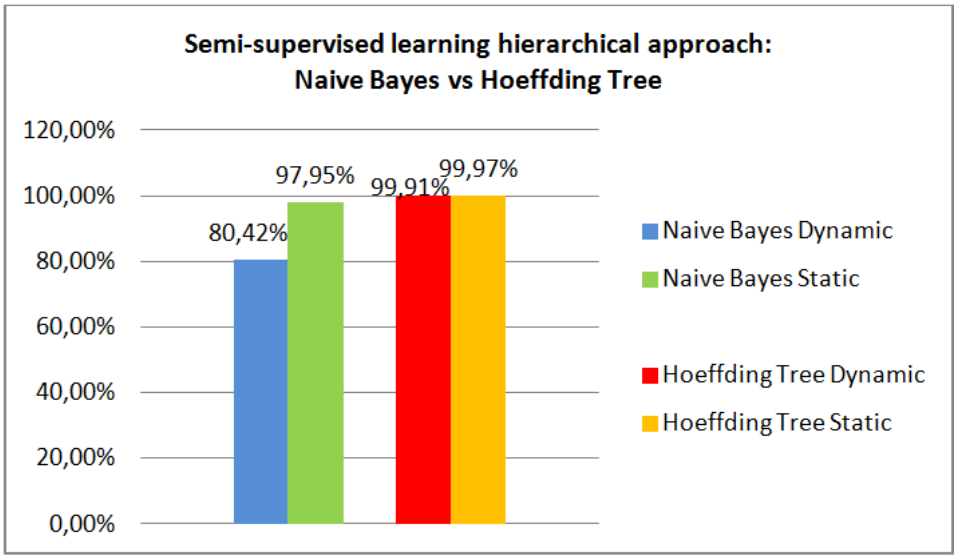


Figure 38 Classifiers' accuracy on final step of hierarchical classification with a semi-supervised learning approach.

After doing the hierarchical classification (Figure 37 and 38) the labeled data was checked by visual inspection and it was easy to observe that Hoeffding Tree tend to label data on the first classification as Dynamic (probably because the dataset is unbalanced and the dynamic class is the majority one). Naïve Bayes seems more balanced when labeling new data in the first classification of the hierarchical approach.

At last we tested how using the two classifiers together would affect the classification (Figure 39).

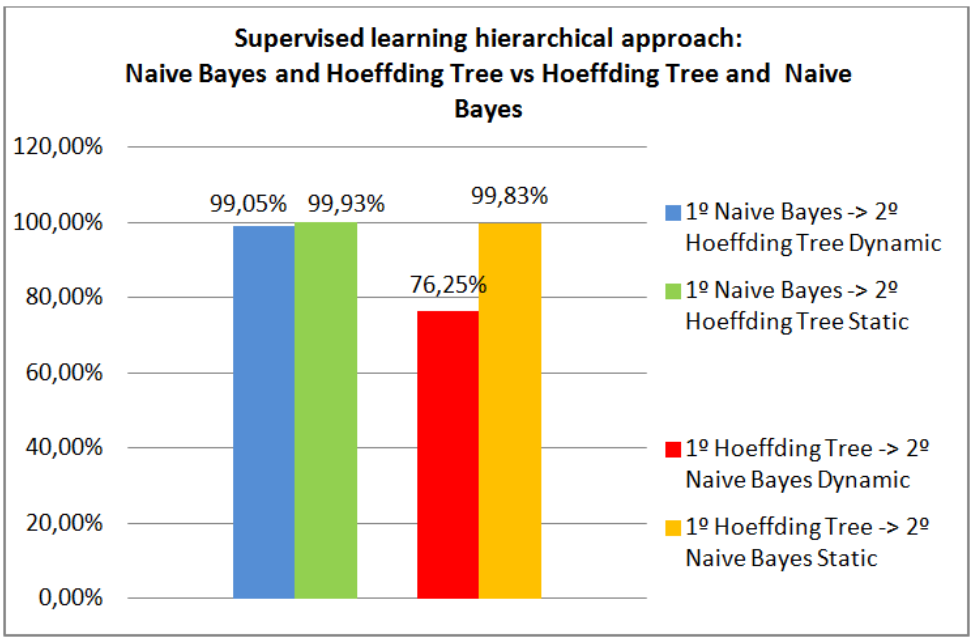


Figure 39 Final classifiers' accuracy using hierarchical supervised classification with different classifiers for each step.

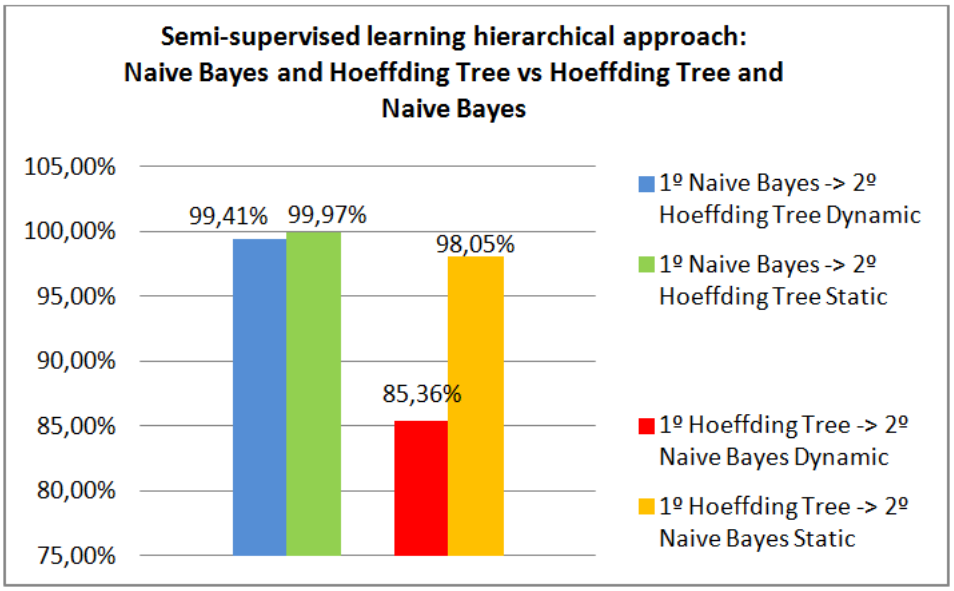


Figure 40 Final classifiers' accuracy using hierarchical semi-supervised classification with different classifiers for each step

The balance characteristic of Naïve Bayes mentioned before can be verified on Figure 40, giving better results when used in the first classification. The tendency of Hoeffding Trees to classify, in the first step, the data as a Dynamic movement has influence on the second classification where Naïve Bayes has difficulties to label data because it gets lots of Static labeled data as Dynamic data from the first step. Overall better accuracy is achieved when using the Naïve Bayes classifier on the first classification (Dynamic or Static movement) and Hoeffding Tree on the second classification.

The accuracy is not the only indicator of the classifiers' performance. The precision and recall are also important.

Precision: This is the percentage of retrieved documents that are in fact relevant to the query.

Recall: This is the percentage of documents that are relevant to the query and were in fact, retrieved.

It is common to plot a graph of precisions at many different levels of recall; a higher curve represents a better-quality information retrieval system [25].

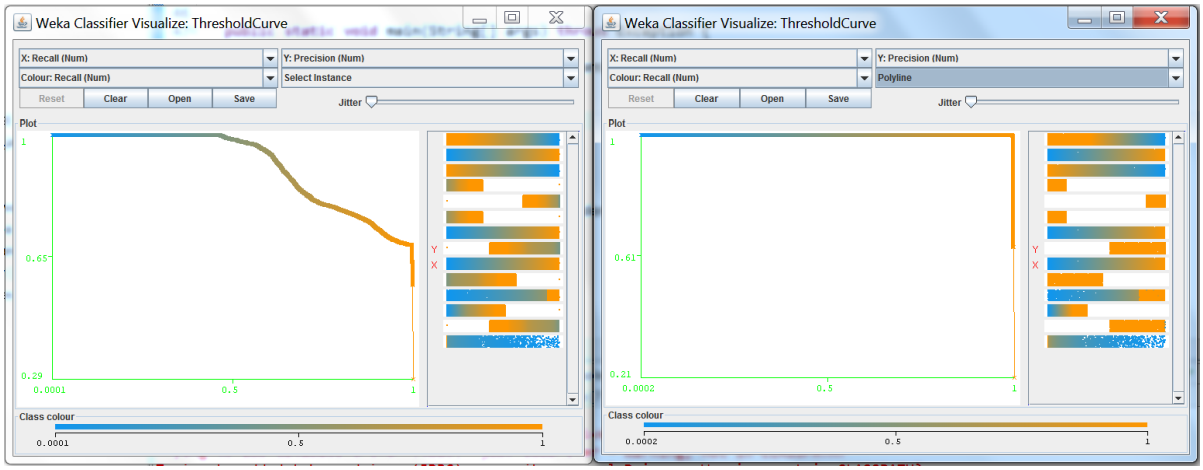


Figure 41 Precision/recall graph of Naïve Bayes used in one-step classification. Semi-supervised learning (left), supervised learning (right)

In this case it is possible to see that with the semi-supervised learning approach the precision tends to drop when the recall goes to a maximum, what means that during the process of training the model we are getting a small amount of relevant information but the majority of that information is being useful (Figure 41).

This also happens in other cases but in others to have a semi-supervised or a supervised learning is almost the same, like in Figure 42.

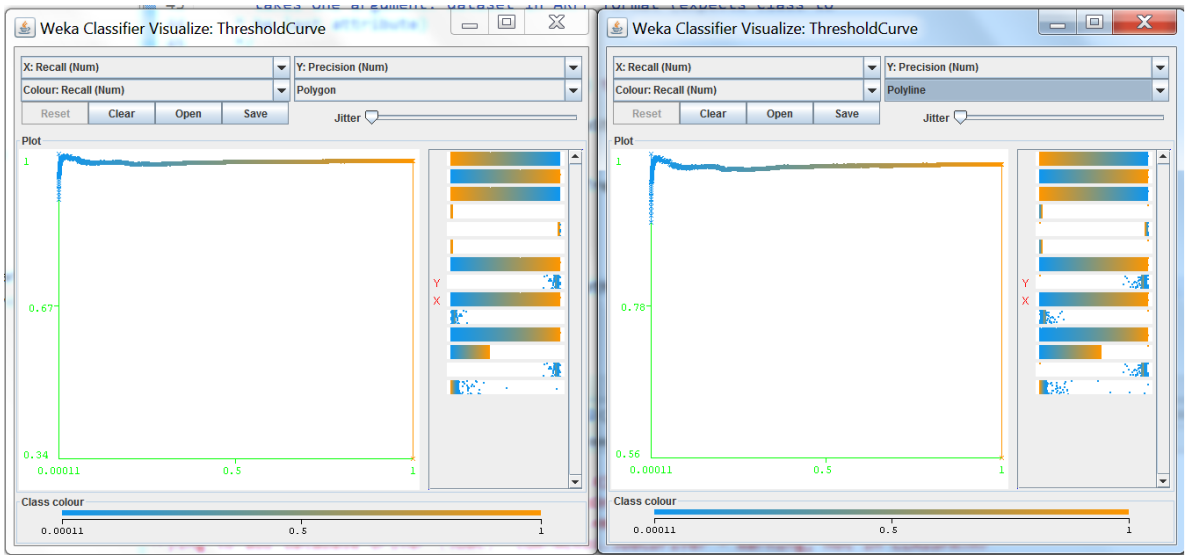


Figure 42 Precision/recall graph of Hoeffding Tree used in second classification (Hoeffding Tree also used in first classification) of the hierarchical approach (Running and Walking). Semi-supervised learning (left), supervised learning (right)

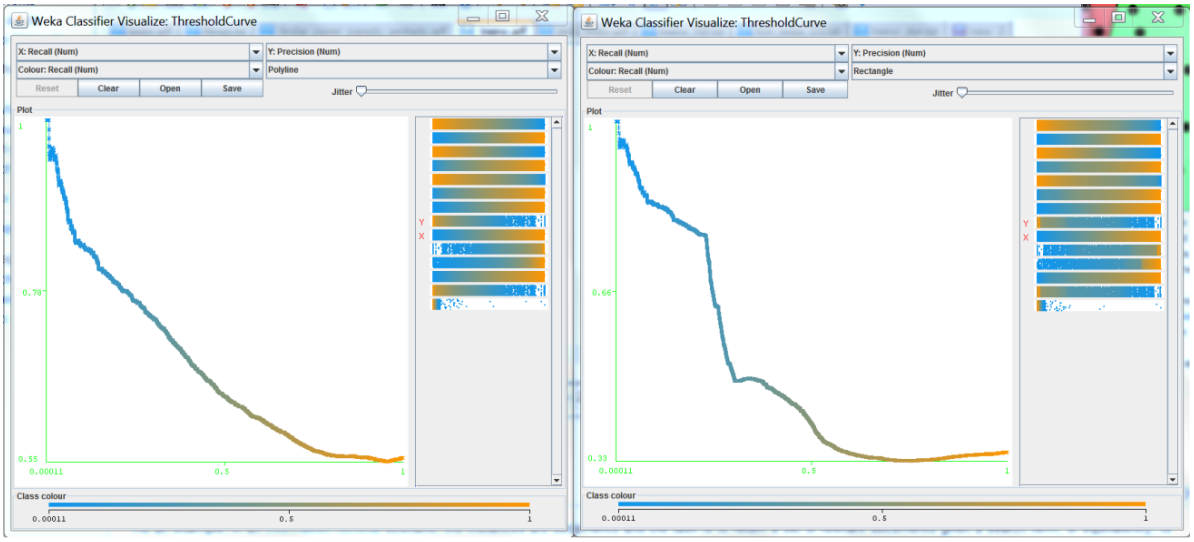


Figure 43 Precision/recall graph of Naïve Bayes used in second classification (Naïve Bayes also used in first classification) of the hierarchical approach (Running and Walking). Semi-supervised learning (left), supervised learning (right)

This shows that in both cases the model tends to have a small variation.

The different graphs prove that although a technique presents more accuracy that does not mean that the information retrieval system is better.

Another thing that we noticed is that the information retrieval system of Hoeffding Trees is better than Naïve Bayes maybe because it does not treat the data as unrelated. That can be seen by the higher curve that these methods present (Figure 42 and 43)

Only some precision/recall graphs are here to show what did happen during the experiments and to make it easier to explain. The rest of the graphs are on the attachments.

The application had, also, concerns about the battery usage and memory.

In order to test the battery usage we created a stress situation where the app did both the hierarchical classification and the one step classification. In order to do it two models were created using the data of about 43.000 lines of labeled data, and doing the classification of ten unlabeled instances (in reality it only does one instance every 16s).

This experiment told us that the usage of battery needs a maximum of 600.0mW for the CPU and between 500mW and 600mW for the LCD, which gives a total between 1100 and 1200mW on hierarchical classification – Figure 44.

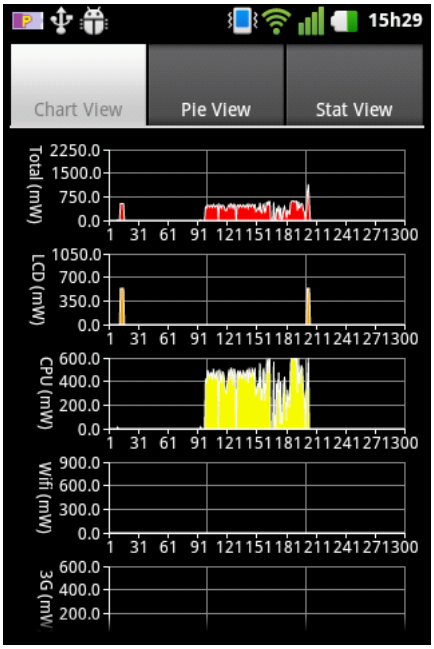


Figure 44 Power used by Hierarchical approach with Naïve Bayes

The one step classification only creates one model. The battery usage needs a maximum of 526mW for the CPU, the LCD needs the same power as the hierarchical approach, of course. Running the application five times, in a row, we got an energy usage of 120.8J for the CPU in hierarchical classification. However in one step classification we get a total of 110.3J – Figure 45.

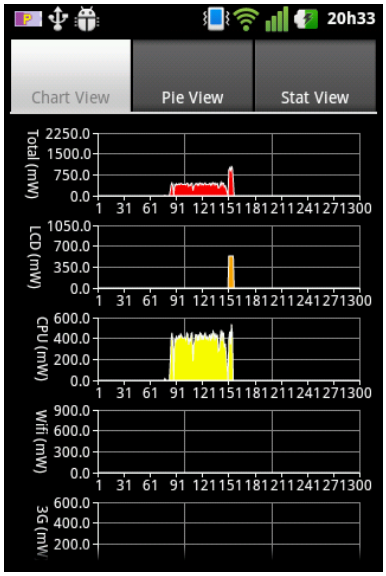


Figure 45 Power used by One step approach with Naïve Bayes

Creating models and classifying multiple instances takes almost 60s.

And it is easier to check the difference of the CPU usage from hierarchical classification to one step classification.

Creating models and classifying multiple instances takes almost 60s.

In terms of memory, the prototype is about 3Mb, and the files used for training the model having about 43000 lines are 1.466kB each. These files will grow until the limit defined by the sequenc-based window is reached because new labeled data from the user will be added (like aforementioned if the certainty of the classification is bigger than 70%) in order to have a semi-supervised learning approach. When the limit is reached the old data will be replaced by the recent one. We will have three files for training (hierarchical approach).

Capítulo 5

Conclusions and Future Work

The encouraging results of the experiments lead us to affirm that a step forward has been taken in the study of activity classification and that the project objective stated in the beginning has been achieved with success.

To achieve good results, the techniques do not need to be too complex, like it was shown using Naïve Bayes. A fair conclusion after analyzing the Figures 41 and 42 is that hierarchical approach gives better results with Naïve Bayes doing the first classification and Hoeffding Tree dealing with the final one.

With less complex techniques less power of the mobile is needed, leading to a minor impact on the performance of the mobile. So, if Naïve Bayes does not decrease the accuracy it is better to use it in order to save memory and battery.

The battery usage confirms that the app can be used non-stop.

It would be thrilling and of greater convenience to create a way that could swap classification techniques when the battery was low so it could be saved and the application did not have to stop. Changing from a hierarchical classification to one step would have not a big impact on the final results as we have seen on the Figures 41 and 42 where the accuracy dropped maximum of 2%, where Hoeffding tree provided the better results.

The model used only has to be created when the application starts working. It is used for classifying until the app is shut down. It has only to classify one instance every 16s which is enough to do it, so the duty cycles work perfectly.

Taking into account that the better results are achieved with the hierarchical approach with semi-supervised learning and that the model is trained every time we start the application it would be important to have a variable that had the total of instances labeled with a certainty equal or superior to the 70% defined by us and added to the training file. A limited of instances

could be created and when the variable with the total of new instances was equal to the limit a new model would be created. Another thing that can improve the application is when the majority of the instances is being classified with a certainty above the threshold defined it can be changed to a higher value in order to try to improve the model and therefore the accuracy of it. A process to control the addition of the new labeled data to the training file should be considered, because it is important to keep the data balanced. As we have seen in the tests above unbalanced data can induce the model into tending to classify unlabeled data with a determinate class leading to errors that can only be spotted on a manual revision of the data.

This would not have memory implication because regarding the memory usage a limit on the training files can be created, when this limit is reached the older data can be erased and new data added. This allows the adaptation of the application to new users as long as the application is being used by these new users.

The application can be improved by making possible to wear the mobile on other location, for example on women purses and testing other classifiers and how to process data, in order to run faster. Even with the existing power of the processors on the smartphones like quad core it is possible to conduct new experiments like if we were working on a computer.

New tests can be made using data from people with mobility constraints. Improving the application so it can adapt to this kind of people can be important if an accurate prediction can be made. Studies of patients with diseases that tend to degrade the ability to move can be accomplished to prevent, for example, falls or just to study how the movements change. This prevention can also be applied to elder people.

A better understanding of different movements can be acquainted.

With this knowledge, people who practice sports can also benefit. For example, understanding how their body posture can be corrected in order to achieve better results is like training with a personal trainer that is always helping to prevent injuries. Also measuring the actual time that an activity was being performed can be used in endurance training for example. Knowing that we spent 45 minutes it is different than going for a run of one hour actually just running 30 minutes and spend the other 30 minutes walking on a fast pace.

This is just the beginning of an application that can be expanded in order to provide a better intimate experience between users and mobile phones.

References

[1] P. Domingos & G. Hulten, (2000). Mining high-speed data streams. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*, 71-80. New York, New York, USA: ACM Press. doi:10.1145/347090.347107

[2] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity Recognition using Cell Phone Accelerometers," *SIGKDD Explorations*, vol. 12, no. 2, pp. 74-82, 2010.

[3] L. Bao & S. S. Intille (2004). "Activity Recognition from User-Annotated Acceleration Data, "1-17.

[4] T. Gu, Z. Wu, X. Tao, H. K. Pung, and J. Lu. epSICAR: An Emerging Patterns based Approach to Sequential, Interleaved and Concurrent Activity Recognition. In Proc. of the 7th Annual IEEE International Conference on Pervasive Computing and Communications (Percom '09), Galveston, Texas, March 9–13, 2009.

[5] X. Zhu (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison

[6] D. Guan, W. Yuan, Y.-K. Lee, A. Gavrilov, and S. Lee, "Activity Recognition Based on Semi-supervised Learning," *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*, no. 1, pp. 469-475, Aug. 2007.

[7] J. Lester and T. Choudhury, "A practical approach to recognizing physical activities," *Pervasive Computing*, pp. 1-16, 2006.

[8] N. Ravi, N. Dandekar, P. Mysore, ML Littman (2005) "Activity recognition from accelerometer data". In: Proc. of the seventeenth conf. on innovative applications of artificial intelligence (IAAI). AAAI, Menlo Park, pp 1541-1546.

[9] J. Gama, "Aprendizagem Bayesiana Introdução." pp. 1-22, Universidade do Porto

[10] J.Gama," Knowledge Discovery from Data Strems", CRC Press, 2010

[11] P. L. Lanzi, "Decision Trees ", Politêcnico di Milano.

[12] G. Holmes,& K. Richard (2005). "Tie-breaking in Hoeffding trees"

References

[13] J. Catlett. Megainduction: Machine Learning on Very Large Databases. PhD thesis, Basser department of Computer Science, University of Sydney, Sydney, Australia, 1991.

[14] A. C. F. Coster, "Classification of basic daily movements using a triaxial accelerometer," *Medical & Biological Engineering*, pp. 679-687, 2000.

[15] M. Schneider, M. Velten, and J. Haupert, "The ObjectRules Framework - Providing Ad Hoc Context-Dependent Assistance in Dynamic Environments," *2010 Sixth International Conference on Intelligent Environments*, pp. 122-127, Jul. 2010.

[16] Ovidiu Ivanciuc. (2005). SVM - Support Vector Machines. http://www.support-vector-machines.org/

[17] N. S. Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, "A Framework of Energy Efficient Mobile Sensing for automatic user state recognition.pdf," pp. 179-191, 2009.

[18] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*, Second. Morgan Kaufmann, 2011.

[19] B. Babcock and M. Datar, "Sampling from a moving window over streaming data," *of the thirteenth annual ACM-SIAM*, 2002.

[20] G. Bieber, J. Voskamp, and B. Urban, "Activity Recognition for Everyday Life on Mobile Phones," *Universal Access in HCI, Part II, HCII 2009, LNCS 5615*, pp. 289-296, 2009.

[21] MOA Team. (2012). MOA Massive Online Analysis. http://moa.cs.waikato.ac.nz/

[22] Google. Android developers. http://developer.android.com/index.html

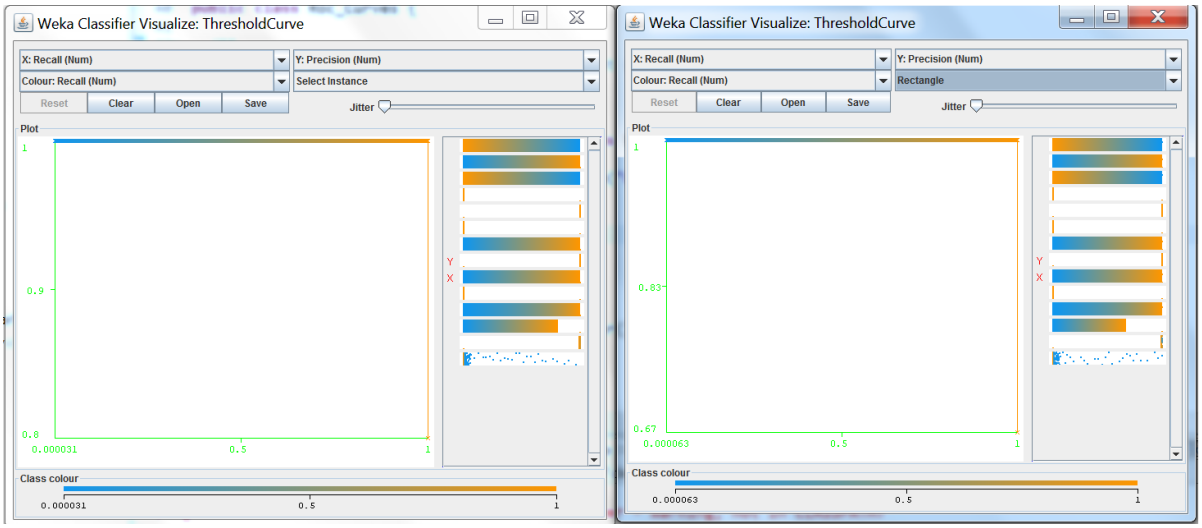
[23] University of Michigan. Power tutor. http://ziyang.eecs.umich.edu/projects/powertutor/

[24] S. Sprager and D. Zazula, “A cumulant-based method for gait identification using accelerometer data with principal component analysis and support vector machine,” *WSEAS Transactions on Signal Processing*, vol. 5, no. 11, pp. 369–378, 2009

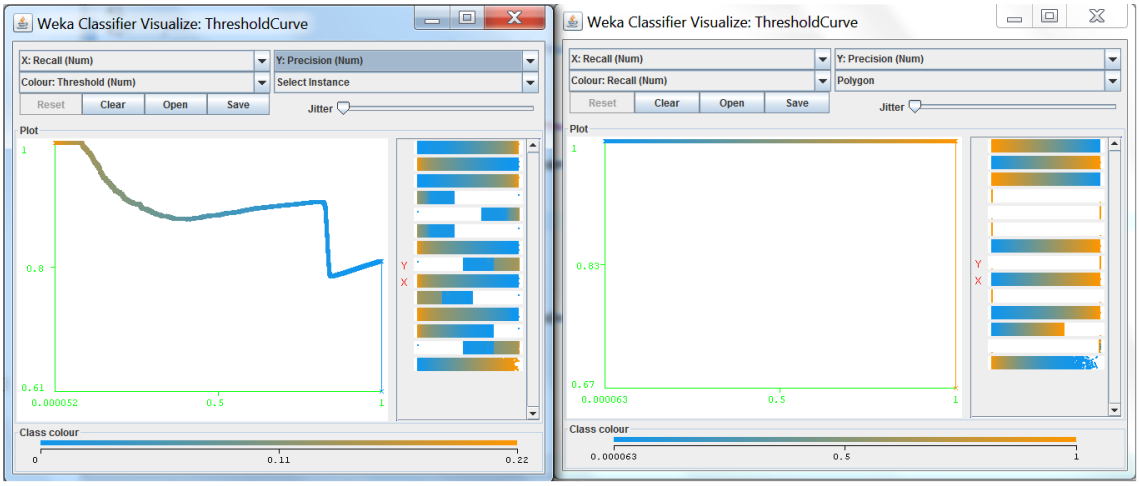
[25] H. Jiawei (2001). Data mining: concepts and techniques. *San Francisco, CA, itd: Morgan Kaufmann*. Morgan Kaufmann

Appendix A

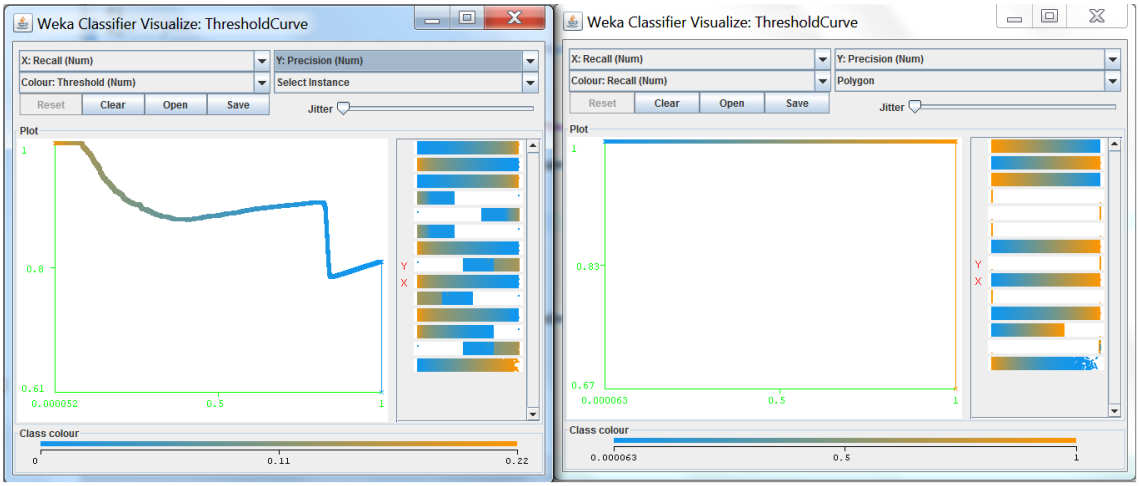
Precision/Recall graphs



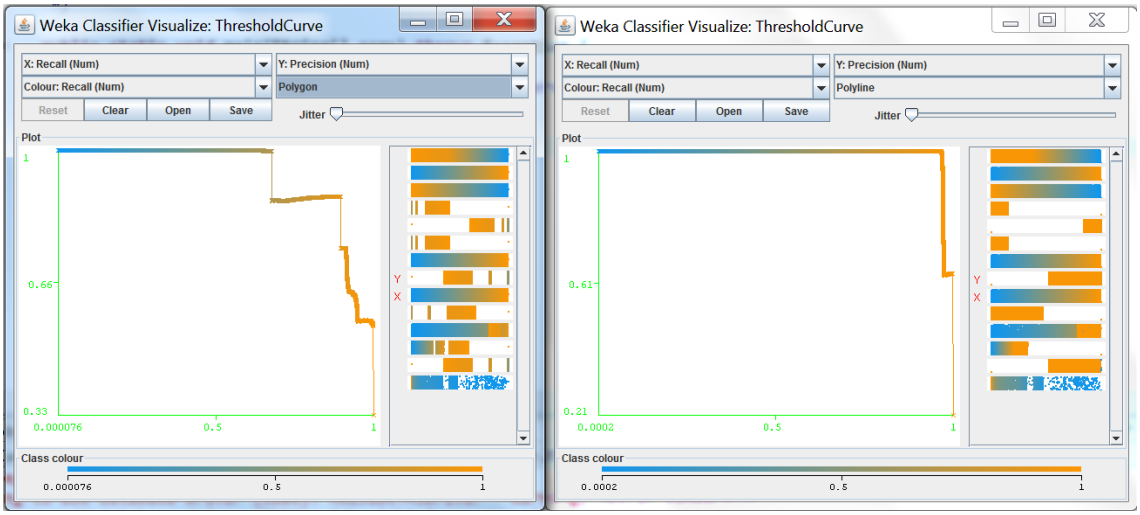
Att. Figure 1 Precision/recall graph of Naïve Bayes used in second classification of the hierarchical approach (Static). Semi-supervised learning (left), supervised learning (right)



Att. Figure 2 Precision/recall graph of Hoeffding Trees used in first classification of the hierarchical approach (Dynamic and Static). Semi-supervised learning (left), supervised learning (right)



Att. Figure 3 Precision/recall graph of Naïve Bayes used in first classification of the hierarchical approach (Dynamic and Static). Semi-supervised learning (left), supervised learning (right)



Att. Figure 4 Precision/recall graph of Hoeffding Trees used in one-step classification. Semi-supervised learning (left), supervised learning (right)