FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

On Making Feasible Smartphone-based Human Activity Recognition

Francisco Miguel Lamares Martins Barbosa

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Professor Doutor João Manuel Paiva Cardoso

July 3, 2019

On Making Feasible Smartphone-based Human Activity Recognition

Francisco Miguel Lamares Martins Barbosa

Mestrado Integrado em Engenharia Informática e Computação

July 3, 2019

Abstract

In the last few years, we witnessed a massive evolution in smartphones at, practically, all levels. As a result, their processing power has increased and is making possible the execution of more complex applications. This dissertation aims to contribute to the feasibility of implementing a machine-learning-based Human Activity Recognition (HAR) systems in a smartphone.

HAR systems can be of high importance for daily life, from recommendations, monitoring, context-aware applications, to emergency situations. We focus our analysis of smartphone-based HAR systems, in several factors that may interfere with the system performance (performance of the classifier, classification time and energy consumption). This dissertation presents the impact of the number of features used via feature selection techniques (Cfs and ReliefF), a comparison of performance between kNN, Naive Bayes, Hoeffding tree and an ensemble using the three classifiers. We also present the impact of using simple rules or a binary Hoeffding tree to avoid the use of the more complex classifiers in some circumstances, and the impact of using a top layer with the intention to correct some predicted activities based on transitions between activities.

The proposed approaches proved to be a success, allowing to keep an acceptable accuracy and reduce the computational cost of the system. It was possible to achieve better accuracies with a lower number of features, the rule-based classifier lowered significantly the energy consumed and the filter improved the accuracy in almost every experiment.

Resumo

Nos últimos anos, verificou-se uma grande evolução nos smartphones a praticamente todos os níveis. Como resultado, o seu poder de processamento aumentou, possibilitando a execução de aplicações mais complexas. Esta dissertação visa contribuir para a viabilidade da implementação de um sistema de reconhecimento de atividade humana (HAR) num smartphone, através de técnicas de *Machine Learning*.

Os sistemas HAR podem ser de grande importância para a vida quotidiana, desde alterações no estilo de vida, aplicações *context-aware*, até situações de emergência. Focamos a nossa análise na implementação de sistemas HAR para smartphones, nos vários fatores que podem interferir com o desempenho do sistema (precisão do classificador, tempo de classificação e consumo de energia). Esta dissertação apresenta o impacto do número de *features* obtidas utilizando técnicas de *feature selection* (Cfs e ReliefF), uma comparação de desempenho entre os classificadores kNN, Naive Bayes, Hoeffding tree e um Ensemble com os três classificadores. Apresentamos, também o impacto do uso de regras simples ou de uma Hoeffding tree binária para evitar o uso de classificadores mais complexos em algumas circunstâncias e o impacto do uso de uma camada pós-classificação com a intenção de corrigir as atividades já classificadas com base nas transições entre atividades.

As abordagens propostas provaram ser um sucesso, permitindo manter uma precisão aceitável e reduzir o custo computacional do sistema. Foi possível obter melhores precisões com um número menor de *features*, o classificador baseado em regras diminuiu significativamente a energia consumida e o filtro melhorou a precisão em quase todas as experiencias.

Acknowledgements

I would like to thank my parents and brothers, for all the patience and encouragement all these years, always helping me getting better results.

I also thank my girlfriend, Sandra Silva, for all the support, always being proud and believing in me.

I am also grateful to my friends that have always been my by side, helping me stay focused.

Finally, this dissertation could not be achieved without the support of the members of SPECS laboratory and especially without my supervisor, João Cardoso, that guided me throughout this dissertation.

Francisco Barbosa

"If plan A doesn't work, the alphabet has 25 more letters"

Claire Cook

Contents

List of Figures

LIST OF FIGURES

Acronymous

ANN	Artificial Neural Network
ARFF	Attribute-Relation File Format
CFS	Correlation-based Feature Selection
DA	Discriminative Analysis
DBN	Deep Belief Network
DEI	Departamento de Engenharia Informática da FEUP
DT	Decision Tree
FEUP	Faculdade de Engenharia da Universidade do Porto
FFT	Fast Fourier Tranform
GMM	Gaussian Mixture Model
GPS	Global Positioning System
HAR	Human Activity Recognition
HMM	Hidden Markov Model
IDE	Integrated Development Environment
INESC TEC	Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência
JB	Join Boosting
kNN	k-Nearest Neighbours
KPCA	Kernel Principal Component Analysis
NB	Naive Bayes classifier
NN	Neural Network
PCA	Principal Component Analysis
RF	Random Forest
SLGMM	Supervised Learning Gaussian Mixture Model
SVM	Support Vector Machine

Chapter 1

Introduction

Human Activity Recognition (HAR) is a field responsible for recognizing a human activity, performed by a real user. In a simple way, a HAR system consists of gathering information with the help of sensors and then process this information using machine learning techniques to recognize human activity [HUMA18]. This dissertation focus on studying methods and techniques to make feasible the implementation of this kind of system on a smartphone.

1.1 Context

This dissertation is related to the CONTEXTWA project [IT] developed through the partnership between INESC TEC and FEUP. The CONTEXTWA project focuses on the development of context-aware mobile applications, through the implementation of new algorithms, techniques and technologies, combining the areas of signal processing, machine learning, embedded, mobile and cloud computing [CMV18]. So, the aim of the project is to be able to detect human activity in real-time efficiently.

Thus, the present dissertation is related to the CONTEXTWA project, having some areas and some problems in common. Also, the research and technological advances made during it are extremely important for the development of this dissertation.

1.2 Motivation and goals

The outcomes of this project may be of extreme importance to society. There are several situations where real-time HAR systems are fundamental, such as situations of emergency, like fall detection in children or elderly people [GCM+18] or more lifestyle related issues through the detection of patterns, sedentary lifestyle and lack of sleep [CMV18]. Recent researches correlates sedentary behaviors with cardiovascular diseases, elevated risk of diabetes, obesity and other causes of mortality [SAL+16]. Thus, a HAR system can advise people to replace their lazy lifestyle and help preventing chronic diseases. It can also be crucial for the evolution of context-aware applications and services.

Although current HAR systems can achieve good accuracy, they still face energy consumption issues, making real-time implementations on smartphone impracticable. Therefore, the main goal of this dissertation is to develop new methods and techniques for making feasible a smartphone-based HAR prototype. These techniques intend to increase the performance of an existent HAR system, by decreasing the classification time, the energy consumption and keep an acceptable accuracy. Initially, the measurements of accuracy were made on a desktop application, the classification time was tested on a mobile application (offline version) and the energy measurements were made on a ODROID-XU board. To be able to build this system, it is required to focus on the issues related to the practical implementation in the smartphone, based on the knowledge and progress made by members of DEI in some previous projects.

Thus, the prototype was developed with the help of the techniques and algorithms already developed during these projects and the necessary improvements to achieve the primary goal.

1.3 Problem

The problems of this dissertation arise in the area of data preprocessing and machine learning techniques used to improve classifiers performance. This problem is due to the HAR system being exclusively implemented in a smartphone and, as it is known, the mobile devices have a lower processing power compared to a desktop. Energy consumption is another challenge that comes up in the smartphone context. The literature studies the state of the sensors used and their position, and in the case of wearable sensors, it is also a challenge since it can affect the accuracy of HAR, although this problem is not focused in this dissertation. Other factors can influence the performance of a HAR system, such as the classification models, the sliding window size and the number and quality of the features used in the preprocessing phase [GCM+18]. Another problem is related to the diversity of human activity, requiring a careful selection of sensors that have different characteristics and capabilities. Thus, it is essential to understand and define the activities to be recognised, but there is a need to deal with the fact that one activity can be performed in many different ways [BBS13].

In an attempt to discover possible solutions to the stated problems, several articles have been studied that address the implementation of HAR systems in smartphone and desktop. The analysis of these articles is presented in the State of the Art section.

1.4 Contribution

The project developed aims to contribute to the area of mobile computing and machine learning techniques for HAR systems. Specifically, it studies existent techniques and introduces novel methods that simplify the classification step and the HAR system in general. Thus, the main goal of this dissertation is to test the following hypothesis:

1. Restricting transitions between activities improves the accuracy of the HAR system;

2. The reduction of the number of features of the system leads to a significantly decrease of the classification time and energy consumption, keeping an acceptable accuracy;

3. The implementation of a rule-based classifier decreases the classification time and energy consumption, and keeps an acceptable accuracy.

1.5 Structure of the dissertation

This dissertation is structured as follows.

Chapter 2 presents the background of a general HAR system, describing the common steps, concepts and algorithms necessary to understand the project.

Then in Chapter 3, it is shown the literature review, the common problems of a HAR system and approaches to solve them. This Chapter focus on comparing the most used methods and algorithms, the number of features and which features used for obtaining a better performance (high accuracy, low classification time and energy consumption).

Chapter 4 starts by presenting the tools used to developed this dissertation and the base HAR implementation. Then it explains the additions and improvements made to the base system in order to prove the hypothesis stated above.

Chapter 5 demonstrates the details of all the experiments, their results and the conclusions for each experiment.

Finally, Chapter 6 presents the final conclusions about the impact of the techniques used and some suggestions and possible improvements for the future work.

Chapter 2

Background

In this chapter, some of the fundamental concepts are explained in order to better understand the project and what has been achieved in this field. Firstly, it is given the common architecture of HAR systems and then their typical stages: Data Acquisition, Data Pre-processing and machine learning techniques. The structure of a HAR system commonly follows the structure of Figure 2.1.

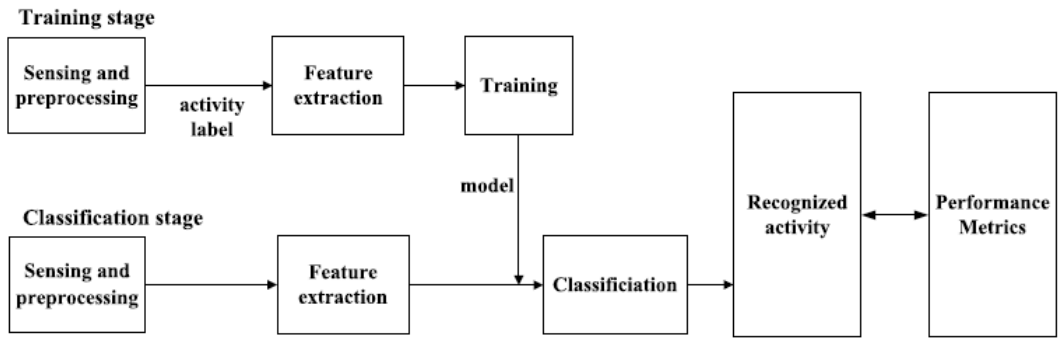

Figure 2.1: Typical Activity Recognition Chain.
Source: [CWZ+17]

2.1 Human Activity Recognition (HAR) Systems

Human Activity Recognition (HAR) systems [AT14] are systems capable of detecting human activity, such as walking, sitting, sleeping, among others. HAR systems can be crucial in real-world applications as monitoring daily activity to support medical diagnosis for rehabilitation or to encourage people to adopt a healthier lifestyle by tracking all their routines. Lately, these systems have been used in game consoles, Nintendo Wii and Microsoft Kinect [BBS13].

2.2 Data Acquisition

Data acquisition is the process of obtaining physical world information, usually done through the use of sensors.

2.2.1 Sensors

In this dissertation, the focus is given to smartphone built-in sensors, such as accelerometers, gyroscope, magnetometer, GPS, among others. Wearable sensors are also important in this study, and they consist of wearable accessories with sensors that can be worn by humans. It is also relevant to mention that these sensors are capable of physiological, biochemical and motion sensing [XFYTCP08]. These devices have some technological limitations: battery duration and lifetime, the ease of use and the fact that they need to be comfortable for the user since one of the goals is to collect data from daily activities during long periods. Another problem arises with the sensors placement, namely, the locations where the wearable devices are placed and how they are attached to the human body. The previous problem has a direct impact on the data measured, and later it can interfere with the accuracy of HAR [AMD+15].

Accelerometer

The signals measured by accelerometer are composed by a gravity and motion component [MA17]. This sensor is capable of measuring the acceleration and the tilt angle of the device for the different axes, x, y and z. This way, it is possible to detect the movement speed and direction of the device as well as collisions. This sensor can also be used to record the number of steps when performing some activities like walking or running. [PGY+11] The literature demonstrates that accelerometer is the most studied sensor and the one with the most impact on the activity recognition [MA17].

Gyroscope

The signals obtained by gyroscope allows the measurement of device rotation for the different axes, x, y and z [SAL+16].

Magnetometer

The signals obtained by magnetometer allows the measurement of the ambient geomagnetic field for the different axes, x, y and z [SAL+16].

2.3 Data Pre-processing

Among the raw data obtained in the data acquisition stage, there is unreliable data that can result in out of range values, impossible combinations, missing values. This noisy data negatively affects the performance of HAR, so the noise and inconsistent data need to be removed. Pre-processing not only involves noise reduction but also, data reduction, performed in feature extraction and

selection stages. This stage prepares the data ideally before the classification algorithms in order to increase the effectiveness and accuracy of the machine learning algorithms during classification [GLH14].

2.3.1 Data Segmentation

Data segmentation is responsible for identifying segments of data streams, defined by a start time *t1* and an end time *t2*, that are expected to have crucial information about the activities [SAL+16]. This information is useful for the classification step but can also play an important role for other purposes, for example, when no activity is sensed, the activity recognition can be turned off to save power [BBS13]. One of the most important challenges to deal in this step is to achieve a proper division of the raw data into the best possible set of segments for the activity recognition, since each segment corresponds to an instance of an activity and their exact boundaries are hard to delimit [NPCN16].

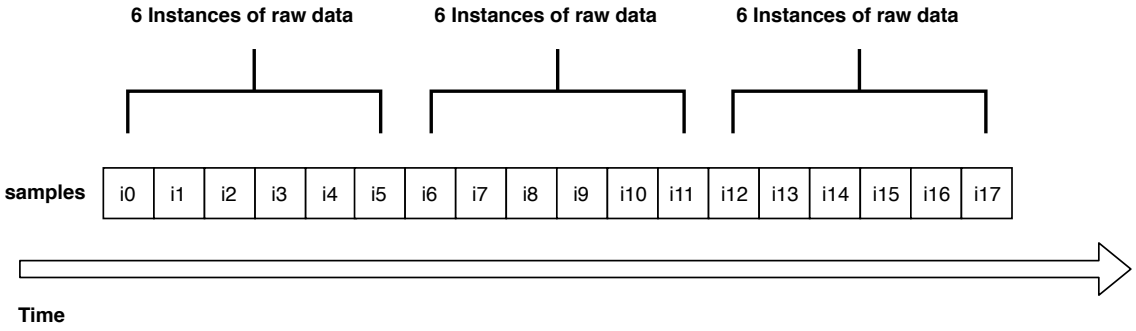

Figure 2.2: Example of data Segmentation in windows of 6 instances.

Data Overlapping

Data overlapping in this scenario can be translated in windows of data that use instances from the previous windows. For example, Figure 2.3 shows an example of data overlapping of 20% using windows with 5 samples of length.

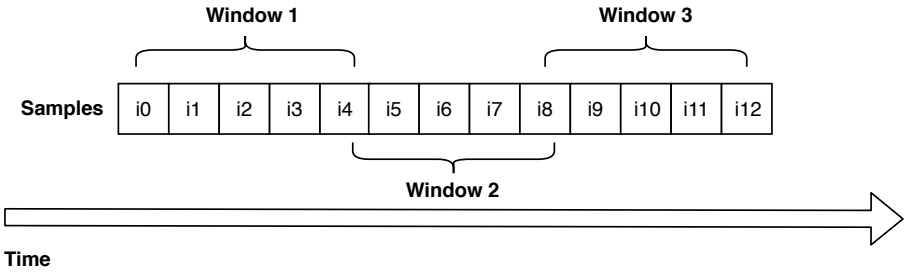

Figure 2.3: Example of 20% of data overlapping between windows with 5 samples of length.

2.3.2 Feature Selection and Extraction

Feature selection consists in choosing a subset of features, based on its purpose. The goal of this process is to recognize important features in the whole data set and discard the irrelevant ones. Feature extraction is the process that extract new features as a function of the original ones. These processes can improve the learning efficiency, allowing to reduce the computational cost [GLH14]. Later, the vector of features resultant of these methods is used during the learning phase. The sensor signals can be classified in three different domains: time domain, frequency domain and discrete representation domains, although in this dissertation the only one studied is the time domain. In Figure 2.4 are presented some techniques that are usually applied in each domain for feature extraction.

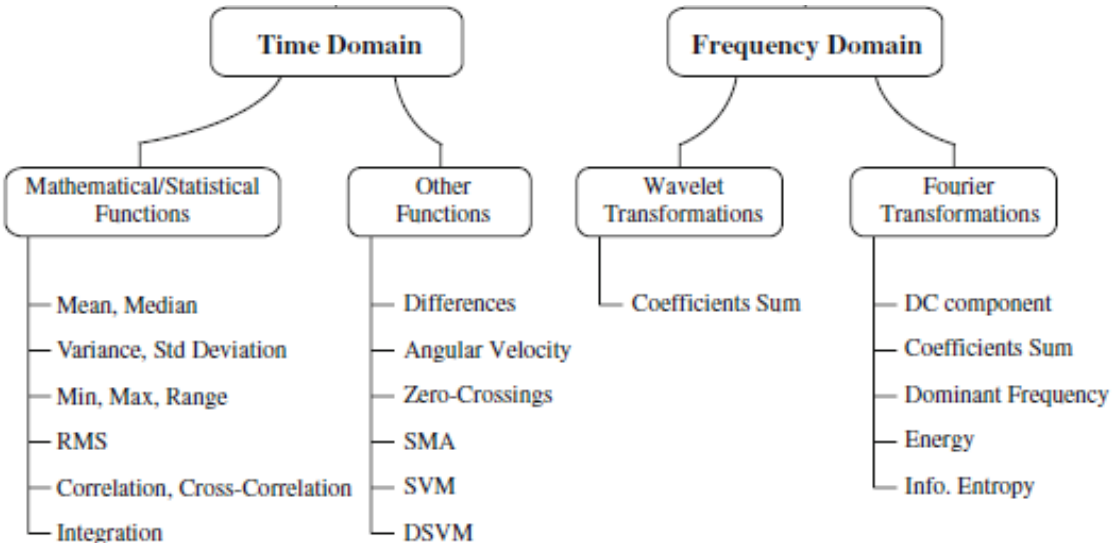

Figure 2.4: Feature Domains.
Source: adapted from [FDFC10]

2.4 Machine Learning

Arthur Samuel described the term Machine Learning as "the field of study that gives computers the ability to learn without being explicitly programmed" [Sam59]. This is an older, informal definition. Tom Mitchell provides a more modern definition in [Mit97]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." We start by defining a mathematical model with some parameters, and the learning process is the execution of a computer program to optimize those parameters based on training data. This model is based on the theory of statistics, and its main goal is to make an inference from a sample [Alp10]. In this dissertation, the learning process studied is supervised learning.

2.4.1 Supervised Learning

In this learning process, we have our training set, used to train the model, and we know its correct output. Thus, there is a relation between the input and output. In supervised learning category there are two classical problems: classification and regression. In classification, there is a specific number of classes and categories used to predict a sample. These classes and categories are known by the learning algorithm and the unseen examples must be assigned with one of these classes, after the training of the model [GLH14]. In regression problems we have a training phase like in classification, where we have knowledge of the inputs and outputs, but here we are dealing with infinite values, and the output variables take continuous values [Alp10]. Summarizing, in classification we try to predict results in a discrete output, while in regression we need to map input variables to a certain continuous function.

2.4.2 Unsupervised Learning

In unsupervised learning we only have knowledge of the input data, and no information about the output, so the aim is to find regularities in the input, namely, to find patterns that happen more often than others. Two of the main methods used in this type of learning are called clustering and non-clustering. The first ones consist in finding clusters or groups of input data and in the second ones we need to find a structure in a chaotic environment [GLH14].

2.4.3 Online Learning vs Offline Learning

Offline learning uses a static data set to train the predictive model. After the generation of the model this is loaded by the classifier to make the predictions. On the other hand, online learning updates continuously the model with the new collected data and there's no need to retrain the model as it should be done in offline learning.

2.4.4 Machine Learning Algorithms

In order to implement a HAR system with success, it is extremely important to choose the right algorithm to classify the activities. The most typical machine learning algorithms used for this type of systems are: Support Vector Machine, k-Nearest Neighbor, Hidden Markov Model, Naive Bayes and Decision Trees.

Support Vector Machine

SVM [BI] is a supervised learning approach for linear classification and regression problems. This algorithm consists in finding an optimal hyperplane, used to separate patterns, with the maximum distance between the data points of both side. The data points close to that hyperplane are called support vectors and they are the most difficult to classify. The decision function is based on the support vectors.

k-Nearest Neighbor

kNN [SSBD14] is a simple algorithm and consists in predicting the label of a new instance after memorizing the training set. This prediction is based on the labels of the K closest neighbors, in the training set, of that instance.

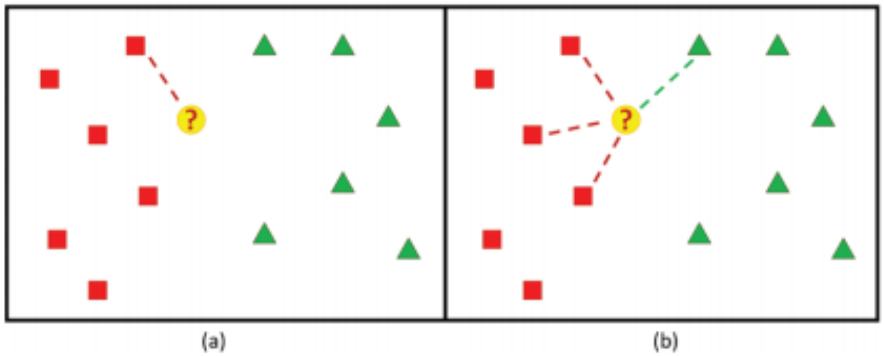

Figure 2.5: (a) The kNN with k = 1 checks the closest neighbor. The point "?" is classified with the same label of the red class since the closest point is red. (b) The kNN with k = 4 checks the 4 closest neighbors, 3 are red and 1 is green so the point "?" is assigned to the red class.
Source: [MPP09]

Hidden Markov Model

A Markov chain [JM] is a model related to probabilities of sequences of states and each state can be associated with a value from a specific set. This model is responsible for making very strong assumptions and when predicting the future, only the present matters. An HMM [MR15] consists of a doubly stochastic process, and by analyzing a sequence of observed symbols of a stochastic process, we can infer the hidden stochastic process. These hidden states represent an unobservable, or latent, attribute of the process being modeled. Thus, this model is widely used to analyze features or observations in order to predict the most probable sequence of states.

Naive Bayes

Naive Bayes is a probabilistic classifier based on the assumption that every feature is independent. This is one of the most simple and used algorithms due to its high scalability and low computational cost. This algorithm is based on Bayes' theorem 2.1 and assumes a strong independence between features.

$$Bayes' theorem: \quad P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)} \tag{2.1}$$

Decision Trees

Decision trees [SSBD14] are classifiers that predict the labels of the instances by traveling from a root node of a tree to a leaf. The successor child of each node is chosen considering the division of the input space. This division is related to the values of features or based on a set of predefined division rules.

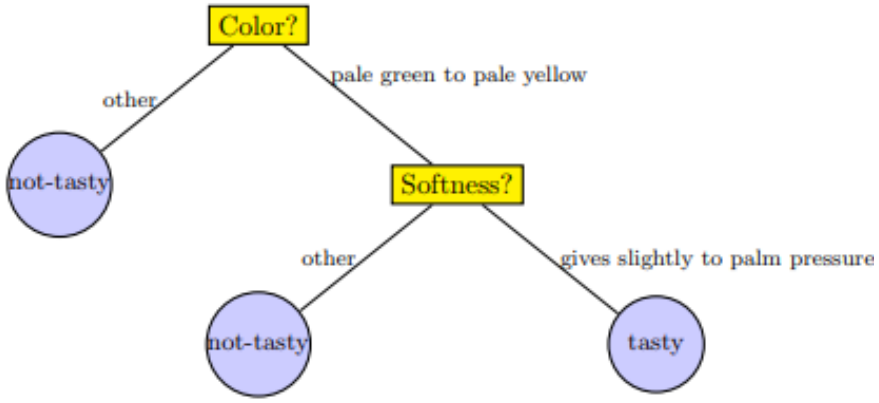

Figure 2.6: Example of Decision tree to check if the taste of a papaya is good.
Source: [SSBD14]

Ensemble Learning

Over the years, ensemble learning has proven to be a very efficient technique in a wide variety of domains and applications. Ensemble-based decision making can be related with our human decision making, for example, when we need to make an important decision, we usually seek opinions of more than one expert. Thus, these kind of technique combine the output of different classifiers and as a consequence it can achieve better accuracies in classifier although it has a higher computational cost compared to a single classifier.[Pol12]

2.5 Summary

This chapter provides an overview of the basic concepts, explaining some of the techniques and algorithms used in this dissertation. Here, are described the typical stages of a HAR system such as data acquisition and the typical sensors used during this step, data preprocessing, including the concepts of noise reduction, feature extraction and selection, data segmentation and data overlapping. Then, it is presented a small explanation on supervised and unsupervised learning, the comparison between online and offline learning, and the most common machine learning algorithms used on HAR systems.

Chapter 3

State of the Art

In this chapter, the related literature is analysed, including some concepts introduced in Chapter 2: data acquisition, data pre-processing and machine learning techniques. In the end, a conclusion is made based on the methodologies used and results achieved in the related literature.

The first article analysed [AMD+15] focus HAR using wearable sensors. It describes sensors placement, data acquisition, data pre-processing and data classification. Then, in [SCF+18], Santos et al. study context-aware mobile applications. The goal is to infer the user context and later publish the results on remote servers. Another paper presented here [HUMA18] proposed a smartphone-based approach for HAR, using DBN [Bon18]. In [SAL+16], it was developed an Android smartphone application in order to test algorithms designed to monitor the physical activity of people in the workspace. The main uses of this application are not only to monitor the physical activity of people in the workplace, but also to collect new data in order to get a complete dataset in addition to the one retrieved from the literature [Lab13]. It is also intended to allow the management of a user profile and then to share this data in an online remote web server. In the last article, analysed [BBS13], the authors studied the problem of recognising hand gestures using wearable sensors, such as accelerometers and gyroscopes. Their focus was to compare different design decisions in activity recognition and to demonstrate their impact on recognition performance. Qin Ni et al. [NPCN16] shows the development of an algorithm to do dynamic detection of window's starting positions. The identification of the starting position of windows helps to determine transitions during the user's activity.

3.1 Sensors placement

According to McAdams et al. [ECN+10], wearable sensors placement affects the accuracy of bodily motions measurement. The sensors need to be correctly attached to the human body, fixed directly to the skin in order to avoid errors during the measurements [BKA+03], [UAM+05].

Waist-placement of the wearable sensors can better represent the human motion since it is close to the centre of mass of the human body. Combining sensors can become weighty for the user, so it is also a challenge to determine the minimum number of sensors as well as their placement in order to reach an accurate activity recognition. Chamroukhi et al. [Por09] obtained the best results for configuration with three accelerometers located at the chest, thigh and ankle. Thus, they concluded that HAR can achieve better results by combining accelerometers located on the upper and lower parts of the body.

3.2 Data acquisition

Ferhat Attal [AMD+15] compare the results obtained on a real dataset using standard supervised and unsupervised machine learning approaches. They used 3 MTx 3-DOF inertial trackers. Each of these sensors is composed by a tri-axial accelerometer that measures the acceleration in the 3-D space. The MTx are connected to a central unit called Xbus Master, that is placed at the belt level of the subject. The inertial trackers transfer the acquired data to the central unit through a wired link, and then this data is transferred to a computer via Bluetooth. It is worth pointing out that the sampling frequency of the system used was 25Hz. The experiments were made in six healthy subjects with different profiles and to whom were given instructions to perform activities in their usual manner. The dataset is composed of 6 sequences (1 per subject) with the same sequential activities order, 12 activities each sequence. There are some activities weakly represented, and the best-represented ones are standing activities (32% of the dataset).

In the project presented in [SCF+18], the main components of the system are a sensor-aggregating node (BlueSentry node), a set of sensors and the mobile device (Sony Ericsson W910i and a Nokia N95). The communication between the sensor node and the smartphone is made via Bluetooth. Data acquisition was made using wearable sensors, such as three-axis accelerometers, temperature sensor, light sensors, sound sensor, humidity, GPS sensors embedded in a backpack or in a vest prototype. They also used two sensors internal to the smartphone, one internal accelerometer and a virtual sensor (to get information related with time) and a third external Bluetooth GPS receiver node. The JSR-256 Mobile Sensor API [Ora19] is used for the internal accelerometer in the Sony Ericsson W910i and JSR-82 Bluetooth API [Mic06] is used for the other data acquisition devices. Data acquisition is made at regular intervals, with a fixed rate. The communication between the devices is made via request, and the smartphone sends a request to the sensors module and the sensors connected to the module retrieve the data. Then, the data is buffered in the smartphone to start the preprocessing phase. As the external sensors may have a different acquisition rate, the acquisition is forced to run at the slowest rate, so all the sensors have the same acquisition rate. On the other hand, the internal sensors acquire data two times faster than the external sensors.

In the approach presented by Mohammed Hassan et al. [HUMA18], two kinds of sensors were used: triaxial accelerometers and gyroscopes, for the data acquisition. Both sensors worked with a sampling rate of raw signals of 50Hz.

The datasets used in [SAL+16] belong to [Lab13] and were recorded with Android smartphones. There are two kinds of datasets used in this study, a bigger dataset recorded by the users in the real world and a smaller one recorded in a controlled laboratory. In this system, two services are running in the background even when the app is closed, one that is responsible for the sensors data collection and another one for the activity tracking. The tracking service is only active when the screen is off and when the smartphone is recognised as in the user's pocket. To ensure real-time performance, these services are split into two different threads.

The lab dataset collection was made in a controlled laboratory environment with the help of 36 volunteers using the smartphones: Nexus One, HTC Hero and Motorola Backflip. The volunteers were asked to do a set of 6 activities, and while performing them, the smartphone had to be placed in their front pants pocket. The total time of the dataset recording was around 15 hours, which corresponds to 25 minutes per user. [SAL+16]

The real-world dataset collection had no constraints and was recorded freely by the users in their daily life. This dataset contains data from 563 users, and 67 have an injury that affects how they walk. Also, the total time of this dataset recording was around 42 hours. [SAL+16]

Andreas Bulling et al. [BBS13] acquired data related to two people performing continuous arm movements. These movements contained eight different gestures as well as typical movements executed during a tennis game. After performing one activity, the users had to make a break before starting the next activity. Thus, the authors included the NULL class, which corresponded to no activity being performed, in a total of 12 activities. The activities were recognized with the help of three custom Inertial Measurement Units located on the outer side of the right lower and upper arm of each participant. The Inertial Measurement Units contained a two-axis gyroscope and a three-axis accelerometer, programmed to record data at a sampling rate of 32Hz.

3.3 Data structures

The application developed in [SAL+16] needed data structures to save the data collected and the data generated by the app. Thus, an internal database was designed to store the tracking and the training data in different tables.

One problem they had to solve was that the database was growing in size every day, so they had to optimise it. One solution was to store only the absolute timestamp of the window, the label and small set information on the table responsible for the tracking data. Another problem was the time needed to write a group of tuples in the database. This operation took around 1.5 seconds when using the classes from Android SDK [Goo], so they decided to use SQLite JDBC [Sai], which was much faster, around 100 milliseconds.

3.4 Energy consumption

D. Chakraborty et al. [YSC+12] study some techniques to improve energy consumption. The authors concluded that by using lower sampling rates, it is possible to minimise the energy con-

sumption, however, they lose accuracy in the activity recognition. Another constraint that affects energy consumption is the number of sensors used, and in [WLA+09], it was developed a framework that has this problem in mind. Thus, they propose a system that can turn on and off the sensors dynamically in real-time. This way, the system recognises a situation when some sensors are not useful and turn them off, saving energy. For example, when the user is performing the activity "sitting" or "standing still" the GPS sensor is not needed and can be turned off.

In [KIR+05] it is studied the relation between the sampling rate and the battery lifetime of a smartwatch. By decreasing the sampling frequency from 20Hz to 6Hz, the battery lifetime increased from 9.2 to 17 hours, corresponding to an increase of 85 per cent.

3.5 Context awareness

Liang Cao [CWZ+17] developed a layer based on context awareness to reduce classification error by "correcting" classifiers opinion. This approach adds constraints to transitions of activities, for example, based on real-life experience if the current activity is lying, walking is impossible to be the next activity since between these two activities standing must happen. Another similar approach is presented in [ZWR+17], where authors define a window of activities with length 2k +1 and change the current activity (in the middle of the window) to the activity with more occurrences in that window. This approach "corrects" the classifier opinion with a delay of k activities in real time classification.

3.6 Machine learning

Ferhat Attal et al. [AMD+15] presented two cases, one using raw data and another one using feature extraction and selection. In the first case, using raw data, they applied supervised and unsupervised machine learning techniques and compared the results.

Taking into account supervised machine learning, to implement an SVM model, the authors used LIBSVM toolbox [CL]. They applied a grid search method in order to estimate the hyper-parameters C and Y and found the optimal values C=2 and Y=5. For the RFs [SSBD14] algorithm, only one parameter was needed, the number of trees. In this case, the best number of trees was 20. In the case of SLGMMs [Bon18], 12 diagonal Gaussians were used. For the kNN method, the best value of k varies from 1 to 20. In this case, the value of k that gives the best accuracy was k = 1. [AMD+15]

Considering unsupervised machine learning, it was used the HMM toolbox [K98] to implement HMM with GMM [Bon18] emission probabilities. In this method, two hyper-parameters were used, the number of states and the number of mixtures. The number of activities to recognise was 12, so the number of states used was 12 too with an ergodic topology. In the other hand, the number of mixtures varies from 1 to 4 and to get the best accuracy rate they used a mixture of 2 diagonal Gaussians. For the K-means algorithm, it is only needed to estimate the number of clusters. In this case, it is the number of the activities (k = 12). In the GMM algorithm, it was

also needed to estimate one parameter, the number of mixture, which corresponds to the number of activities (12 diagonal Gaussians), as in K-means. [AMD+15]

The architecture of the system implemented in [SCF+18] starts with a training period, used to collect the classified examples. These examples are essential to induce a decision tree that is later used to identify the context. When the context is recognised, the results are published to a remote server.

Some sensors used in this system can map directly raw values to a category, but the obtained data by accelerometer data needed more elaborated preprocessing. To get a more elaborated and precise categorization, instead of using instant values, this system used a window of sensor readings and then pre-processed them. For almost all sensors, the system applied thresholds to the mean value using an 8-sample buffer window. However, for the three-axis movement accelerometer, the preprocessing consisted in computing the variance for each axis using a 16-sample buffer window. Then, they compared the three variances obtained (for each axis) with a threshold, in order to differentiate between "moving" and "not moving" activities. If the activity "moving" was recognized, the system would compute a Fast Fourier Transform [Bon18] on the last 32 samples. The last step consisted in verifying if the result was higher than a specific threshold and if so, the category was concluded as "moving fast". On the other hand, if the result was lower than that threshold, the category "moving" was reported. [SCF+18]

The decision tree built during the training period can be updated in real-time when the user trains the system with new contexts. Thus, users are allowed to delete learned contexts and train the system with new ones. This process of rebuilding the tree can be done in real-time, and it is so fast that the user does not notice it. The implementation of the context inference is based on the ID3 algorithm [Qui86] and for comparison, Andre Santos et al. have also implemented the C4.5 algorithm [Qui93]. [SCF+18]

In one experiment [Por09], the system was trained with four human activities, walking, running, idle and resting and they were performed in four different environments. To collect the training data, they used a training period of 5 minutes for each of the four activities in each of the four environments. The training set used included 1200 examples for each activity, and it was used to build the decision tree. On the other hand, the testing set had 800 examples for each activity, and it was used to get the accuracy of the classifiers of the user contexts. To build the decision trees, two algorithms were used: ID3 and C4.5. The trees obtained using ID3 tend to be larger and deeper than the ones obtained using C4.5. In general, the ID3 trees overfit the training data, however the results showed they achieve higher accuracy compared to C4.5. Thus, ID3 had a higher accuracy than C4.5 in all activities except "walking". [SCF+18]

Firstly, during the feature extraction and selection stage in [HUMA18], the noise is removed, using median and low-pass Butter-worth with 20Hz of frequency. Low-pass Butter-worth was used again in the acceleration signals with gravitational and body motion components to get the most relevant information about body acceleration and gravity. It was considered a 0.3Hz as optimal corner frequency to obtain a constant gravity signal. More signals were studied such as body angular speed magnitude, body angular speed, body acceleration jerk and gravity acceleration

magnitude.

The next step was to perform statistical analysis on sliding windows with a fixed size, to get better-improved features. The signals were sampled with sliding windows with a time of 2.56 s and overlapping of 50% between consecutive windows. Feature extraction was based on related work [YC17], and then, the authors applied dimension reduction with the help of kernel PCA [SSBD14]. [HUMA18]

Deep Belief network [Bon18] modelling is divided into two phases: pre-training and fine-tuning. The first is based on a Restricted Boltzmann Machine (RBM) with two hidden layers, and after the pre-training, it is used a fine-tuning algorithm to adjust the weights of the networks. In order to get an updated weight matrix, it is applied a Contrastive Divergence algorithm. After the pre-training phase, it is performed a conventional back propagation algorithm to adjust all parameters (fine-tuning). Using the proposed features, the training error is almost zero for a number of epochs equal to 1000. [HUMA18]

During the recordings in [SAL+16], the sampling frequency used was not regular, so it was a problem for the classifiers to analyse their best performances. This happens since, to compare performances, the features extracted had to be from time windows with the same sampling frequency, and when the sampling frequency is different, the performances can vary a lot. With this problem in mind, they established a range for the sampling frequency, minimum of 18Hz and maximum of 22Hz, and discarded all the values outside of this range. The raw data was written sequentially on a text file and needed to be divided per user and per activity. Thus, the windowing technique used consisted of incrementally fill in the window during the file reading operation and then, when a change of activity or user was detected, the window was truncated. In this study, the classification was made offline with the help of the WEKA toolkit [Lab13]. To train and test all the classifiers, the authors used a 10-fold cross-validation technique to get better results. The classifiers were evaluated based on their performances: precision, recall and F-score, and also by analysing confusion matrices.

The pre-processing phase during [BBS13] started with the segmentation the signals and testing different window lengths. Then, the authors tested a different combination of features and compared their results. The classifiers studied were the SVM, kNN with k=1, HMM, NB [SSBD14] and DA [SSBD14].

3.7 Results

In the article [AMD+15], it was only studied the accuracy of the HAR. To do so, the evaluation criteria used was the average of the accuracy rate, its standard deviation and F-measure. In this study, the variable β is set to 1, in order to give the same importance to precision and recall.

In the first case, using raw data and considering supervised approaches, kNN algorithm gave the best performances, followed by RF [SSBD14], SVM and SLGMM [Bon18]. Taking into account the unsupervised approaches, HMM achieved the best results followed by GMM [Bon18]

and then K-means [SSBD14]. Although supervised approaches outperformed unsupervised approaches, HMM showed very encouraging results (80% accuracy rate) because unsupervised techniques do not use data labeling, which is time-consuming. [AMD+15]

In the second case, using features extraction and selection, they used nine accelerometer signals, acquired by the three MTx, and extracted 11 time-domain features (mean, variance, median, interquartile range, Skewness, kurtosis, root mean square, zero crossing, peak to peak, crest factor and range). They also extracted 6 frequency-domain features (DC component in FFT spectrum, energy spectrum, entropy spectrum, sum of details coefficient of wavelets, sum of squared details coefficients of wavelet, energy of detail wavelets coefficients and energy of approximation wavelets coefficients). Also for each MTx it was calculated a set of 45 correlation coefficients of an axis, 6 mean and the variance of the norm. [AMD+15]

For each sliding window, 213 characteristics were calculated. The sliding window had the size of 25 samples (1 second) with 80% overlapping, and the transition activities took about 2 seconds. After the extraction of the features, they needed to find a minimal subset of features that could characterise and differentiate the set of activities. So, to do this, they used a wrapper approach based on a Random Forest [SSBD14] feature selection algorithm. As a result of this selection, they got a set of 12 features representing more than 80% relevance as an input of classifiers. In this case, using the supervised approaches, the authors demonstrated that kNN algorithm gives the best results followed by RF, SVM and at last SLGMM [Bon18]. Comparing to the results using raw data, it can be noticed an improvement using feature extraction/selection. Taking into account the supervised approaches, the results obtained showed an improvement in the correct rate of 3% in kNN and RF and an improvement of 1% in SVM and SLGMM. Considering the unsupervised approaches, the results obtained on extracted/selected features showed an improvement in the correct rate of 3% in HMM, 4.5% in k-means and 2% in GMM [Bon18]. [AMD+15]

Although it was observed an improvement of performance when using the extraction/selection of features as the input of different algorithms, the feature extraction/selection requires implementing additional models and routines and requires an additional computational cost which can be penalizing in real-time applications. It can also be concluded that supervised approaches are more accurate than unsupervised ones, but the last ones are more computational efficient. [AMD+15]

The performance of the system developed in [SCF+18] was analysed considering scalability and energy consumption. The main system loop (sensing, preprocessing, feature extraction and context inference) took around 0.415 seconds according to the results. Using a Sony Ericsson W910i, the execution times for preprocessing operations over a window of 16 samples showed that non-optimized DFT implementation gets the worst result followed by FFT [Bon18] that is 5 times faster. The mean and variance achieved the best execution times.

The authors also studied the behaviour and execution times needed for sensing and feature extraction with a varying number of additional sensors. Thus, they used two different smartphones (w919i and N95), adding up to three addition GPS sensors connected via Bluetooth. The results showed that by adding new sensors, the execution increases for the sensing and feature extraction steps, but it was not that significant. The battery life was also tested during the execution of the

system main loop with the context updating every second and sensor data acquisition at intervals of 500 milliseconds. The study also showed that the system could operate for at least 15 hours and is limited by the external GPS sensor node battery (the sensor with the shortest lifetime battery). The smartphone operated without any battery recharge for 20 hours and the BlueSentry node, composed by five sensors, operated for 48 hours. [SCF+18]

The experiments performed in [HUMA18] were tested using a public database that includes 12 activities. For the training and testing were used 7762 and 3162 events, respectively, but the number of samples was not equally distributed. The algorithms used in the experiments were ANNs [SSBD14], SVMs and DBNs [Bon18]. In terms of mean recognition rate, ANNs got 65.31%, SVM 82.02% and DBN 95.85%, showing ANN is not good in differentiation some activities. ANN got 0% recognition of one activity, and the other two algorithms achieved more than 50% recognition for every activity, but this is not conclusive since there are a different number of samples per activity. Considering overall accuracy, the accuracy of ANN was 89.06% (2816/3162 samples were rightly recognised), SVM got an accuracy of 94.12% and DBN 95.85%.

The classifiers tested in [SAL+16] were k-NN, Neural Network and Decision Tree, and to train them, the lab dataset, mentioned before, was used. After evaluating the performance of each classifier, they concluded that NN [SSBD14] classifier gives the best results, with an accuracy of 0.996 and the DT the worst results with an accuracy of 0.905. Taking into account the time to train the models, the faster one was k-NN with 0.1 seconds, followed by DT with 2.88 seconds and at last NN with 967 seconds. Finally, considering the model file size, both DT and NN gave a model of around 1MB, and the k-NN produce a model of around 10MB. Although the DT classifier had the worst performance (accuracy of 0.905), it accomplished the best overall results. Thus, the authors chose DT as the base classifier for the physical activity monitoring system.

In order to achieve better performance in using DT, the problem in differentiating the recognition of some activities had to be solved. This problem occurred because the data was recorded in a controlled environment and some activities, like “walking” were performed at a fixed pace. This way, the walking activity executed at a different or irregular pace was misclassified as “upstairs” or “downstairs”. One of the solutions was to generalise the walking data to obtain a larger range of paces. To do this, they used the data from the real-world dataset and reached better performance in F-score in almost every activity, except “Downstairs” and “Upstairs”. This problem of differentiating activities is easily explained as the users in the real world usually place the smartphone with a different orientation compared to the one used in the lab dataset, causing errors in classification. Thus, the classifier needed to be more insensitive, considering the variation of the smartphone orientation, and in all the different ways users perform the same activity. [SAL+16]

A bootstrap aggregating [Bre96] and pairwise classification [PF07] was implemented to improve the classification model. When this model was applied, it was achieved an overall improvement in performance with an average F-score of 0.988. The problems of the smartphone orientation still affected the activities “Upstairs” and “Downstairs” so it was created a reduced model with two activities: “active” and “not active” activities, which is acceptable for the desired system. With this implementation, they achieved an average weighted F-score of 0.988, and the

smartphone orientation did not interfere with the results. [SAL+16]

The results obtained in [BBS13] show a precision of 94.1% using all the sensors and using the kNN classifier. It was, made a comparison between the impact on the precision, using different combinations of sensors, located in different positions. Using two sensors, located on the hand and lower arm or located on the hand and upper arm, precisions of 92.1% and 93.3% were obtained. Using only one sensor, located on lower arm or upper harm achieved lower precision results, 83.3% and 83.8%, respectively. The authors also tested five cases using five different combinations of features. The best performance was achieved by using the features mean and variance. Usually adding more features leads to better precision, however adding low-quality features to the classifier k-NN is not a good practice, since it is susceptible to the quality of features and it can affect the performance of the system negatively. The sliding window was also tested, and the best performance is achieved by using a window length of 1 second. Later it was made a comparison between other classifiers and the best results were achieved by the SVM (96%) and the k-NN(94%). The other classifiers did not get more than 90% of precision, Naive Bayes (NB) [SSBD14] got 78.2%, Join Boosting (JB) [Bon18] 81.3%, HMM 83.6% and DA [SSBD14] 87.3%.

3.8 Overview

This section shows an overview of the most important approaches presented during this chapter in order to achieve the main goal: improve the feasibility of a smartphone-based HAR system.

Ferhat Attal [AMD+15] used feature extraction and selection to achieve better accuracies than using raw data, although the computational cost is higher. Also, the experiments they made show that supervised approaches give better accuracies and require a higher computational cost compared to unsupervised techniques.

André C. Santos et al. [SCF+18] improved their system performance using only mean and variance as features. They also found that the rise of the execution time during the acquisition and preprocessing stages by adding new sensors is not that significant.

Susanna Spinsante et al. [SAL+16] compared the performance of kNN, DT and NN. NN achieved the best accuracy, while DT was the worst. In terms of training time, kNN was the fastest classifier, followed by DT and then NN. Considering the model file size, DT and NN produced models of around 1 MB while kNN produced a model of around 10 MB. In this case, the authors concluded that DT gave the best overall results.

Andreas Bulling et al. [BBS13] obtained the best accuracy for the system using only the mean and variance as features. They also reduced the number o sensors used, however the accuracy decreased significantly. In addition, they compared the classifiers kNN, SVM, NB, JB, HMM and DA, achieving the best results with SVM and kNN.

Liang Cao et al. [CWZ+17] and [ZWR+17] proposed a layer above classification with the aim to improve the opinion of the classifiers. The first approach [CWZ+17] proposed constraints to transitions between groups of activities and studied the probability of these transitions. The second

approach [ZWR+17] proposed a filter of activities based on the occurrences of the activities within a specified window.

3.9 Summary

This chapter provides important information about the typical problems of HAR systems and some solutions on how to improve the performance of these systems both for desktop and mobile implementations. That includes studies about the sensors placements and their impact on the accuracy of the system, about data acquisition, number of sensors used and their sampling rates, and the data structures used to save the data collected. Then, it is given a focus to the techniques used to reduce the energy consumption , such as sampling rates and number of sensors used. The context awareness studies were very important for this dissertation since both filters implemented and explained in chapter 4 were based on these studies and focus only on increasing the accuracy of the system. After that, it is shown a comparison between the methods of machine learning implemented and the results achieved. Finally, it is shown an overview of the literature with the most relevant information for this dissertation.

Chapter 4

Our HAR Approach

In this chapter are present the tools, methods and techniques implemented in order to achieve the goals initially proposed, as well as the architecture of the applications in which the work was developed.

4.1 Technologies

The implementation of the prototype of the HAR system requires the use of the following technologies: Eclipse IDE for Java development [Foua], Android studio for Android development [Goo] and the framework of machine learning MOA [BHKP10].

The base implementation of the desktop was developed using the Eclipse IDE and Java as the programming language. This application was mainly used for offline learning and to test the impact of the experiments made on the accuracy of the system.

For the development of the Android application, it was used the Android Studio IDE. The mobile application was implemented to measure the classification time per activity on the smartphone.

MOA is an open source framework, written in java, for data stream mining that includes a set of machine learning algorithms and the respective evaluation tools. In this dissertation the MOA framework was used in both desktop and mobile applications since it provides the implementations of the base classifiers (kNN, Naive Bayes and Hoeffding tree) and the implementation of the feature selection techniques (Cfs and ReliefF). This framework is also essential for the development of an online and incremental mobile HAR system.

The programming language Python [Foub] was used to prepare the Pamap2 [RS12] data, originally collected at 100Hz, with other sampling rates, 50Hz, 20Hz, 10Hz, 5Hz, 2Hz and 1Hz. In an online context, lowering the sampling rates of the sensors allows to lower the energy consumption, since the data acquisition occurs with a lower frequency.

4.2 Base HAR

The base HAR system used is presented in [Mag19]. Its main architecture, without considering online/incremental learning, is shown in Figure 4.1 and consists of data segmentation, feature extraction, training and classification phases.

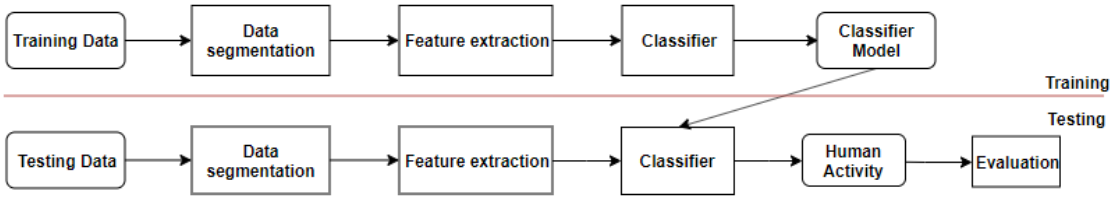

Figure 4.1: Base HAR system architecture.

In this case, the desktop application starts by reading a training file in the Attribute-Relation File Format (ARFF). Then, data segmentation and feature extraction techniques are applied, in order to generate the classification model and then test it by loading the test data, also in the ARFF format. In this HAR system, it is possible to use kNN, Naive Bayes and Hoeffding tree separately or to use a majority voting ensemble with these classifiers.

4.2.1 Preprocessing

In more detail, the data was segmented in sliding windows of fixed size, 9 seconds of data, and with overlapping of 10% between each window [Mag19]. Initially, ten time-domain features were extracted per sensor, mean on the three axes (x, y and z) and the global mean, the standard deviation on the three axes and the correlations between every two axes [FDFC10]. Table 4.1 presents a brief description of each feature. In the case of the Pamap2 dataset, these features imply a total of 90 features per window.

Table 4.1: Description of the features.

Feature	Description
Mean-X, Mean-Y, Mean-Z	Mean of sensor values acquired in each axis separately.
Mean	Mean of Mean-X, Mean-Y and Mean-Z.
Sd-X, Sd-Y, Sd-Z	Standard deviation of sensor values acquired in each axis separately.
Corr-XY, Corr-XZ, Corr-YZ	Set of correlations between the sensor values of two axes.

4.2.2 Classification

During the experiments were used three classifiers and their base implementation present in the MOA framework [BHKP10]: kNN with k = 3, Naive Bayes, Hoeffding tree and the ensemble of them. Table 4.2 shows the complexity of the algorithms used, where n is the number of training windows, f is the number of features of each sample, k is the k value of kNN and d is the depth of the Hoeffding tree. In the training phase, the predicted model is created with the vector of features extracted per window. Then this model is used in the classification stage that is responsible for the final predictions.

Table 4.2: Time complexity of kNN, Hoeffding tree and Naive Bayes in the worst case scenario [KZ09][Elk98][GLG+18].

Algorithm	Time complexity
kNN	*O(nfk)*
Hoeffding tree	*O(nfd)*
Naive Bayes	*O(nf)*

It is possible to conclude that kNN is the most computationally heavy algorithm. Hence, higher classification times and higher energy consumption are expented, when using this algorithm.

4.2.3 Validation

Validation is the process of using a test dataset to evaluate the trained model. In this project, the Leave-one-out cross-validation was used. In this context, it consists of using all data for training except the data from a chosen user. The data used for testing belongs to the chosen user.

4.3 Improved HAR

In this dissertation, the base desktop HAR system was extended, by a feature selection step, using two different algorithms (Cfs [AH00] and ReliefF [RSK03]) and then, by introducing a rule-based classifier. With the proposed techniques, it was expected to reduce the classification times and energy consumptions, at the cost of lower accuracy. Finally, it was implemented a filter post classification that focuses only on improving the accuracy lowered by the methods proposed (see Figure 4.2).

The mobile HAR prototype was also improved. Figure 4.3 shows the layers of the mobile application. Our contributions were layer 5 and 6 and also the vector of features extracted that was obtained via feature selection using ReliefF or Cfs. The rule-based classifier (layer 5) skips the use of the base classifiers in some samples of data and the filter post-classification (layer 6) corrects the predictions of the classifiers. The other layers, 1, 2 and 4 were already implemented. Feature selection is done in the desktop application, first the initial features are extracted and then feature selection algorithms are applied. The resultant vector of features is then loaded to the mobile application. The same happens with the classification layer, the training phase is done in

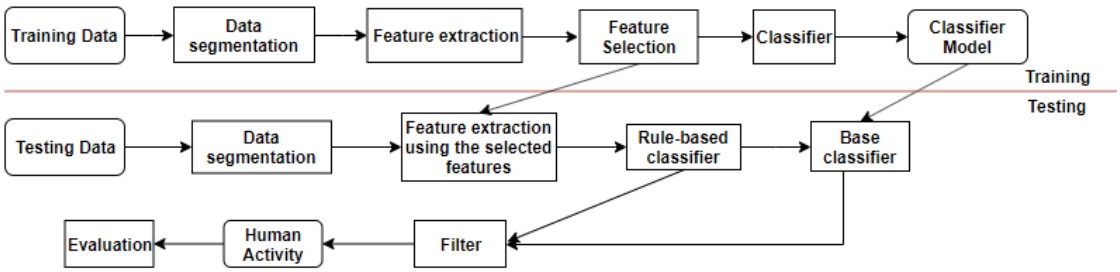

Figure 4.2: The architecture of the improved desktop HAR system.

the desktop application and the training model file is generated (.moa file). Next, this model is loaded to the mobile application. The rule-based skipping and the filter layers work similarly as in the desktop application.

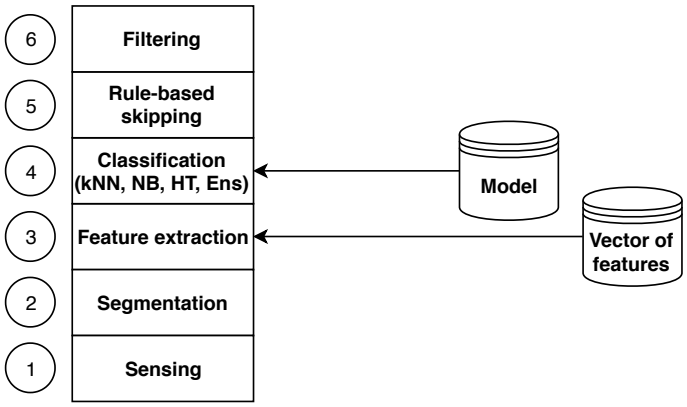

Figure 4.3: Layers of the mobile application.

4.3.1 Feature selection: Cfs and ReliefF

In order to obtain the best set of features, it was studied two different algorithms of feature selection: ReliefF [RSK03] and Cfs [AH00].

ReliefF algorithm is an extension of Relief for multiclass problems, and it is based on estimating the quality of the features considering the differences and similarities between features of instances that are near to each other. Thus, given an instance *Ri*, the algorithm searches for the *k* nearest instance with the same class of *Ri*, called nearest hit *H*, and then searches for the *k* nearest instances from each of the different classes, called nearest misses *Mj*. Then, the quality of all attributes *W[A]* is calculated. It is also important to emphasise that this algorithm has some problems in handling redundant features and noisy data. The pseudo code of ReliefF is presented in Algorithm 1.

Cfs algorithm is based on calculating the correlation between feature-feature and feature-class. Thus, a useful feature is correlated with the class and uncorrelated with other features, otherwise it

Algorithm 1 Pseudo code of ReliefF. Source: [RSK03]

Require: Vector of feature values and class value for each training instance
  $W[A] \leftarrow 0$
  while $i < m$ **do**
    Randomly select an instance Ri
    Find nearest hit H and nearest miss M
    while $A < a$ **do**
      $W[A] \leftarrow W[A] - diff(A,Ri,H)/m + diff(A,Ri,M)/m$
    end while
  end while

is irrelevant. The merit of a subset of features S is defined in equation 4.1, where k is the number of features, $\overline{r_{cf}}$ is the average value of all correlations between features-class and $\overline{r_{ff}}$ is the average value of all correlations between feature-feature. [AH00]

$$Merit_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{4.1}$$

Using these two techniques of feature selection, it is possible to rank the features by importance and select the best features for a specific number.

4.3.2 Rule-based classifier

It was, also, developed a very simple classifier based on a rule. This classifier emerged with the idea of reducing the use of the base classifiers, for example, some activities are performed during long periods of time and there is no need to call a complex classifier everytime. Thus, if a user is running during 1 hour, calling a classifier in every sample of data is a waste of energy. With this approach it is intended to decrease the complexity of the system by reducing the number of times a complex classifier is called.

Firstly, the activities are grouped into two sets, the dynamic activities and the static activities. For example, considering the activities of the PAMAP2 dataset [RS12], the two groups consist of the following:

- **Dynamic activities** — cycling, running, Nordic walking, walking, vacuum cleaning, ironing, rope jumping, ascending and descending stairs;
- **Static activities** — standing, lying and sitting;

Then it is compared a chosen value obtained from the features with a chosen threshold for each sample of data in order to classify that sample as a dynamic activity or a static activity.

This value is obtained by finding out the sensor with the most important features and then compare the mean of the variance of the values from this sensor with the threshold. According to the Table 4.3, the value used is given as $\overline{Var_i}$.

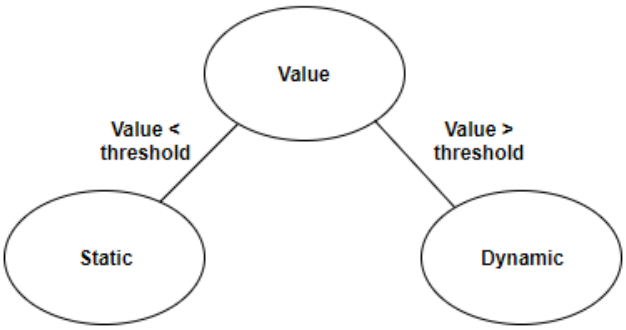

Figure 4.4: Rule classifier.

Later, instead of using this rule classifier it is trained a binary Hoeffding tree with only one feature (the value used in the rule classifier) and with two labels, static and dynamic. The purpose of this tree is the same as the rule classifier, although it is not needed to find an optimal threshold value.

Table 4.3: Variances of the gyroscope values.

Sensor	Variance
Hand gyroscope	Var_1 = Sd-X * Sd-X
	Var_2 = Sd-Y * Sd-Y
	Var_3 = Sd-Z * Sd-Z
Ankle gyroscope	Var_4 = Sd-X * Sd-X
	Var_5 = Sd-Y * Sd-Y
	Var_6 = Sd-Z * Sd-Z
Chest gyroscope	Var_7 = Sd-X * Sd-X
	Var_8 = Sd-Y * Sd-Y
	Var_9 = Sd-Z * Sd-Z

This rule classifier or the binary Hoeffding tree is used along with the base classifiers (kNN, Naive Bayes, Hoeffding tree or ensemble). Firstly, the base classifier generates 2 models, one that trained only dynamic activities and the other that trained only static activities. Later in chapter 5 it is shown the impact of using this 2 models, comparing to using only 1 model. In a first phase, the rule classifier decides which model should be used on the base classifier. Then, it gives the prediction of the first sample and the rule classifier is called in the next sample. If the activity predicted by the base classifier belongs to the classification of the rule classifier on the next sample, the final prediction is the same in every sample until the rule classifier prediction group does not include the base classifier activity. In this case, the base classifier must be called. Fig 4.5 shows an example of this technique. The first prediction, given by a base classifier is *walking*, a dynamic activity. The next sample calls the rule classifier and is classified as a dynamic activity. As the first activity, *walking*, is a dynamic activity, the final prediction of the second sample remains the same as the previous one. When the fourth sample calls the rule classifier, it classifies as a static

activity. As the previous activity, *walking* is not a static activity, the base classifier is called and gives the correct prediction, *standing*.

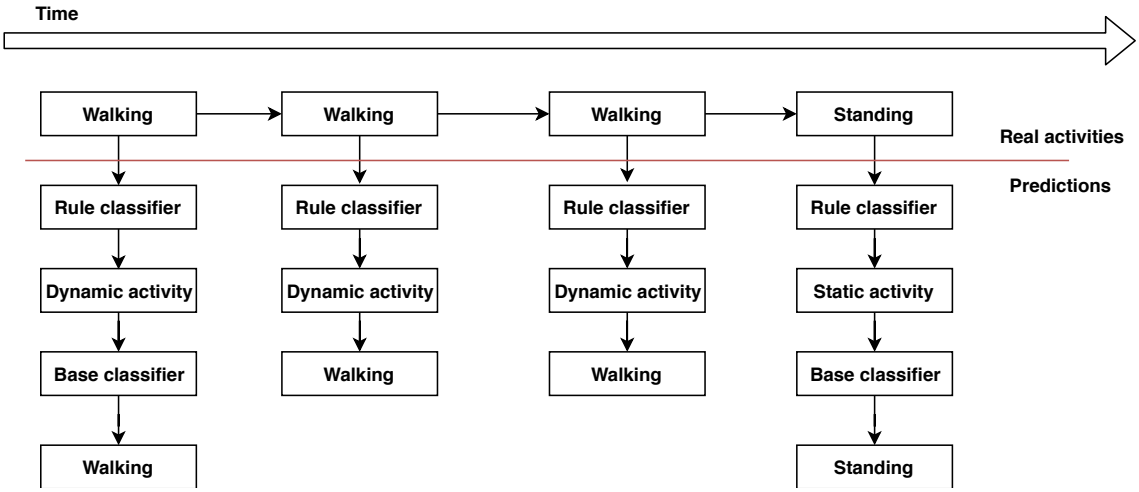

Figure 4.5: Rule-based classifier.

This strategy has a problem when it happens a long sequence of different dynamic or static activities, the rule-based classifier will probably classify it as the same group, and the activity prediction will be the same. For example, if the real sequence is *running* → *walking* → *cycling* → *cycling* the base classifier predicts the first sample as running and then the rule-based classifier predicts the next samples as dynamic, and the final sequence of predictions is *running* → *running* → *running* → *running*. Fig 4.6 shows this problem that leads to wrong predictions and consequently, to a high error in accuracy.

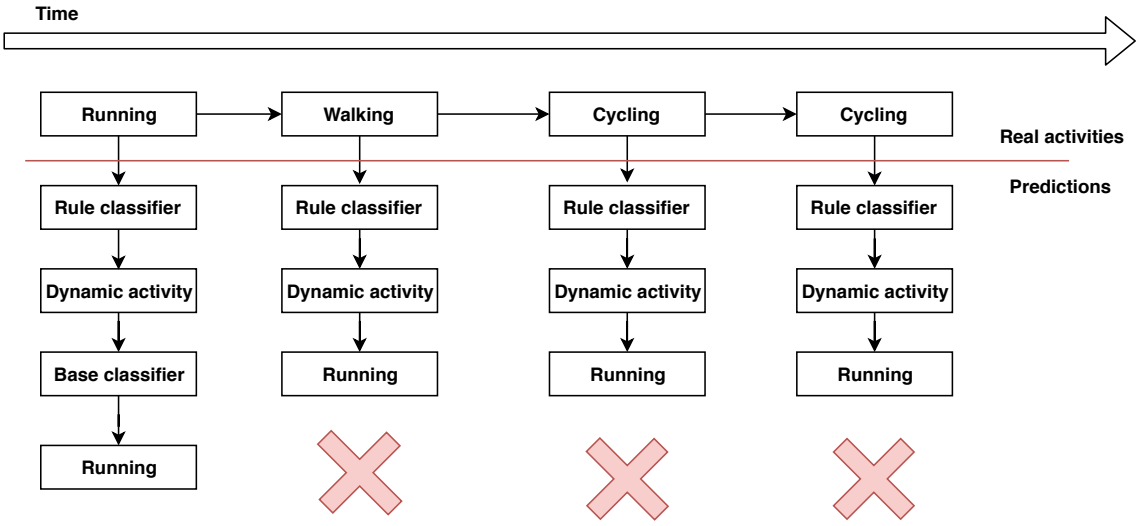

Figure 4.6: Rule-based classifier problem.

A possible solution to the problem mentioned is to force the classification using the base classifier at least every N samples.

4.3.3 Filter

After the classification itself, our HAR system uses a filter to improve the accuracy achieved by the chosen classifier. Thus, the goal of the filter is to correct the prediction of the classifier evaluating the adjacent activities. Three types of filters are proposed:

- **Transitions-1** — this filter is an adaptation of the approach presented in [CWZ+17], where the authors evaluate the probability of the transitions between groups of activities. Our approach is simpler since it only evaluates the possibility of transitions between two adjacent activities and if the transition is impossible it keeps the previous activity predicted. The impossible transitions between activities are presented in Table 4.4.

Table 4.4: Impossible transitions between activities.

	Lying	Sitting	Standing	Ironing	V. cleaning	A. stairs	D. stairs	Walking	N. walking	Cycling	Running	R. jumping
Lying				X	X	X	X	X	X	X	X	X
Sitting					X	X	X		X	X	X	X
Standing												
Ironing	X				X	X	X		X		X	X
Vaccum cleaning	X	X		X		X	X		X	X	X	X
Ascending stairs	X	X		X	X					X		
Descending stairs	X	X		X	X					X		
Walking	X											
Nordic walking	X	X		X	X					X		X
Cycling	X	X			X	X	X		X			X
Running	X	X		X	X							X
Rope jumping	X	X		X	X				X	X	X	

- **Transitions-2** — this filter is also based on [CWZ+17]. The difference is our approach only evaluates the possibility of transitions between activities. Here, the possibility of the transitions between activities is evaluated using a window of *2*n+1* activities. If there is an impossible transition between the current activity and the previous or the next activity, one of these activities is changed to the one with most occurrences in the window. For example, in Figure 4.7 the transition between *Running* and *Lying* is impossible and the activity *Lying* has more occurrences (3) than the activity *Running* (1). Thus, the activity *Running* is changed to *Lying*.

- **Occurences filter** — this filter was used in [ZWR+17] and it evaluates the number of occurrences of an activity within a window of *2*n+1* activities. In Figure 4.8, the current activity changes from *Sitting* to *Lying*, since the majority of activities classified within this window is *Lying*. If the current activity is already equal to the majority of that window, nothing changes.

Both transitions-2 and occurrences filter have a problem in common considering an online implementation since it evaluates the transitions with a delay of *n* activities (considering the size of window *2*n +1*).

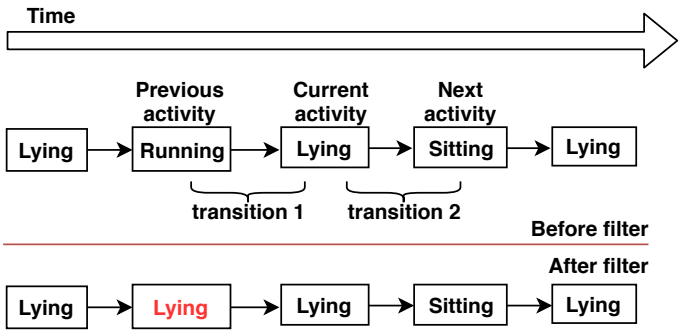

Figure 4.7: Example of transitions filter.

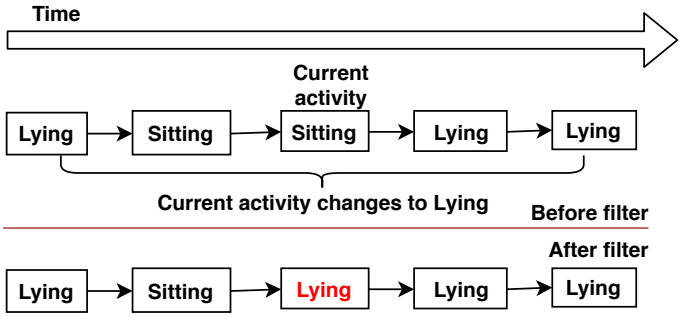

Figure 4.8: Example of Occurences filter.

4.4 Summary

This chapter starts by explaining the technologies used for the implementation and test of the desktop and mobile HAR system. Afterwards, it described the architecture and the behavior of the base HAR system, namely, the process of data segmentation, feature extraction, classification and validation. Next, the contributions of this dissertation were presented, such as, two feature selection techniques, Cfs and ReliefF, a rule-based classifier to reduce the energy consumption and the classification time by reducing the frequency that a base classifier is used. Finally, three filters based on occurrences or transitions between activities were explained, and its goal is to increase the accuracy of the system.

Chapter 5

Results

This chapter presents the results of all experiments performed, evaluating the impact of the sampling rates, filters, the number of features, base classifiers and rule-based classifier on the performance of the system. Firstly, the experiments were done in a desktop application previously developed in DEI. Then, the successful experiments made in the desktop application were tested in a mobile prototype also previously developed in DEI.

5.1 Experimental setup

All the experiments use a window length of 9 seconds, 10% of overlapping and the dataset Pamap2 [RS12].

The desktop version was executed on a computer with an Intel Core i7-5930k (3.50GHz) and 32GB of memory RAM. The mobile experiments were performed on a Samsung Galaxy J4+. It was used an ODROID-XU [Har] board to measure the energy consumption of the system. This device consists of four big Cortex-A15 cores and four small Cortex-A7 cores. The experiments run on performance mode and it was only used one core of the Cortex-A15, with a clock frequency of 1.6 Ghz. Some of the metrics used during the experiments are detailed in Table 5.1.

5.1.1 Dataset

Pamap2 dataset [RS12] contains information about 18 physical activities, divided in different groups [GCM+18]:

- **Basic activities** — cycling, running, Nordic walking and walking;
- **Posture activities** — standing, lying and sitting;
- **Everyday activities** — ascending and descending stairs;
- **Household activities** — vacuum cleaning and ironing;

Table 5.1: Description of the metrix used during the experiments.

Metric	Description
Accuracy	The accuracy of the system corresponds to the number of activities correctly predicted dividing by the total number of activities. When using more than 1 user, the accuracy is the mean of all users' accuracy.
Mobile classification time	The mobile classification time is the average time it takes to classify one activity on the smartphone.
Desktop classification time	The desktop classification time is the average time it takes to classify one activity on the desktop.
Energy	The energy measured on the Odroid-XU corresponds to the energy consumed by one core of the Cortex-A15 during the classification phase (classification of all the testing data).

- **Fitness activities** — rope jumping;
- **Optional activities** — watching TV, car driving, watching TV, playing soccer, house cleaning and folding laundry.

Although the dataset contains 18 physical activities, this study only uses 12 activities (the optional activities are not taking into account).

The data was collected at a sampling rate of 100Hz with the help of 9 different users. During the data acquisition step, each user was wearing three 3-axis accelerometers, three 3-axis gyroscopes and three 3-axis magnetometers, one of each kind per location, as shown in 5.1, resulting in a total of 9 sensors. It is also worth pointing out that this dataset has a total of 1,926,896 samples of raw data from all the nine users.

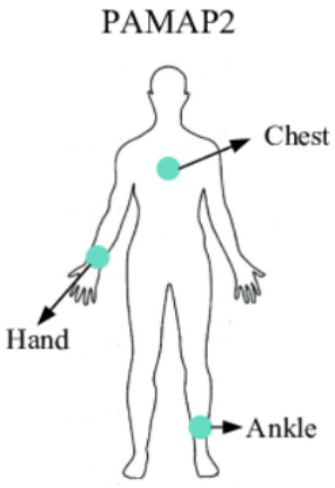

Figure 5.1: Pamap2 sensors location.
Source: [WCH+18]

Table 5.2 shows the distribution of Pamap2 data per user, namely, the number of activities each user performed and the corresponding accuracy. User 9 data achieved an accuracy of 100%, using the ensemble of kNN, Naive Bayes and Hoeffding Tree, although it only has data from one activity (Running). The data with the best ratio between number of activities and accuracy belong to user 7 and 4.

Table 5.2: Number of activities performed and accuracy per user, using Ensemble of kNN, Naive Bayes and Hoeffding tree.

User	Acc. (%)	No. activities	Num samples
1	67.95	12	312
2	79.33	12	329
3	74.65	8	217
4	91.35	10	289
5	88.24	12	340
6	82.05	11	312
7	91.03	11	290
8	74.92	12	327
9	100	1	7

Firstly, the procedures were tested using Leave-one-out cross-validation considering all nine users in Pamap2, and in the end, it is calculated the average of the values measured for all users. Afterwards, some experiments were replicated but using a specific user (user 7).

5.2 Sampling Rates and Filter

The first experiments study the impact of the sampling rates and the filter, explained in Chapter 4, on the accuracy of the system using kNN, Naive Bayes, Hoeffding tree and the ensemble of them (see Figures 5.2, 5.3, 5.4, 5.5). These results corroborate state of the art, namely, by decreasing the sampling rates, the accuracy also decreases, with some exceptions. It is still worth pointing out that the transitions-2 filter has a significant impact on the accuracy, improving it by at least 2% in every case and the occurrences filter is almost always better than the transitions-2 filter. There is one case, using kNN and 1Hz, where the occurrences filter improves the results for approximately 10%, from 70.86% to 80.56%. The transitions-1 filter does not seem to have a significant impact on the accuracy of the system and in some cases the accuracy gets even worse. Thus, the following experiments do not consider transitions-1 filter.

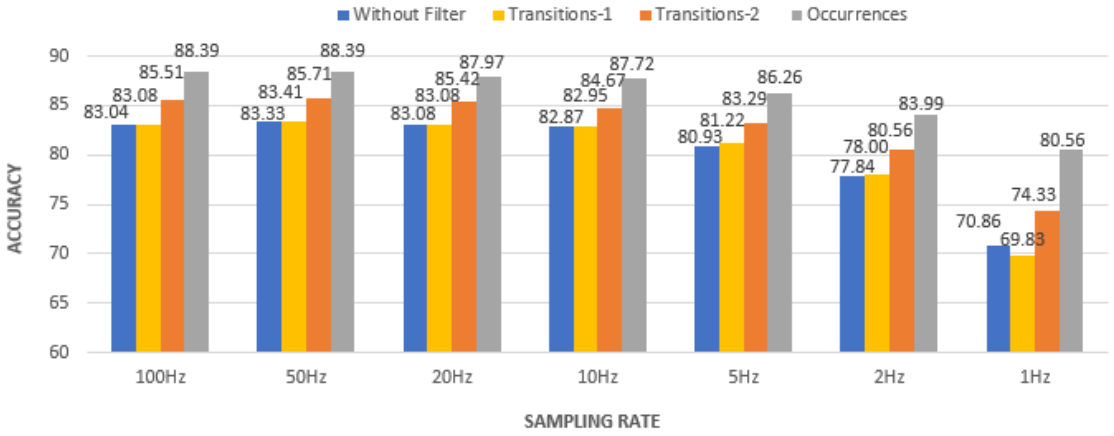

Figure 5.2: Impact of sampling rate and filter on accuracy using kNN.

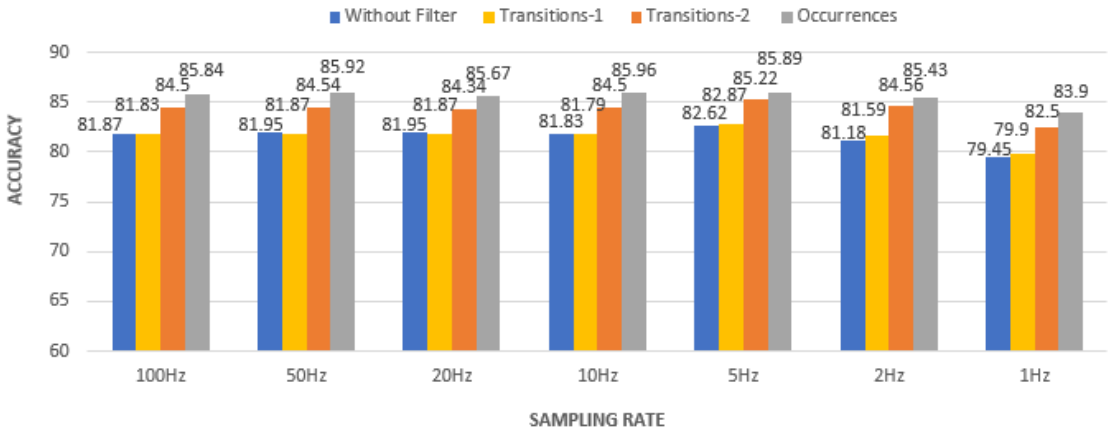

Figure 5.3: Impact of sampling rate and filter on accuracy using naive bayes.

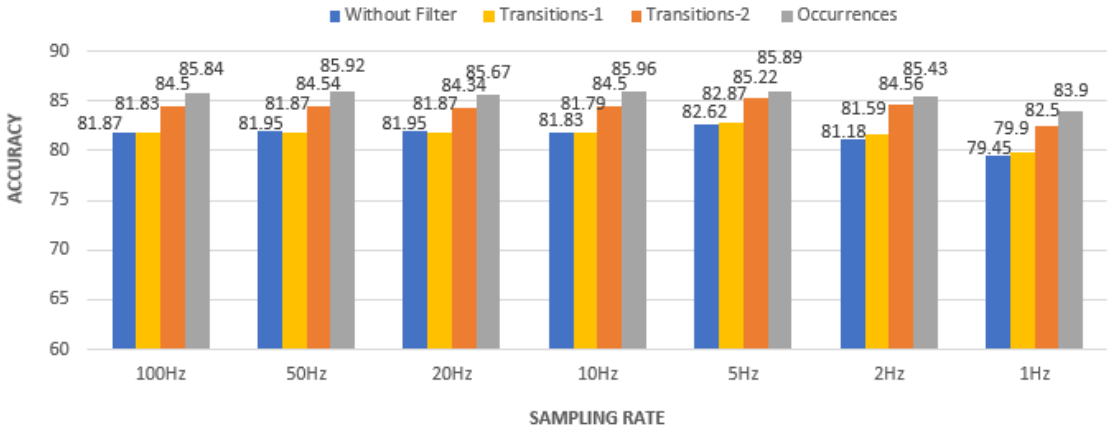

Figure 5.4: Impact of sampling rate and filter on accuracy using hoeffding tree.

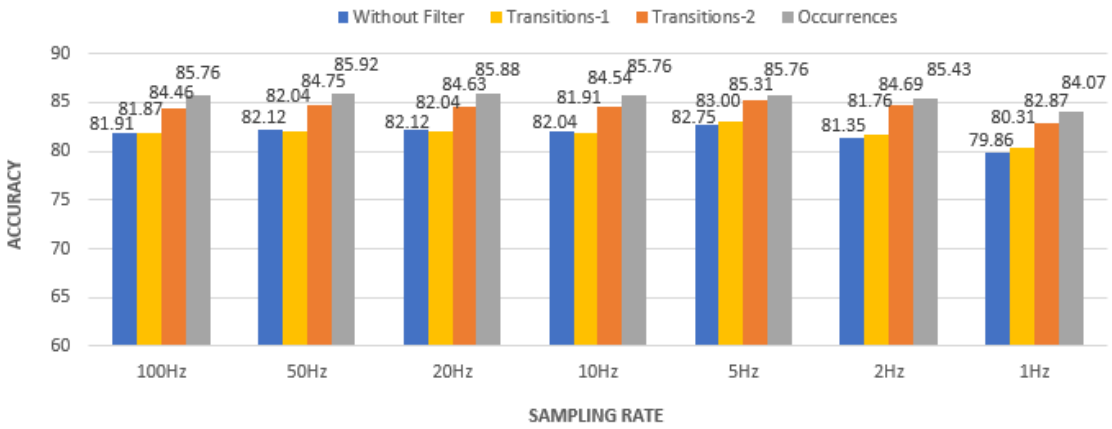

Figure 5.5: Impact of sampling rate and filter on accuracy using ensemble of kNN, naive bayes and hoeffding tree.

Afterwards, it was studied the impact of the sampling rate on the feature extraction time. Taking into account Table 5.3, it is possible to understand that by decreasing the sampling rate, the number of samples per window also decreases since the sliding window used has a fixed size of 9 seconds. This reduction of the number of samples per window leads to a consequent decrease on the feature extraction time. Note that using 2Hz the feature extraction time decreased from 20 ms to 0.85 milliseconds. Thus, it was decided to use the 2Hz as a sampling rate since it has a good trade-off between accuracy and feature extraction time.

Table 5.3: Impact of sampling rate on feature extraction time using windows with length of 9 seconds.

Sampling rate (Hz)	**Samples per window**	**Feature extraction time (ms)**
100	900	20
50	450	10
20	180	4
10	90	2.25
5	45	1.35
2	18	0.85
1	9	0.6

The classification time is not related to the sampling rate, therefore the variation of the sampling rate does not have any effect on the classification time. It is only affected by the number of features and by the classifier used (see Table 5.4). As expected, kNN has a much higher classification time, when compared to Naive Bayes and Hoeffding tree.

Table 5.4: Impact of each classifier (kNN with $k = 3$) on mobile classification time using windows with length of 9 seconds.

Classifier	M. classification time (ms)
kNN	133
Naive Bayes	1.4
Hoeffding tree	1.6
Ensemble	142

5.3 Number of Features

Here, it is presented the impact of the number of features, used during the classification, on accuracy and on the classification time. Figure 5.6 shows the variation of accuracy with a different number of features selected with Cfs [AH00] and ReliefF [RSK03], and Figure 5.7 allows to understand the behavior of the classification time when the number of features decrease.

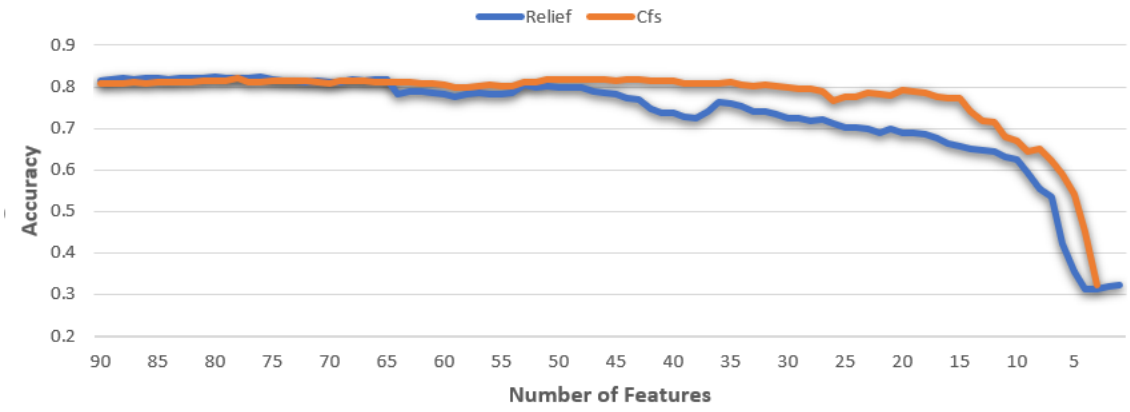

Figure 5.6: Impact of the number of features on accuracy using the ensemble of naive bayes, kNN and hoeffding tree using a sampling rate of 2Hz.

The results show that the decrease of the number of features leads to a linear decrease in the classification time. This reduction occurs since the complexity of the algorithms is linear, and all the three classifiers depend on the number of features. Notice that when using 30 features selected by Cfs, the accuracy only decreases 1% and the classification time reduces from approximately $7ms$ to approximately $2ms$. Also, feature selection using Cfs proved to be more efficient compared to ReliefF, particularly when the number of features selected is lower than 50.

In the following tables (5.5 and 5.6) are visible some scenarios. For example, for a system that requires a specific minimum accuracy it is shown the most efficient configuration.

These studies lead to the conclusion that the system uses a lot of redundant and useless features since by reducing from 90 to 29 features the accuracy only decreases approximately 1%, and the maximum accuracy is obtained using 76 features. Considering the use of the occurrences filter, the maximum accuracy achieved is 88.32% with 18 features. In this case, the increase of performance

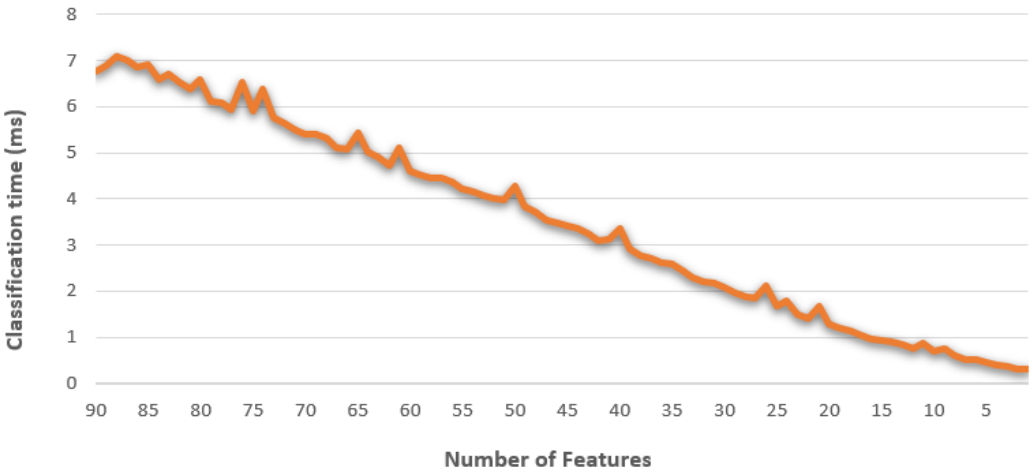

Figure 5.7: Impact of the number of features on desktop classification time using ensemble of naive bayes, kNN and hoeffding tree using a sampling rate of 2Hz.

Table 5.5: Best system configuration for a specific scenario using ensemble classifier and without using filter.

Scenario	Acc. (%)	No. features	F. Selection	M. classification time (ms)
Ensemble	81.35	90	-	152
Max. accuracy	82.41	76	Relief	124
80 % accuracy	80.23	29	Cfs	42
75 % accuracy	77.14	13	Cfs	18
70 % accuracy	71.44	10	Cfs	14

Table 5.6: Best system configuration for a specific scenario using ensemble classifier and using occurrences filter.

Scenario	Acc. (%)	No. features	F. Selection	M. classification time (ms)
Ensemble	85.43	90	-	152
Max. accuracy	88.32	18	Cfs	25
80 % accuracy	80.31	10	Cfs	14
75 % accuracy	76.64	8	Cfs	12
70 % accuracy	71.85	5	Cfs	9

was huge since comparing to the base scenario, Ensemble without filter, the accuracy increased 7% and the classification time per sample decreased from 152 to 25 milliseconds.

5.4 Rule Classifier

The studies present in this section are essential to obtain the optimal values for the rule classifier. Firstly, it is listed to which sensor belong the most important features, selected using Cfs, in order to discover the sensor with the most important features. Observing Table 5.7, it is concluded that

the values acquired by gyroscope are, by far, the most impactful. Then, as stated in Subsection 4.3.2, it is compared the mean of the variance of the values from this sensor, in this case gyroscope, with the threshold defined.

Table 5.7: Number of features selected per sensor using Cfs for the 10 most important features.

User	Accelerometer	Gyroscope	Magnetometer
1	1	6	3
2	2	7	1
3	1	8	1
4	1	7	2
5	0	9	1
6	2	6	2
7	1	6	3
8	2	7	1
9	1	8	1
Total	11	64	15

Now the optimal value was obtained, it must be compared with a threshold in order to differentiate static and dynamic activities (see Figure 5.8).

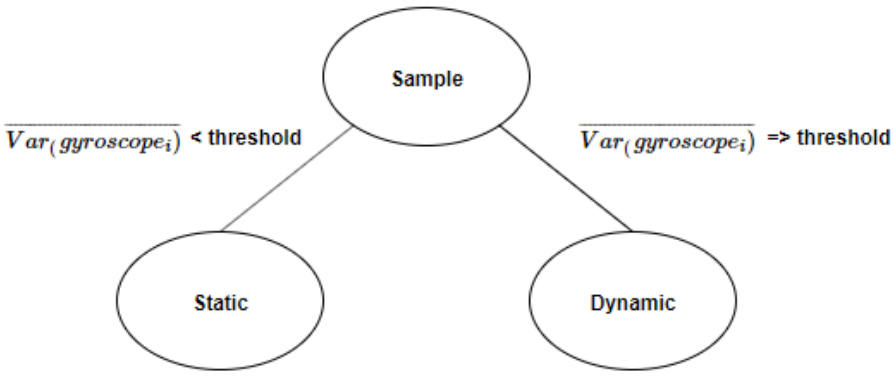

Figure 5.8: Rule classifier.

The threshold value was calculated empirically by exploring different values and verifying the accuracy of the application. Considering Table 5.8 it is possible to conclude that the optimal value for the threshold is 0.06, and it gives an accuracy of 92.45% when differentiating static and dynamic activities.

Table 5.9 shows the accuracy of the base classifiers kNN, Naive Bayes, Hoeffding tree and Ensemble for this binary problem. It is possible to understand that the accuracies achieved by using simple rules, 91.54 % using a threshold value of 0.06, are better than using Naive Bayes and Hoeffding tree.

Table 5.8: Comparison between threshold values for binary rule classifier.

Threshold	Acc. (%)
0.01	88.61
0.02	90.47
0.03	91.00
0.04	91.09
0.05	91.50
0.06	91.54
0.07	91.33
0.08	90.88
0.09	90.63
0.10	90.01

Table 5.9: Comparison between classifiers for binary classification.

Classifier	Acc. (%)
kNN	94.55
Naive Bayes	90.38
Hoeffding tree	90.88
Ensemble	94.76

5.5 Rule-based Classifier

The accuracy of the rule-based classifier without defining a maximum number of skips is very low, 38.57%, according to the explanation given in 4.3.2. In such a way, it was studied the behavior of the system by defining the maximum number of windows skipped by the complex classifier using 2 models (1 static and 1 dynamic). Table 5.10 shows the percentage of ensemble calls and the accuracy of the system using a specific maximum number of windows skipped. By skipping a maximum of 2 windows it is possible to achieve 76.51% of accuracy. Comparing to the base classifier, the accuracy decreases from 81.35% to 76.51%, but the number of calls of the complex classifier also decreases from 100% to 37.74%.

Considering Figure 5.9, the impact of the number of features on this classifier is slightly lower, compared to when using a base classifier. In this case, the accuracy does not vary much when decreasing from 90 to 20 features and the features selected by Cfs achieve a better performance compared to the ones selected with ReliefF.

Tables 5.11 and 5.12 present the most efficient system configuration for the specified scenarios, using rule-based classifier with 2 models and with or without filter. Without using the filter it was not possible to achieve an accuracy higher than 80%. Note that with 44 features and using occurrences filter, it is achieved an accuracy of 80.99% and the classification time decreases from 73 milliseconds to 35 milliseconds. Comparing to the baseline classifier (Ensemble with 90 features and without filter), the accuracy achieved with 44 features and with filter is lower only 0.35% and the classification time decreased from 152 to 35 milliseconds.

Table 5.10: Comparison of performance by skipping a maximum number of windows. The total number of windows tested is 2423.

No. windows skipped	Ensemble calls (%)	Acc. (%)
0	100	81.35
1	53.31	77.71
2	37.74	76.51
3	30.12	74.86
4	25.35	73.32
5	22.49	72.95
6	20.71	71.29
7	18.97	69.42
8	17.52	68.31
9	16.57	69.68
10	15.87	67.44
Without max. no. skips	8.5	38.57

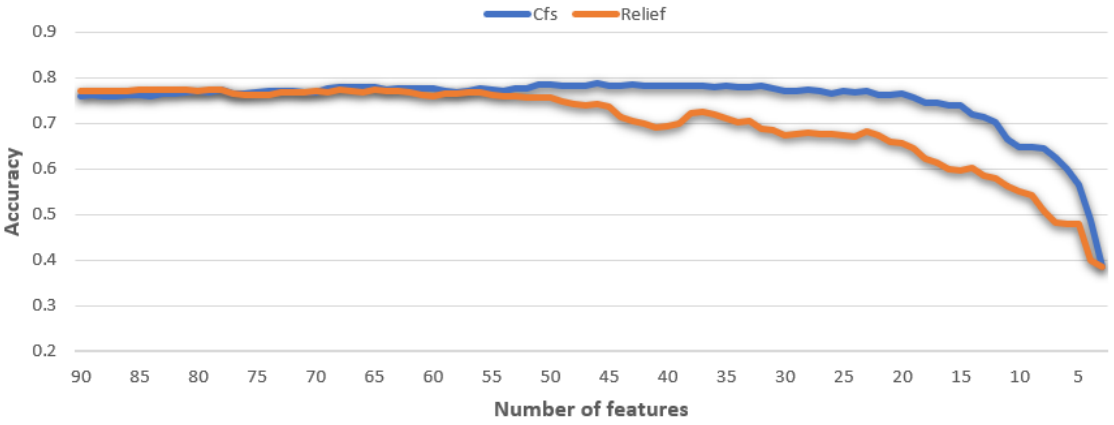

Figure 5.9: Impact of the number of features on accuracy using rule-based + Ensemble classifier.

Table 5.11: Best system configuration for a specific scenario using rule based ensemble classifier with a maximum number of windows skipped = 2 and without using filter.

Scenario	Acc. (%)	No. features	F. Selection	M. classification time (ms)
Rule-based + Ensemble	76.51	90	-	73
Max. accuracy	78.62	44	Cfs	35
80 % accuracy	-	-	-	-
75 % accuracy	75.6	17	Cfs	13
70 % accuracy	70.17	10	Cfs	9

5.6 Results using a specific user

In this Section, are shown the results obtained when testing the improved HAR system with user 7 for testing, and training it with the data of the other N users.

Table 5.12: Best system configuration for a specific scenario using rule-based + Ensemble classifier with a maximum number of windows skipped = 2 and using occurrences filter.

Scenario	Acc. (%)	No. features	F. Selection	M. classification time (ms)
Rule-based + Ensemble	78.79	90	-	73
Max. accuracy	80.99	44	Cfs	35
80 % accuracy	80.32	30	Cfs	23
75 % accuracy	76.22	13	Cfs	11
70 % accuracy	72.2	10	Cfs	9

5.6.1 Impact of the filter

The results in Table 5.13 show that the two filters evaluated (i.e., Transitions-2 and Occurrences filters) are efficient, improving the accuracy of almost all activities. The occurrences filter achieved a better global accuracy when compared to the transitions-2 filter, improving the global accuracy by more than 4%. The occurrences filter has more impact on *Ironing*, improving 17% of the predictions. This activity is often wrongly classified as *Standing* and the transitions-2 filter can not correct it since the transition between *Ironing* and *Standing* is possible. *Ironing* is classified once as *Vacuum cleaning* and the transitions-2 filter can correct this one (corresponding to the 3% of improvement) since the transitions between *Ironing* and *vacuum cleaning* is impossible. The transitions-2 filter has more impact on *Vacuum cleaning* and on *Cycling*. *Cycling* is wrongly classified one time as *Sitting* and another time as *Vacuum cleaning* (both impossible transitions). *Vacuum cleaning* is wrongly classified 3 times as *Ironing* (impossible transition) and 1 time as *Standing* (possible transition). The activity with lower accuracy if *Running* with 75% and none of the filters can improve it. This activity only has 4 samples and it is wrongly classified one time as *Standing* (possible transition). Also, the occurrences filter can not correct this activity since the sequence of activities is given as *Cycling* $\rightarrow$ *Cycling* $\rightarrow$ *Standing* $\rightarrow$ *Running* $\rightarrow$ *Running* and the number of occurrences of *Running* is the same as *Cycling* (2 times each), consequently *Standing* is not corrected.

5.6.2 Rule-based classifier

In this case, Table 5.14 shows that the best threshold value for the rule classifier is 0.02, so the next experiments use a threshold equal to 0.02.

Table 5.15 presents the performance of the system with a different number of maximum windows skipped by the complex classifier using 2 models (1 static and 1 dynamic). It is possible to conclude that by increasing the number of windows that skip the complex classifier, the ensemble calls decrease as well as the energy consumption. By skipping a maximum of 5 windows, the energy consumption decreases from 36.1 J to 8 J at the cost of approximately 4% lower accuracy. Therefore, the next experiments with the rule-based classifier use a maximum of 5 windows skipped by the complex classifier.

Then, it is verified if there is any activity that skips more or less windows, using a maximum of 5 windows of complex classification skipped. As it is shown in Table 5.16, almost every activity

Table 5.13: Impact of filter per activity, using an Ensemble consisting of kNN, Naive bayes, Hoeffding tree and using 90 features.

Activity	Samples (%)	Acc. (%)		
		Without filter	Transitions-2 filter	Occurrences filter
Lying	11.03	93.75	96.88	93.75
Sitting	5.2	86.67	86.67	86.67
Standing	11.38	100	100	100
Ironing	12.41	80.56	83.33	97.22
Vacuum cleaning	9.31	85.19	92.59	88.89
Ascending stairs	7.59	86.36	90.91	90.91
Descending stairs	5.17	80	80	86.67
Walking	14.48	100	100	100
Nordic walking	12.41	94.44	94.44	100
Cycling	9.66	92.86	100	100
Running	1.38	75	75	75
Global	100	91.03	93.45	95.52

Table 5.14: Comparison between threshold values for binary rule classifier

Threshold	Acc. (%)
0.01	92.76
0.02	93.45
0.03	93.10
0.04	92.41
0.05	92.76
0.06	92.76
0.07	92.41
0.08	90.34
0.09	89.31
0.10	87.24

Table 5.15: Comparison of performance by skipping a number of windows.

No. windows	Ensemble calls (%)	Acc. (%)	Energy (J)
0	100	88.93	36.1
1	52.6	86.85	19.9
2	37.02	85.81	15.3
3	28.72	85.12	12.2
4	24.22	83.39	9.2
5	21.45	84.43	8.0
6	19.03	77.85	7.8
7	17.65	76.82	7.3
8	16.61	75.78	6.8
9	15.22	78.55	6.4
10	14.53	76.12	6.1

calls the complex classifier approximately in 20% of their windows, according to Table 5.15 and

using a maximum of 5 windows. *Standing* is the activity with more Ensemble calls, with 37.5% of calls. This happens because the activity *Standing*, a static activity, is usually wrongly classified as a dynamic activity, 9 times out of 33, as verified in Table 5.17. Thus, the confusion between static and dynamic activities leads to more calls of the complex classifier.

Table 5.16: Number of skips of complex classifier per activity using Rule-based classifier.

Activities	**No. Skips**
Running	6 out of 8 predicted
Walking	35 out of 42 predicted
Cycling	22 out of 27 predicted
Nordic walking	25 out of 30 predicted
Standing	15 out of 24 predicted
Lying	23 out of 28 predicted
Ascending stairs	20 out of 24 predicted
Descending stairs	10 out of 13 predicted
Vacuum cleaning	25 out of 30 predicted
Ironing	30 out of 42 predicted
Rope jumping	3 out of 4 predicted
Sitting	13 out of 17 predicted

Table 5.17: Classification of activities as dynamic or static. The number of times an activity was wrongly classified appears in red.

Activities	**Dynamic**	**Static**
Lying	4	28
Sitting	2	13
Standing	9	24
Ironing	34	2
Vacuum_cleaning	27	0
Ascending_stairs	22	0
Descending_stairs	15	0
Walking	42	0
Nordic_walking	36	0
Cycling	27	1
Running	3	1

Table 5.18 presents a comparison between the performances of the different base classifiers and the rule-based classifiers using 2 models (1 static and 1 dynamic). It is also present the improvement of the accuracy using only occurrences filter since previous experiences showed that this filter is better than the transitions-2 filter.

The ensemble of the three algorithms consumes more energy than the single classifiers, and comparing the single classifiers, kNN is the one that consumes more energy followed by Hoeffding tree and finally by Naive Bayes. Another obvious conclusion is that the ensemble of the three base classifiers is not worth it since by only using kNN, it is possible to achieve similar accuracies, lower classification time and lower energy consumption. Also, with the rule-based classifier +

Table 5.18: Comparison between classifiers.

Classifier	Acc. (%)	Acc. with filter (%)	M. classification time (ms)	Energy (J)
kNN	91.38	96.9	133	55.3
Naive Bayes	91.03	95.52	1.4	1.4
Hoeffding tree	91.03	95.52	1.6	1.5
Ensemble	91.03	95.52	142	55.6
Rule-based + kNN	88.24	89.62	37	8.3 J
Rule-based + Naive Bayes	84.43	85.81	1.6	0.98
Rule-based + Hoeffding tree	84.43	85.81	1.6	0.99
Rule-based + Ensemble	84.43	85.81	39	8.4

kNN, it is achieved almost 90% accuracy and the classification time and energy consumption are much lower using compared to the base kNN.

Finally, some scenarios were analysed, namely, the most efficient configuration for a system that requires a specific minimum accuracy, using rule-based + kNN or simply the base kNN (Table 5.19 and Table 5.20). Considering the rule-based + kNN, by decreasing the number of features to 66 and using the occurrences filter, it is possible to achieve 90% accuracy and the energy consumed is 2 Joule lower.

Consider the scenario for 75% accuracy the first scenario, and the scenario for 70% accuracy the second one. The first scenario requires only eight features, six of the eight features were acquired using the gyroscope and two using the magnetometer. The second scenario requires five features, four were acquired using gyroscope and one using the magnetometer. In an attempt to reduce the number of sensors used and consequently the energy consumption during the data acquisition step, the magnetometer features were replaced by another gyroscope features. This way, replacing the magnetometer features with others from the gyroscope, the first scenario achieved an accuracy of 44.98 % and the second scenario 33.22 %. Although the impact of the number of sensors on the energy consumption is not studied in this dissertation, it is concluded that reducing the number of sensors used is not worth since the accuracy decreased significantly.

Table 5.19: Best system configuration for a specific scenario using rule-based + kNN or only kNN without using filter.

Scenario	Acc. (%)	No. features	Sensors	F. Selection	Energy (J)
Rule-based + kNN	88.24	90	acc, gyr, mag	-	8.3
Maximum accuracy	88.24	61	acc, gyr, mag	Cfs	5.98
80 % accuracy	83.74	30	acc, gyr, mag	Cfs	2.6
75 % accuracy	76.82	8	gyr, mag	Cfs	1.5
70 % accuracy	71.28	5	gyr, mag	Cfs	1.5

Next, it was tested the impact of using only one model in the rule-based classification comparing it to using two models. Table 5.21 compares the performance of rule-based + kNN with one and two models. As expected the accuracy is lower while using 2 models since when the rule-classifier misses between classifying a dynamic or static activity, the base classifier always gives a wrong prediction. The energy consumption decreases from 13.75J to 8.3J. In this case the

Table 5.20: Best system configuration for a specific scenario using rule-based + kNN and using occurrences filter.

Scenario	Acc. (%)	No. features	Sensors	F. Selection	Energy (J)
Rule-based + kNN	89.62	90	acc, gyr, mag	-	8.3
Maximum accuracy	90.31	66	acc, gyr, mag	ReliefF	6.3
80 % accuracy	85.47	30	acc, gyr, mag	Cfs	2.6
75 % accuracy	78.55	8	gyr, mag	Cfs	1.5
70 % accuracy	73.01	5	gyr, mag	Cfs	1.5

accuracy achieved using 2 models is still high and it is possible to save more energy, but other systems that privilege accuracy, using 1 model is a better choice.

Table 5.21: Comparison of performance of rule-based + kNN with one or two models.

Rule-based + kNN	Acc. without filter (%)	Acc. with filter (%)	Energy (J)
1 model	92.39	93.08	13.75 J
2 models	88.24	89.62	8.3 J

Finally, instead of using a simple rule classifier, it was tested the performance of using a binary Hoeffding tree to classify dynamic and static activities and two kNN models. Tables 5.22 and 5.23 show the performance of the system using this binary Hoeffding tree together with kNN, for the scenarios specified.

Comparing the results of this approach with the Rule-based + kNN, the energy consumption is almost the same, although the accuracy is approximately 1% better with the same number of features without filter. Considering the use of the filter, the accuracy is 2% better compared to the previous approach. It is, also worth noting that, when using filter, it is possible to achieve 80% accuracy and the number of features needed is only 8. For the same scenario, the rule-based + kNN needs 30 features, consumes 1.1 more Joule and needs accelerometers.

Table 5.22: Best system configuration for a specific scenario using binary Hoeffding tree and kNN and without filter.

Scenario	Acc. (%)	No. features	F. Selection	Energy (J)
Binary Hoeffding tree + kNN	89.27	90	-	8.4
Max. accuracy	89.27	60	Cfs	5.6
80 % accuracy	84.77	30	Cfs	2.5
75 % accuracy	77.85	8	Cfs	1.5
70 % accuracy	71.97	5	Cfs	1.5

5.7 Summary

This Chapter shows the performance of the methods and techniques implemented with the aim of reducing the energy consumption and the classification time while keeping an acceptable accuracy.

Table 5.23: Best system configuration for a specific scenario using binary Hoeffding tree and kNN and using occurrences filter.

Scenario	Acc. (%)	No. features	F. Selection	Energy (J)
Binary Hoeffding tree + kNN	92.38	90	-	8.4
Max. accuracy	92.38	60	Cfs	5.6
80 % accuracy	80.97	8	Cfs	1.5
75 % accuracy	75.09	5	Cfs	1.5
70 % accuracy	70.93	3	Cfs	1.5

The first experiments used the data of the 9 users from Pamap2 and study the impact of the filter, the sampling rate, the number of features and the rule-based classifier. The results show that decreasing the sampling rate from 100Hz to 2Hz allows to reduce the feature extraction time from 20 to 0.85 milliseconds. Using 2Hz, the accuracy does not decrease significantly, exluding the case of kNN (decrease of 5%). Also, reducing the number of features of the system, using feature selection techniques, allows to get faster classification times. The maximum accuracy using Ensemble of kNN, Naive Bayes and Hoeffding tree is obtained with 76 features. Also, it is possible to get 80% accuracy using only 29 features and in this case the classification time decreases from 152 to 42 milliseconds at a cost of 1% accuracy comparing to using the initial 90 features. The occurrences filter showed to be a success, allowing to get accuracies of 88.32% using only 18 features. It is also worth noting that Cfs got a better performance comparing to ReliefF when selecting the best features. The rule-based classifier got good results too, achieving an accuracy of 76.51% with 90 features and a classification time of 73 milliseconds. Comparing to the base Ensemble classifier it has an accuracy 5% lower although the classification time is 80 milliseconds lower.

Then, some experiments were replicated using only one user (user 7) and the results were similar. Thus, the occurrences filter improves the accuracy significantly and with a low number of features is possible to achieve acceptable accuracies with lower classification times. In this case, the kNN classifier got a better performance than the Ensemble. Also, using the Rule-based + kNN with 90 features, the energy consumption decreases from 55.3 to 8.3 Joule, when comparing with the base kNN 90 features. The cost of the accuracy is not too high, it decreases approximately 3%, from 91.38% to 88.24%. Finally it was tested the performance of the binary Hoeffding tree + kNN and the energy consumption comparing to the rule-based + kNN is almost the same. However, the accuracies achieved with the Binary Hoeffding tree + kNN are higher. Hence, the performance of this classifier is better than the rule-based classifier in every aspect.

Chapter 6

Conclusion

6.1 Concluding Remarks

This dissertation presented our efforts to make feasible the implementation of a HAR system on a smartphone, by achieving real-time classification and lower energy consumptions with acceptable accuracies. We focus on three main techniques to reach our goals: reduce the number of features by using feature selection algorithms, a classifier that avoids the use of more complex classifiers in every sample of data (using a rule classifier or a binary hoeffding tree), and three types of filters post-classification to improve the accuracy of the system. With these techniques, we were able to improve an existent HAR system in many ways.

The dataset used (Pamap2) was collected using a sampling rate of 100Hz. However, we re-sampled the data to different frequencies. Using a sampling rate of 2Hz, the accuracy of the base classifiers was not affected too much, excepting kNN (from 83.04% to 77.84%). The feature extraction time also decreased from 20 to 0.85 milliseconds since we used a window with duration of 9 seconds and the number of samples per window decreased from 900 to 18. This decrease of the sampling rate results in an energy saving during the data acquisition phase since the sensors work at a lower frequency, although this was not measured.

We propose 3 types of filters: transitions-1, transitions-2 and occurrences filter. The transitions-1 filter is the simplest, it only evaluates the possibility of a transition between the current activity and the previous and if it is not possible it keeps the previous activity. The transitions-2 evaluates the possibility of transitions between the current and the previous activity or the current and the next activity within a window of 5 activities. If there is an impossible transition it changes the current activity to the activity with the most occurrences within that window. The occurrences filter does not evaluate the possibility of transitions, it changes the current activity to the activity with most occurrences withing a window of 5 activities. The experiments with these filters post-classification, showed it improved the accuracy by at least 1% using transitions-2 and occurrences filter in every experiment and in some cases it improved the accuracy by more than 4%. The occurrences filter showed to be better than the transitions-2 and the transitions-1 did not have a positive impact on the accuracy of the system and it was discarded after the first experiments. However,

the users in Pamap2 collected each activity separately which does not present scenarios implying possible wrong classifications between transitions of activities.

Then, we decreased the number of features used in the existent HAR system (90), using ReliefF [RSK03] and Cfs [AH00], and it led to a linear decrease in classification time and energy consumption. In some cases, we achieved better accuracies with less features, and globally lower classification times and lower energy consumption, showing that the base system uses many redundant features. It is also important to note that we achieved higher accuracies using Cfs comparing to ReliefF.

To classify static and dynamic activities we use a rule-based classifier using only one feature (mean of variances) and compare it with a selected threshold found empirically. Another approach to classify static and dynamic activities is to use a hoeffding tree with the same feature. When using the rule-based classifier we define a maximum number of skips of the complex classifier allowing us to reduce the complexity of the system. With this approach we can reduce the calls of complex classifier to 21.45%, skipping a maximum of 5 windows the complex classifier. The rule-based classifier allowed to reduce the energy consumption of the system significantly, and still achieved high accuracies (90% with kNN, 66 features and occurrences filter). In some cases, to achieve 75% and 70% accuracy it was only needed 8 (2 features from magnetometer and 6 from gyroscope) and 5 features (1 features from magnetometer and 4 from gyroscope). In an attempt to reduce the number of sensors used and consequently the energy consumption during the data acquisition step, the magnetometer features were replaced by another gyroscope features, however the accuracy obtained was too low (44.98% using 8 features or 33.22% using 5 features). The binary Hoeffding tree + kNN achieved higher accuracies than the rule-based + kNN (92.38% and the energy consumptions were almost the same, using the same number of features. Another advantage of using the binary Hoeffding tree as a rule classifier is that we do not need to find a threshold value empirically. Comparing to the baseline kNN with 90 features (91.38 % accuracy), using the binary Hoeffding tree + kNN we achieved an accuracy of 89.27%. Although the accuracy decreased approximately 2%, the energy consumption decreased from 55.3 Joule to 8.4 Joule, showing that this method can achieve significant energy savings and still achieve high accuracies. The maximum accuracy (92.38%) was obtained with 60 features selected by Cfs and the energy consumed was 5.6 Joule.

6.2 Future Work

The methods and techniques implemented proved to improve the performance of the base HAR system, although the dataset tested may not be representative of some real scenarios. Some users performed different numbers of activities, and the fact that the activities were trained separately may interfere with the results of the filter. Hence, we should define a protocol to collect our own dataset.

Future work plans shall consider a more sophisticated method to possibly dynamically select the number of windows for which the system may avoid the use of more complex classifiers, based

on the current activity.

We should also consider the use of kNN using LSH focused in [Mag19] and evaluate the performance of the rule-based classifier proposed with kNN + LSH .

Another possible improvement could be to implement dynamic segmentation, instead of using windows with fixed length and to test its impact on the performance of the system.

When using the rule-based classifier it is possible to reduce the number of sensors between some activities since we only need the features from the gyroscope when the complex classifier is not called. Thus, it is important to study the impact on the energy consumption of turning the sensors off during the skips of the complex classifier.

Finally, one of the reasons to use the MOA library was its implementation of incremental/online Machine Learning algorithms. Thus, future work shall improve the current Android application implementing incremental/online learning.

Conclusion

References

[AH00] Mark Andrew. Hall. Correlation-based feature selection for machine learning. *Department of Computer Science*, 19, 06 2000.

[Alp10] Ethem Alpaydin. *Introduction to Machine Learning*. Massachusetts Institute of Technology, second edition, 2010.

[AMD+15] Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. Physical human activity recognition using wearable sensors. *Sensors*, 15:31314–31338, 12 2015.

[AT14] O. C. Ann and L. B. Theng. Human activity recognition: A review. In *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*, pages 389–393, Nov 2014.

[BBS13] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46, 01 2013.

[BHKP10] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

[BI] R. Berwick and Village Idiot. An idiot's guide to support vector machines (svms). Available at http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf.

[BKA+03] Najafi B, Aminian K, Paraschiv-Ionescu A, Loew F, Bula CJ, and Robert P. Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly. *Biomedical Engineering, IEEE Transactions*, pages 711–723, 2003.

[Bon18] Giuseppe Bonaccorso. *Mastering machine learning algorithms*. Packt Publishing - ebooks Account, 2018.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[CL] Chih-Chung Chang and Chih-Jen Lin. Libsvm toolbox. Available at https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[CMV18] João Cardoso, João Moreira, and Luís Veiga. Contextwa - middleware and context inference techniques from data-streams for the development of context-aware services using mobile devices. *Impact*, 2018:18–20, 03 2018.

REFERENCES

[CWZ+17] Liang Cao, Yufeng Wang, Bo Zhang, Qun Jin, and Athanasios Vasilakos. Gchar: An efficient group-based context-aware human activity recognition on smartphone. *Journal of Parallel and Distributed Computing*, 118, 05 2017.

[ECN+10] McAdams ET, Gehin C, Noury N, Ramon C, Nocua R, Massot B, Oliveira A, Dittmar A, Nugent CD, and McLaughlin J. Biomedical sensors for ambient assisted living. *Advances in Biomedical Sensing, Measurements, Instrumentation and Systems.*, pages 240–262, 2010.

[Elk98] Charles Elkan. Naive bayesian learning. 12 1998.

[FDFC10] Davide Figo, Pedro C. Diniz, Diogo R. Ferreira, and João M. P. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7):645–662, Oct 2010.

[Foua] Eclipse Foundation. Eclipse ide. Available at https://www.eclipse.org/ide/.

[Foub] Python Software Foundation. Python language reference, version 3.7.3. Available at https://www.python.org/.

[GCM+18] Kemilly Dearo Garcia, Tiago Carvalho, João Mendes-Moreira, João M. P. Cardoso, and André C. P. L. F. de Carvalho. A preliminary study on hyperparameter configuration for human activity recognition. *CoRR*, abs/1810.10956, 2018.

[GLG+18] Eva García-Martín, Niklas Lavesson, Håkan Grahn, Emiliano Casalicchio, and Veselka Boeva. Hoeffding trees with nmin adaptation. *CoRR*, abs/1808.01145, 2018.

[GLH14] Salvador García, Julián Luengo, and Francisco Herrera. *Data Preprocessing in Data Mining*. Springer, 2014.

[Goo] Google. Android sdk platform. Available at https://developer.android.com/studio/releases/platforms.

[Har] Hardkernel. Odroid-xu. Available at https://archlinuxarm.org/platforms/armv7/samsung/odroid-xu.

[HUMA18] Mohammed Mehedi Hassan, Md. Zia Uddin, Amr Mohamedc, and Ahmad Almogren. A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems*, pages 307–313, April 2018.

[IT] INESC-TEC. Contextwa project. Available at https://www.inesctec.pt/en/projects/contextwa#intro.

[JM] Daniel Jurafsky and James H. Martin. Hidden markov models. Available at https://web.stanford.edu/~jurafsky/slp3/A.pdf.

[K98] Murphy K. Hidden markov model (hmm) toolbox for matlab. Available at http://www.ai.mit.edu/~murphyk/Software/HMM/hmm, 1998.

[KIR+05] A. Krause, M. Ihmig, E. Rankin, D. Leong, Smriti Gupta, D. Siewiorek, A. Smailagic, M. Deisher, and U. Sengupta. Trading off prediction accuracy and power consumption for context-aware wearable computing. In *Ninth IEEE International Symposium on Wearable Computers (ISWC'05)*, pages 20–26, Oct 2005.

REFERENCES

[KZ09] Quansheng Kuang and Lei Zhao. A practical gpu based knn algorithm. *International Symposium on Computer Science and Computational Technology (ISCSCT)*, 01 2009.

[Lab13] WISDM Lab. Wisdm dataset. Available at http://www.cis.fordham.edu/wisdm/dataset.php, 2013.

[MA17] Jafet Morales and David Akopian. Physical activity recognition by smartphones, a survey. *Biocybernetics and Biomedical Engineering*, 37(3):388 – 400, 2017.

[Mag19] Ricardo Manuel Correia Magalhães. Energy Efficient Smartphone-based Users Activity Classification. Master's thesis, 2019.

[Mic06] Sun Microsystems. Jsr 82 bluetooth api and obex api. Available at https://docs.oracle.com/javame/config/cldc/opt-pkgs/api/bluetooth/jsr082/index.html, 2006.

[Mit97] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[MPP09] Antonio Mucherino, Petraq J. Papajorgji, and Panos M. Pardalos. *k-Nearest Neighbor Classification*, pages 83–106. Springer New York, New York, NY, 2009.

[MR15] Awad M. and Khanna R. Hidden markov model. *Efficient Learning Machines*, pages 81–104, 2015.

[NPCN16] Qin Ni, Timothy Patterson, Ian Cleland, and Chris Nugent. Dynamic detection of window starting positions and its implementation within an activity recognition framework. *Journal of Biomedical Informatics*, 62:171 – 180, 2016.

[Ora19] Oracle. Jsr 256: Mobile sensor api. Available at https://jcp.org/en/jsr/detail?id=256, 2019.

[PF07] Sang-Hyeun Park and Johannes Fürnkranz. Efficient pairwise classification. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Machine Learning: ECML 2007*, pages 658–665, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[PGY+11] Chunmei Pei, Huiling Guo, Xiuqing Yang, Yangqiu Wang, Xiaojing Zhang, and Hairong Ye. Sensors in smart phone. In Daoliang Li, Yande Liu, and Yingyi Chen, editors, *Computer and Computing Technologies in Agriculture IV*, pages 491–495, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Pol12] Robi Polikar. *Ensemble Learning*, pages 1–34. Springer US, Boston, MA, 2012.

[Por09] A. Portela. *Inferência de contexto para aplicações móveis*. PhD thesis, IST Technical University of Lisbon, 2009.

[Qui86] J. R. Quinlan. *Induction of decision trees*. Kluwer Academic Publishers, 1986.

[Qui93] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

REFERENCES

[RS12] Attila Reiss and Didier Stricker. Creating and benchmarking a new dataset for physical activity monitoring. *ACM International Conference Proceeding Series*, 06 2012.

[RSK03] Marko Robnik-Sikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, 53:23–69, 10 2003.

[Sai] Taro L. Saito. Sqlite jdbc. Available at https://bitbucket.org/xerial/sqlite-jdbc.

[SAL+16] Susanna Spinsante, Alberto Angelici, Jens Lundstrom, Macarena Espinilla, Ian Cleland, and Christopher Nugent. A mobile application for easy design and testing of algorithms to monitor physical activity in the workplace. 2016.

[Sam59] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, July 1959.

[SCF+18] André C. Santos, João M.P. Cardoso, Diogo R. Ferreira, Pedro C. Diniz, and Paulo Chaínho. Providing user context for mobile and social networking applications. *Pervasive and Mobile Computing*, pages 324–341, 2018.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[UAM+05] Lindemann U, Hock A, Stuber M, Keck W, and Becker C. Evaluation of a fall detector based on accelerometers: A pilot study. *Medical and Biological Engineering and Computing*, pages 548–551, 2005.

[WCH+18] Jindong Wang, Yiqiang Chen, Lisha Hu, Xiaohui Peng, and Philip S. Yu. Stratified transfer learning for cross-domain activity recognition. *CoRR*, abs/1801.00820, 2018.

[WLA+09] Y. Wang, J. Lin, M. Annavaram, Q.A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. *In Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, pages 179–192, 2009.

[XFYTCP08] Teng X-F, Zhang Y-T, Poon CC, and Bonato P. Wearable medical systems for p-health. *Biomedical Engineering, IEEE Reviews*, pages 62–74, 2008.

[YC17] C. Shen Y. Chen. Performance analysis of smartphone-sensor behavior for human activity recognition. *IEEE Access 5*, pages 3095—-3110, 2017.

[YSC+12] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. *2012 16th International Symposium on Wearable Computers*, pages 17–24, 2012.

[ZWR+17] Lingxiang Zheng, Dihong Wu, Xiaoyang Ruan, Shaolin Weng, Ao Peng, Biyu Tang, Hai Lu, Haibin Shi, and Huiru Zheng. A novel energy-efficient approach for human activity recognition. *Sensors*, 17:20–64, 09 2017.