FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Predicting Activities from Smartphones

Hugo Louro Cardoso

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Mendes Moreira

Co-Supervisor: João Gama

June 27, 2016

Predicting Activities from Smartphones

Hugo Louro Cardoso

Mestrado Integrado em Engenharia Informática e Computação

June 27, 2016

Abstract

Built-in hardware sensors in many of the modern smartphones, such as accelerometers and gyroscopes, open a world of multiple opportunities for novel applications based on the context perceived from the data they provide. Human activity recognition (HAR) is a direct application of this technology, which despite being a very active field of study in the past years, leaves many strategies left to explore and key aspects left to address. A commonly ignored challenge of HAR is the difference of input signals produced by different people when doing the same activities. As a result, the activity classification method should be able to generate adapted results for each different user. This document proposes and explores a solution to this problem by means of "Online Semi-supervised Learning", an underexplored incremental approach capable of adapting the classification model to the user of the application by continuously updating it as the data from the user's own specific input signals arrives. The ideal scenario for this technology would be the creation of a smartphone application capable from the beginning of classifying the user's activities with a certain error, and as the time passes and the user utilizes the application, without manual input, the system's classification error would decrease autonomously until it is virtually insignificant for that specific user. The success of this approach would result in numerous applications, and could considerably enhance the current interaction between people and their mobile devices, taking the concept of "smartphone" to a whole new level. Several classification models were generated, evaluated and compared, with different online semi-supervised approaches. In the end, we achieved a cross-validated average accuracy increase of 0.18% resulting in a 82.76% accuracy model with Naive Bayes, 0.14% accuracy increase resulting in a 83.03% accuracy model with a Democratic Ensemble, and 0.08% accuracy increase resulting in a 84.63% accuracy model with a Confidence Ensemble. These models were capable of detecting 3 stationary activities, 3 active activities, and all transitions between the stationary activities, to a total of 12 distinct activities.

Resumo

Os sensores embutidos na maior parte dos smartphones modernos, tal como acelerómetros e giroscópios, abrem um mundo de múltiplas possibilidades para um novo tipo de aplicações baseadas no contexto adquirido a partir dos dados que fornecem. O Reconhecimento de Actividades Humanas (HAR) é uma aplicação directa desta tecnologia, que apesar de ser uma área de estudo bastante activa nos últimos anos, ainda possui diversas estratégias não exploradas e aspectos-chave não correspondidos. Um desafio de HAR normalmente ignorado é a diferença de sinais produzidos por diferentes pessoas enquanto praticam as mesmas actividades. Assim, o sistema de classificação de actividades deveria ser capaz de gerar resultados adaptados para cada utilizador individual. Este documento propõe e explora uma solução para este problema à base de Aprendizagem Semi-supervisionada Online, uma abordagem incremental pouco explorada capaz de adaptar o modelo de classificação ao utilizador da aplicação ao actualiza-lo continuamente à medida que os dados específicos gerados pelo utilizador entram no sistema. O cenário ideal para esta tecnologia seria a criação de uma aplicação para smartphone capaz de, logo de raiz, ser capaz de classificar as actividades de um novo utilizador, com um certo erro, e à medida que o tempo passa e o utilizador usa a aplicação, sem qualquer tipo de instruções manuais, o erro do sistema de classificação diminuiria autonomamente, até ser virtualmente insignificante para esse utilizador específico. O sucesso desta abordagem resultaria em numerosas aplicações, e poderia aumentar consideravelmente a interacção actual entre as pessoas e os seus dispositivos móveis, elevando o conceito de "smartphone" a um nível nunca antes alcançado. Vários modelos de classificação foram gerados, avaliados e comparados, com diferentes abordagens de aprendizagem semi-supervisionada. No fim, foram alcançados uma média de aumento de precisão, adquirida através de validação cruzada, de 0.18%, resultando num modelo com precisão de 82.76% com Naive Bayes, um aumento de precisão de 0.14% resultando num modelo com precisão de 83.03% com Democratic Ensemble, e um aumento de precisão de 0.08% resultando num modelo com precisão de 84.63% com Confidence Ensemble. Estes modelos eram capazes de detectar 3 actividades estáticas, 3 actividades móveis, e todas as transições entre as actividades estáticas, para um total de 12 actividades distintas.

Acknowledgements

Immense appreciation and deepest gratitude for the help and support are extended to the following individuals who, in one way or another, have contributed in making this study possible and enjoyable.

João Moreira, my supervisor, for proposing such an interesting subject, and for the assistance in times of struggle.

Graça Louro and David Cardoso, my parents, who paved the path in which I stand today, giving me all the possibilities I could ever ask for.

Mariana Mendes for her love, support, and for being a constant source of joy throughout the entirety of the process.

FEUP, my faculty, for the comfortable and productive environment, and for teaching me, above all, the power of knowledge and how to love learning.

Hugo Louro Cardoso

"To combine is to create,
To engineer, divine."

Orphan Black

Contents

List of Figures

LIST OF FIGURES

Abbreviations

HAR	Human Activity Recognition
KNN	K-Nearest Neighbors
SL	Supervised Learning
SSL	Semi-Supervised Learning

Chapter 1

Introduction

The study of the human body and its anatomy has always been an area of great interest.

Every year new ways of using the human body to interact with technology take shape, as a way of suiting the desire of converting these complex mechanics into input for novel applications.

The creation of sensors was a huge step forward into making a reality out of this desire. Sensors, by themselves, have been around for quite some time, in various forms. The first known ones came to market in the 80's, and have since suffered an exceptional development, from functionality and speed to astonishingly reduced sizes.

The employment of these vastly improved sensors in the human body turned out to be quite effective. They allowed for the collection of extremely convenient data, especially regarding body movement.

The computer science community saw the transformation of this data into knowledge as a vessel of potential. To infer which specific activity was an individual performing, simply from the analysis of its motion data, would become one of the most promising approaches of an area which is nowadays known as Human Activity Recognition, which despite being a very active field of study in the past years, still leaves many strategies left to explore and key aspects left to address.

The introduction of built-in hardware sensors in many of the modern smartphones, such as accelerometers and gyroscopes, unlocked a very interesting reality. The existence of such complex sensors in everybody's pocket allows for the study of the human body, as well as the creation of applications based on the context perceived from the data they provide, in a way so vast that it could never have been envisioned a decade ago.

Furthermore, these smartphones have more than the necessary computation power to process this data in real time, and they are extremely convenient in terms of suitability and familiarity, in contrast with the first multi-sensor systems, which had the huge disadvantage of being strange to the user, in a way that it would be unpractical or even uncomfortable for an everyday user to wear, due to its sizes and shapes.

Although an exceptional, almost ideal, environment has been laid out for the integration of this technology into our everyday lives, the field's technical aspects are still a great challenge for the computer science community.

From portability and obtrusiveness to energy consumption, processing needs and system overall cost, HAR systems must take into account several aspects, all dependent on how ambitious the system is in terms of the number of activities it wishes to recognize.

A commonly ignored challenge of HAR is the difference of input signals produced by different people when doing the same activities. It is no surprise that two individuals may, for example, walk or run in completely different fashions, which will consequentially feed contradictory input into the system, causing it to output an erroneous classification if it does not take into account this irrevocable fact.

As a result, a new, more complex challenge appeared. It is only natural that, if each individual produces different inputs for the same activity, then the activity classification method should be able to generate adapted results for each different user.

The success of this approach would result in numerous applications. For instance in the field of medicine by recognizing the act of falling in order to provide immediate support for the elderly people, or continuously monitoring people's daily activities and life style for health reasons. Even in sports, for providing information on how an athlete's posture could be improved, very much like a personal trainer would do, or simply by accurately logging data of each particular exercise, without the need for the user to manually input which exercise will be performed.

These are just a few examples out of several applications that spawn from the knowledge provided by the user's activity recognition. This technology could considerably enhance the current interaction between people and their mobile devices, possibly taking the concept of "smartphone" to a whole new level.

1.1 Background and Basic Concepts

As stated above, although several advances were made in the area of Human Activity Recognition, it is still a field of great challenge, with no perfect approach.

Until now, one of the most basic approaches to HAR was through computer vision, that is, through the analysis of a video recording of a human subject performing several activities. Unfortunately, not only are these systems extremely error-prone, but they have as a principle that the user will be recorded at all times, which is simply not practical if the final goal is to use this technology as an everyday tool.

On the other hand, the approach of analyzing sensor data from a smartphone is very practical, but to create a system which is able to adapt to its user is easier said than done.

To understand the means by which we can achieve such a system, first we need to understand a few basic concepts.

All of the mentioned approaches rely on the basic principle of Data Mining, which is no more than the discovery of new patterns in large data sets, with the overall goal of extracting knowledge and transform it into a human-understandable structure for further application.

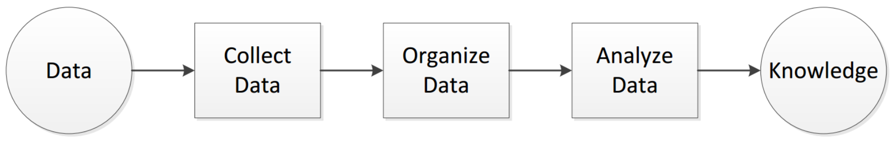


Figure 1.1: Data Mining Diagram

Most common Data Mining approaches make use of static datasets. These datasets are collected and organized a priori, and only afterwards analyzed and processed. They also have the huge advantage of being traditionally labeled, in other words, the data's context is known. For example, data of someone running, or walking.

This labeled data is massively used by Supervised Learning techniques. These Machine Learning approaches analyze the data and generate a model, capable of determining the class labels for unseen instances.

Systems which perform a single training phase on a static dataset and whose models do not change after such phase, are classified as offline.

As interesting as these concepts are, the resulting systems can only be generic. If applied in the intended HAR application, while they might yield decent results using static datasets, gathered from a sample population, they are far from perfect, since as a specific user's movements drift away from the generic, the classification gets worse.

Therefore, the need to explore different approaches flourishes. We need to take adaptation into account.

This is where the concept of Online Learning appears as a riveting method. The data enters the system sequentially, and the system learns from it also in a sequential manner. At each step, it uses the new data to update its classifier for future instances, dynamically adapting to new patterns.

In practice, the smartphone would feed the application with a data stream from its sensors, and the application, instead of simply classifying it with a pre-trained offline model, it would use this data to improve, to adapt to its user.

Yet, data coming directly from the smartphone's sensors wouldn't be labeled. It is not possible to exploit the conventional supervised learning approaches.

This ideology turns a different, quite underexplored approach into a very possible solution. All these issues might be addressed by means of "Online Semi-supervised Learning".

Semi-supervised learning has the particularity of operating on both labeled and unlabeled data, typically a small amount of labeled versus a large amount of unlabeled one.

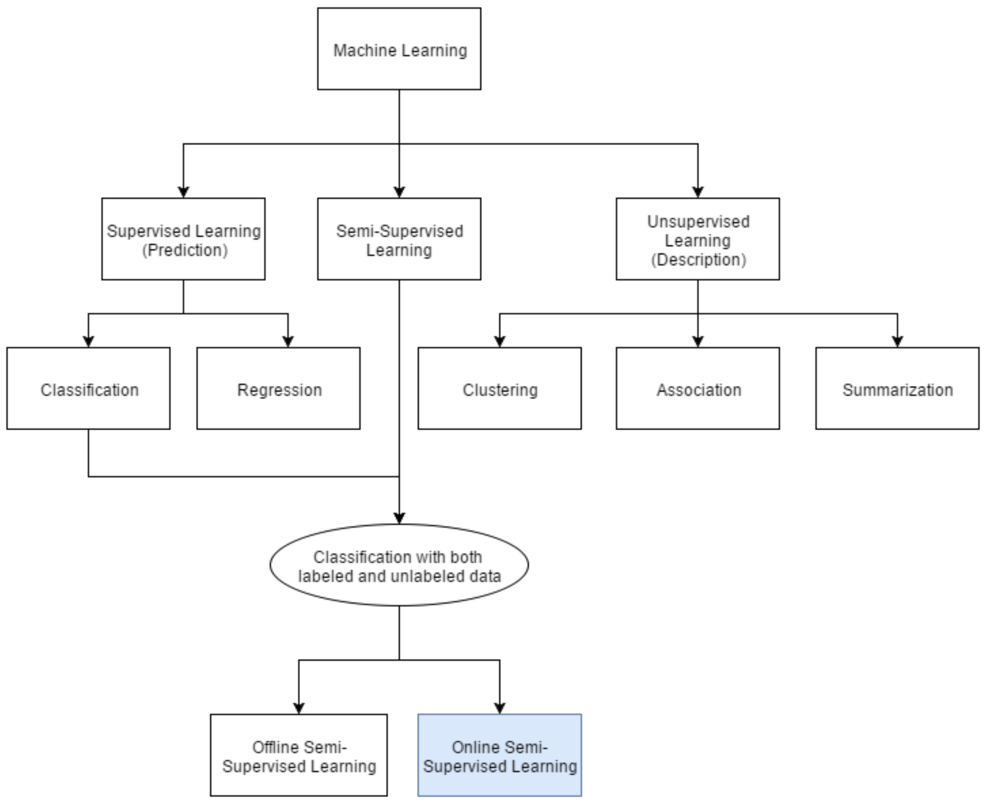

Machine Learning
Supervised Learning (Prediction)
Semi-Supervised Learning
Unsupervised Learning (Description)
Classification
Regression
Clustering
Association
Summarization
Classification with both labeled and unlabeled data
Offline Semi-Supervised Learning
Online Semi-Supervised Learning

Figure 1.2: Online Semi-supervised Learning in Machine Learning

The fact is that it is many times easier to acquire unlabeled data than labeled one. Labeled data often requires a skilled human agent, that is, an entity which can accurately classify the data. On the contrary, the acquisition of unlabeled data is relatively inexpensive.

This incremental approach is capable of adapting a classification model, pre-trained by means of supervised learning, to the user of the application by continuously updating it as the data from the user's own specific input signals arrives.

It is, therefore, a promising approach to solving the problem in question. If the goal HAR application learns in an online, semi-supervised fashion, it could start off as generic, but iteratively adapt to a new user, instead of being bound by a generic model, taught by static, pre-gathered data.

Still, there are a few obstacles that might hinder the development process. Not only are we restrained by smartphones' technical limitations, we also have to deal with phenomena like concept drift, which consists of the unforeseen change over time of the statistical properties of the target variable which a model is trying to predict, causing the predictions to become less accurate as time passes.

1.2 Project

The main objective of this project is to effectively produce a HAR application by means of Online Semi-supervised Learning, a perfectly logical and promising approach which is extremely underexplored, despite answering most of the challenges in the field.

The ideal scenario would be a prototype, capable from the start of classifying its new user's activities, with a certain degree of error. As the user utilizes the application more and more, the error should reduce significantly, with the classification model adapting correctly to the user's input.

To achieve such a goal, a great deal of conceptual research, on several online semi-supervised learning algorithms, incremental classifiers, methods of evaluation, and other necessary techniques, will be performed.

After the initial research, a dataset of activity-labeled human motion will be acquired and analyzed, that is to be utilized throughout the whole development. The data will be pre-processed in order to begin with the empirical research, where the algorithms will be put into practice in order to generate classification models.

Continuous testing will be performed, in terms of model comparison and evaluation, with the goal of keeping track of the project's evolution and progress.

In the end, upon the decision of the best fitting model, conclusions will be drawn, and hopefully the project will finish on a positive note.

1.3 Motivation and Goals

The spread existence of complex sensors in mankind's pockets is a bewildering reality. To be able to utilize them to their fullest is, by itself, a huge drive.

This project allows for the opportunity to contribute with a step further on the HAR field, a technology of endless applications, which could benefit its users world-wide.

Smartphones could suffer a huge step forward, being able to adapt even more to its user's needs, very possibly improving the comfort of everyday life, which is an extremely compelling inspiration.

It is also very satisfying to be able to explore Online Semi-supervised Learning as a realistic and practical technique, to prove that it can be put into work in a real project and actually improve the results.

In the end, the creation and distribution of a public API would be an astonishingly fulfilling conclusion to this project, giving the computer science community the capability of putting this technology into practice to the limits of the imagination. The very idea of contributing to the community is delighting, since I myself made use of its contributions time and time again.

1.4 Document Structure

Besides the introduction, this document is divided into five chapters.

Chapter 2 presents the existing literature and research in the area of Machine Learning, with a concrete focus on data stream mining with semi-supervised learning. Sliding Windows, sampling techniques, incremental classifiers, model evaluation and several semi-supervised learning approaches are introduced in depth, as well as other useful techniques and tools. In the end, other works with the same goal are explored and exposed in terms of successes and failures.

Chapter 3 describes the work environment and methodologies, as well as the project's core architecture. An analysis of the utilized data set will be performed, as well as the different techniques and algorithms empirically applied.

Chapter 4 consists of the actual experiments performed, along with their goals, methodologies, results and conclusions, which will be taken from the resulting classification models from each experiment. The overall system quality and obtained accuracy are demonstrated and discussed.

Chapter 5 concludes the project with an overview of the results of the dissertation as a whole and a discussion on suggestions and ideas for future work.

Chapter 2

State of the art

To achieve the best possible results in this project, it is imperative that we first research what was already developed by other researchers in the field, with the same, similar, or as useful ambitions as ours. Hopefully, the harvested knowledge from those countless hours of work will aid into pushing HAR a step further.

Therefore, in this section, several techniques, algorithms, tools and any piece of useful information that might be manipulated into serving the project's goals will be presented and dissected.

However, this project will focus specifically on the subject of Online Semi-Supervised Learning. Great works exist in the field of HAR, like that of Bao & Intille [BI04], who created a system capable of recognizing twenty activities using decision trees and bayes classifiers.

Although their principles might serve as an astute basis, they will not be presented in this section, since they do not follow the ideology of online learning.

2.1 Data Segmentation and Sampling

Before we even consider starting this project, we must understand that all the data will be handled as streams.

By this, it is meant that data will enter the application sequentially, at all times, originating from the smartphone's sensors' real-time readings.

An activity, by itself, can't be represented by a single reading on the accelerometer and gyroscope's values. It is, instead, a pattern on a sequence, or bunch, of values. It is therefore necessary to explore methods of analyzing the data stream, to properly feed it to a model for training.

2.1.1 Sliding Window

A Sliding window is an effective way of extracting ordered, recent and relevant data from a data stream. It saves computing power and memory usage, because only the data window is used.

Most used approaches to a sliding window are of fixed length. This means that the window size is chosen in advance, and it never changes in runtime. There are two main types of fixed-length sliding windows. However, a dynamic-length window method will also be presented.

2.1.1.1 Non-Overlapping Sliding Window

Non-Overlapping Sliding Windows have the particularity of not sharing any data with consecutive windows. In the context of a stream, this means that if we receive 10 data samples and the window has size 5, the first window will have samples from 1 to 5, and the second from 6 to 10.

2.1.1.2 Overlapping Sliding Window

Unlike the above, Overlapping Windows share data with consecutive windows. The amount of shared data varies to satisfy the context needs. Extending the previous example, consecutive windows could possess samples from 1 to 4, 3 to 6, 5 to 8 and 7 to 10.

2.1.1.3 Dynamic Signal Segmentation

Aside from these very simple, yet most used approaches, there are other interesting alternatives. A very promising one is **Dynamic Signal Segmentation** [KLG11], which is a dynamic-length sliding window algorithm with the goal of dividing the data into intervals that ideally represent a single activity, avoiding the common disadvantage of fixed-length windows which sometimes possess a mixture of activities, making it hard to classify.

The method searches for a significant change between consecutive data samples and divides the data into intervals at that point. The significant change is defined as a sequence of consecutive data samples where the values are in descending order and the difference between the maximum and the minimum element in the sequence is larger than a threshold.

2.1.2 Data Sampling

Since time and space efficiency matter in the smartphone context, it is also important to cover data sampling, in case we end up not having the required resources to process all the data from the stream.

According to Babcock [BDM02], two capable methods are the sequence-based window, and the timestamp-based window. Both are promising and might serve our needs.

2.1.2.1 Sequence-based Window

The Sequence-based window consists of the N most recent data elements to arrive.

The standard algorithm for sampling with this window is to use Vitter's reservoir sampling techniques [Vit85]. It consists of maintaining a reservoir sample for the first N data elements in the stream, and thereafter to stop maintaining the sample except when the arrival of a new data element causes an element present in the sample to expire. In that case, the expired element is replaced with a newly-arrived element.

This algorithm maintains a uniform random sample over a window of the last N elements while requiring only enough memory to store the predetermined number of data elements.

Another simple algorithm is to add each new data element to a "backing sample", and generate the sample of size K by down-sampling from that sample. When an element expire it is removed from the backing sample.

From the desire of memory-efficiency, a different algorithm was born, the Chain-Sample. It consists of including each new element in the sample if it has a determinate probability. This new element will replace another element when it expires, and that same new element will also have a replacement when it arrives, building therefore a chain of potential replacements. When an element is chosen to be discarded from the sample, its chain is also discarded.

2.1.2.2 Timestamp-based Window

Timestamp-based Window consists of all data elements whose arrival timestamp is within a time interval T of the current time.

The best known algorithm is "priority sample", in which each arriving data element is assigned with a random priority. The non-expired element with higher priority is included in the sample. An element is ineligible if there is another element of a most recent timestamp and higher priority.

While choosing the best fit for our needs may require some testing, one thing can be taken for granted. These techniques allow our stream of data to be simplified and dissected for further processing. With this in mind, the next step is to understand in which way this processing might be tackled.

2.2 Classifiers

In the machine learning universe, classifiers are the mechanisms responsible for categorizing new observations, after training on known instances.

Classification is considered a form of prediction and an instance of Supervised Learning, since the training phase makes use of correctly identified observations (an example would be to distinguish between spam and non-spam email). When, instead, we refer to Unsupervised Learning, the procedure is known as description, for example as clustering, which involves grouping data into categories based on detected patterns, without actually knowing what each group of data represents.

In this project, however, Semi-supervised learning will be used, an approach which falls in-between the two mentioned methods. In this case, both labeled and unlabeled data are used, allowing the algorithms to perform classification. Since data will be read from a stream in real-time, the algorithms must be incremental to work properly in an online fashion.

Since it is the scope of this project, only the most promising incremental classifiers, able to be applied in the Semi-supervised Learning context, will be further presented.

Hulten and Domingos [DH00] identified desirable properties of learning systems for efficiently mining continuous, high-volume, open-ended data streams:

- Requires a small constant time per data example.
- Uses a fix amount of main memory, independent from the total number of examples.
- Builds a decision model using a single scan of the training data.
- Generates an anytime model independent from the order of the examples.
- Deals with concept drift.
- Lossless. In other words, for stationary data, it produces decision models that are nearly identical to the ones obtained from batch learners.

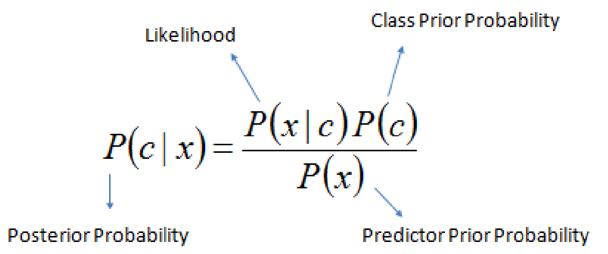
With these concepts in mind, let us now explore our options.

2.2.1 Naïve Bayes Online

The Naïve Bayes Classifier is one of the best known and most commonly used classifier in the data mining universe.

It is based on the Bayesian theorem and is particularly suited when the dimensionality of the inputs is high, but gets its name from its inability to capture correlations. It assumes independence between all features, which may not be ideal.

Despite its simplicity, it can often outperform increasingly sophisticated classification methods.



$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure 2.1: Bayes rule in Naïve Bayes Algorithm

Naïve Bayes can be seen as a frequency table between attribute-values and classes. For a new example, the probability of a class given that example can be calculated by applying Bayes theorem.

The likelihood of a class can be obtained by multiplying the frequency of the example attribute values that result in that class, on the seen data. This can be done because all the attributes are considered to be independent.

The class prior probability can be obtained by simply verifying how frequent has the class been outputted in the seen data.

Since the predictor prior probability is a constant, it can be ignored.

By multiplying these two values, in the end we obtain the relative probabilities of each class, given the new example. The class with more probability is more likely to be the true class of the example.

It is not only a simple, light algorithm, but it also only requires a small amount of training data to estimate the parameters necessary for classification, making it a great candidate in the context of this project, since smartphones have limited processing power and memory space.

It is also naturally incremental, as upon receiving a new example, it simply increments the relevant counts, and predictions can be made at any time from the current counts.

2.2.2 Decision Trees

Although the focus of this topic will boil down to the concept of Very Fast Decision Trees, an incremental decision tree induction algorithm capable of learning from massive data streams, to understand it we first need to understand what decision trees are.

A decision tree uses a divide-and-conquer strategy, which consists in dividing a complex problem into simpler problems and recursively apply the same strategy to sub-problems. Solutions of these sub-problems can then be combined into a solution of the complex problem.

This approach is particularly powerful due to the ability to fit each of the subspaces with different models, and due to its high degree of interpretability.

The decision tree's structure consists of a direct acyclic graph in which each node is either a decision node, usually containing some condition (based on attribute values) and two or more successors, or a leaf node, which is usually labeled with a class.

Each branch forms a rule with a conditional part and a conclusion. The conditional part is a conjunction of conditions. Therefore, the hypothesis space of decision trees is within the DNF (disjunctive normal form) formalism.

These conditions, or tests, correspond to a hyper-plane that is orthogonal to the axes of the tested attribute and parallel to all other axis. The regions produced by these classifiers are all hyper-rectangles. Each leaf corresponds to a region, and these regions are mutually exclusive and exhaustive (cover all the instance space).

2.2.2.1 Very Fast Decision Trees

The Very Fast Decision Trees algorithm is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time.

VFDT is based on the Hoeffding Trees algorithm, with a few enhancements, such as node-limiting strategy, introduction of a tie breaking parameter, grace period of bound calculation, poor attributes removal, fast initialization by using conventional RAM-based learners and the ability to rescan previously-seen examples when data rate is slow.

It considers that a small number of examples is enough to select the correct splitting test and expand a leaf. The algorithm makes a decision, that is, installs a split-test at a node, only when there is enough statistical evidence in favor of a split test.

In VFDT, a decision tree is learned by recursively replacing leaves with decision nodes. Each leaf stores the sufficient statistics about attribute-values, which are needed by a heuristic evaluation function to evaluate the merit of split-tests.

When a new example is available, it traverses the tree from the root to a leaf, evaluating the appropriate attribute at each node, and following the branch corresponding to the attribute's value in the example.

When the example reaches a leaf, the sufficient statistics are updated. Then, each possible condition based on attribute-values is evaluated. If there is enough statistical support in favor of one test over all the others, the leaf is transformed into a decision node.

The new decision node will have as many descendant leaves as the number of possible values for the chosen test. The decision nodes only maintain the information about the split-test installed in this node.

VFDT's main innovation is the use of Hoeffding bound to decide the sample size to observe before installing a split-test at each leaf. This effectively allows to decide if a leaf should be transformed into a decision node that splits on a certain attribute, or if more examples are needed to take a stable decision. The split test is based on the best attribute.

Since the evaluation of the merit function for each example could be very expensive, VFDT only computes the attribute evaluation function when a minimum number of examples has been observed since the last evaluation.

VFDT is capable of processing high-speed data streams of millions of examples using limited computational resources, with a performance similar to a batch decision tree. This is possible due to VFDT being I/O bound, that is, it mines examples in less time than it takes to input them from disk. It does not store any examples in main memory, requiring only space proportional to the size of the tree and associated sufficient statistics.

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

The generated decision tree is asymptotically nearly identical to that of the standard batch algorithm, given enough examples. Also, since every decision (e.g. expanding a node) in VFDT has statistical support, there is no need for pruning.

Finally, it is clearly incremental, since a model is available at any time after the first few examples are seen, and its quality increases smoothly over time.

2.2.3 K-Nearest Neighbors Online

The KNN algorithm is among the simplest of all machine learning algorithms. It works by searching for the K closest training examples in the feature space. A new example is therefore classified by a majority vote of its neighbors, that is, assigned to the class most common among its k nearest neighbors.

The best choice of K depends upon the data. Generally, larger values of K reduce the effect of noise on the classification, but make boundaries between classes less distinct. Also, if the goal is binary classification, it is helpful to choose an odd number as it avoids ties.

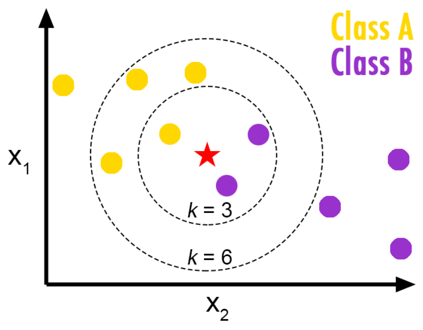


Figure 2.2: K-Nearest Neighbours

KNN is naturally incremental. Being a lazy learner, the training phase consists only of storing the feature vectors and class labels of the training samples. A commonly used distance metric for continuous variables is Euclidean distance, but other metrics can be used depending on the situation, like Hamming distance for discrete variables.

Although easy to implement, it has a few drawbacks, specifically, being computationally intensive for large training sets, as it calculates the distances from the test example to all the stored examples. However, there are ways of reducing the number of performed distance evaluations, by using an appropriate nearest neighbor search algorithm. Its accuracy can also be severely degraded by the presence of noisy or irrelevant features.

Despite its downsides, the KNN is of great simplicity, effectiveness, intuitiveness and competitive classification performance in many domains.

There are many ways of approaching incremental online learning with a KNN classifier. An interesting example is that of Förster et al [FMC+10], who suggest the use of a "teacher", capable of labeling a prediction as correct or error. This way, when a new example is fed into the classifier, the example itself, the prediction and the teacher's signal are fed into the incremental online learning of the classifier, resulting in an adapted classifier.

2.2.4 Bagging Online

Bootstrap Aggregating, better known as Bagging, is an application of Ensemble Learning.

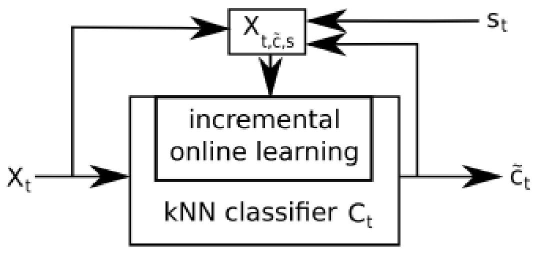


Figure 2.3: Teacher-based Incremental K-Nearest Neighbours

The algorithm works by training multiple classifiers, each on a random distribution (a portion) of the training set, and later combining their predictions through an averaging technique.

Bagging relies on a classical statistical sampling technique called Bootstrapping, which creates a random new subset of data by sampling, usually with replacement, from a given dataset. This means that, from one original dataset, many similar datasets can be generated.

The algorithm consists of creating K learners, each created independently, and bootstrapping the original training set, from which each learner generates its own training dataset. The learning is performed in parallel. To test, we predict on all K of the predictors, and have them do an unweighted combination, for example, majority voting.

Since each classifier is trained with its own individually generated training set, sampled from the original one, no classifier is able to fully memorize the original data, which significantly reduces overfitting.

Bagging tries to work on the bias-variance tradeoff, mainly on reducing variance. To elaborate, bias is the inability to represent the true function in the class of functions that we are willing to tolerate (e.g. only looking at certain classes), whilst variance happens when the model is extremely dependent on the exact data they were trained on (e.g. small training set). These two events are the main sources of error that prevent learning algorithms from generalizing beyond their training set.

A simple example to further understand bias and variance is to imagine a model for the percentage of people who will vote for President X. A way of building the model would be by randomly calling 50 phone numbers from the phone book, and ask each responder who they plan to vote in.

There are two aspects that can go wrong with this approach. First, the fact that the training sample only includes people from the phone book means that it only contains people with listed numbers, leaving out all the others, who might contain different voting patterns. This "discrimination" of classes is a source of bias. By only surveying certain classes of people, it skews the results in a way that will be consistent if we repeated the entire model building exercise, but would not result in an increased scatter of estimates.

On the other hand, the small sample size is a source of variance. If we increased the sample size, the results would be more consistent each time we repeated the survey and prediction. On

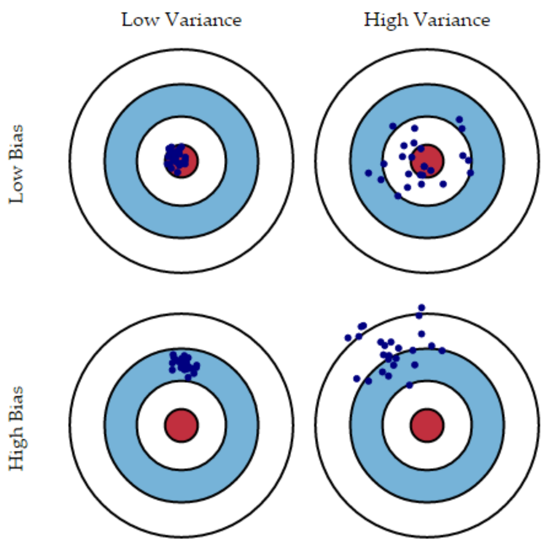


Figure 2.4: Bias-variance tradeoff

Figure 2.4, we can see that high bias results in a deviation from the target estimates, whilst high variance results in a scatter of the estimates.

Bagging works on the high variance situation, by taking a collection of low bias-high variance models and averaging them together. This reduces the complexity of a model class prone to overfit, because although some predictors might overfit on their training dataset, their averaging produces a model that is robust to small variations in the data.

Even though bagging results in more computation per prediction, it has strong benefits. It is very simple to implement, and capable to significantly decrease variation without affecting bias, allowing for better generalization and therefore better classification results.

Nikunj C. Oza [OR01] presented an online version of Bagging. Given a training dataset of size N, each base model's bootstrapped training set contains K copies of the original dataset's examples. As N tends to infinity, the distribution of K tends to a Poisson distribution. Therefore, to perform bagging online, as each training example is presented to the system, for each base model we choose the example K times according to Poisson(1) and update the base model accordingly using an online base model learning algorithm. New examples are classified the same way in online and batch bagging (unweighted voting of the base models).

Online bagging is a good approximation to batch bagging as both produce similar models when trained with similar distributions of training examples.

2.3 Techniques

2.3.1 Ensemble Learning

The principle of Ensemble Learning is very simple. Instead of using a single learning algorithm, this method strategically generates and combines multiple models, in order to solve the same problem and obtain better predictive performance than could be obtained from any of the models individually.

This would allow to further train a specific model that makes good predictions with a particular piece of data, improving its performance, whilst reducing the probability of wasting such data on the possible degrading of an unfortunately poor model selection.

Despite simple, it is a very interesting technique, especially since the predictive models themselves are usually very small in terms of memory requirements, so it is of low practical exigency, and if it boosts the overall performance of the classification system, it may prove a good investment.

2.3.2 Hierarchical Classification

A classification problem is, in general, addressable as a flat classification. This means that each example is assigned to a single class, out of a finite set of classes.

However, some problems can benefit from a multi-step classification, that is, an initial classification into major classes, and then inside each class, further classify into sub-classes.

The hierarchical approach gives the opportunity to choose a classification algorithm for each node, which is a great benefit since by picking an algorithm that better suits a particular situation, we aim for the highest accuracy possible.

Assembling this classification procedure may also allow for better efficiency in terms of computational requirements, since if an instance is classified with a certain general class, specific features needed for further classification on other general classes can be ignored.

Hierarchical classification can be applied in the scope of this project in many ways, being a simple example a two-step classification, first in static or dynamic activities, and second between standing and sitting or between running and walking.

2.4 Semi-Supervised Learning

Semi-Supervised Learning, as already stated, are machine learning techniques which makes use of both labeled and unlabeled data for training. Since the acquisition of unlabeled data is relatively inexpensive, as opposed to labeled data which may even require a skilled human agent to perform the labeling, their conjunction can produce considerable improvement in learning accuracy. It is of great interest both in theory and in practice.

In this project, the goal is to create an online semi-supervised learning system, capable of generating a classification model from an initial labeled dataset, extracted from several different human subjects, and afterwards adapting to a single individual, by further training on its own unlabeled input stream, reducing the model's classification error for that specific user.

Although some classifiers were presented above, not every semi-supervised learning method uses classifiers as components. There are various approaches, each making strong model assumptions. One should use a method whose assumptions fit the problem's structure.

A few of the best known and most used semi-supervised learning methods will be now presented, as well as their assumptions, with the intent of grasping how beneficial they can prove themselves in attaining the project's goal. As always, methods which do not fit the project's foundation will be left out. This includes those incapable of handling unseen data, non-incremental, etc.

2.4.1 Self-Training

Self-training is a commonly used technique for semi-supervised learning. It consists of training a classifier with the available labeled data, and use the classifier to classify the unlabeled data, adding the most confident classifications to the training set. In short, the classifier uses its own predictions to teach itself.

This wrapper algorithm is, in general, hard to analyze, although some analyzers on convergence for specific base learners do exist. However, it is of appealing simplicity and has been put into practice on several actual projects, from natural language processing tasks to object detection systems.

2.4.2 Co-Training

Co-training is a variation of self-learning. It makes strong assumptions:

- Features can be split into two sets (two different, complementary views of the data).
- Each sub-feature set is sufficient to train a good classifier.
- The two sets are conditionally independent given the class.

Assuming that these aspects hold true, the algorithm consists of training two separate classifiers on the labeled data, each on the respective sub-feature set. Each classifier then classifies the unlabeled data, teaching the opposite classifier with the most confident predictions.

We need the assumption that sub-features are sufficiently good, so that we can trust the labels predicted by each learner. The sub-features need to be conditionally independent so that one classifier's high confident data points are independent and identically distributed samples for the other classifier.

Co-training has been applied in several real-world projects. It was for example used to classify web pages using the text on the page as one view and the anchor text of hyperlinks on other pages that point to the page as the other view. Whilst other algorithms had a hit-accuracy of 86%, co-trained models attained 96% accuracy.

This technique can only be beneficial if the sub-feature sets are independent, that is, one of the classifiers correctly labels a piece of data that the other classifier previously misclassified. If both classifiers agree on all the unlabeled data (no independency) labeling the data does not create new information.

Several variations of co-training have been purposed over the years. Nigam and Ghani [NG00] showed it is better to probabilistically label the entire U (unlabeled data) instead of a few most confident data points (co-EM).

Goldman and Zhou [GZ00] attempted to relax co-training's feature splitting assumptions by using two learners of different types, both trained on the whole feature set, and then use one learner's high confidence data points (identified with a set of statistical tests in U) to teach the other learner and vice versa.

They also later [ZG04] proposed a single-view multiple-learner Democratic Co-learning algorithm, an ensemble of learners with different inductive bias that are trained separately on the complete feature of the labeled data and then make predictions on the unlabeled data. If a majority of learners confidently agree on the class of an unlabeled point, that classification is added to the training data and all learners are re-trained on the updated training set. The final prediction is made with a variant of weighted majority vote among all the learners.

Zhou and Li [ZL05] propose tri-training, which uses three learners. If two of them agree on the classification of an unlabeled point, the classification is used to teach the third classifier. This method avoids measuring label confidence of any learner.

2.5 Evaluation

Despite the increasing number of online learning algorithms, the metrics and the design of experiments for assessing the quality of learning models is still an open issue. The main difficulties are:

- Continuous flow of data instead of fixed sample of independent and identically distributed examples.
- Decision models evolve over time instead of static models.
- Data is generated by dynamic environment non-stationary distributions instead of a stationary sample.

Evaluation is used in two contexts: Inside the learning system, to assess hypothesis, and as a wrapper over the learning system to estimate the applicability of a particular algorithm in a given problem. It is imperative to define what are the goals and the evaluation metrics of the learning task, so that we can design the experiments to estimate them.

For predictive learning tasks (classification and regression) the most relevant dimension is the generalization error, the estimation of the difference between the induced function and the goal function, and an estimate of the loss that can be expected when applying the model to future examples.

There are two main methods to achieve evaluation of stream learning models, Holdout and Prequential (A.K.A Predictive Sequential or Interleaved Test-Then-Train).

2.5.1 Holdout

In the Holdout method, an independent test set is chosen to form a validation testing set. At regular time intervals (or sets of examples), the current decision model is applied to the test set. The loss estimated in the holdout is the unbiased estimator.

Despite a very simple method, it is capable of giving a good estimate of how well is a model evolving.

2.5.2 Prequential

In the Prequential method, the error of a model is computed from the sequence of examples. Each individual example in the stream can be used to test the model before it is used for training, and from this the accuracy can be incrementally updated.

For each example in the stream, the actual model makes a prediction based only on the example attribute-values. The prequential error is computed based on an accumulated sum of a loss function between the prediction and observed values.

The results in the model being tested on examples it has not seen. It has the advantage that no holdout set is needed for testing, making maximum use of the available data. It also ensures a smooth plot of accuracy over time, as each individual example will become increasingly less significant to the overall average.

2.6 Tools

2.6.1 MOA

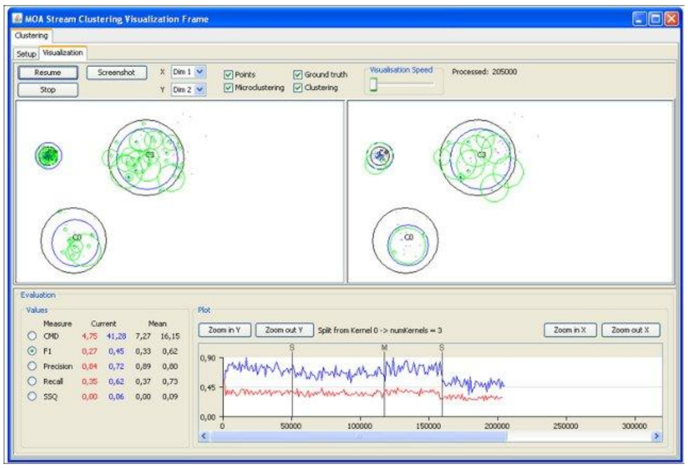
MOA is an open source framework for implementing and running algorithms for online learning from data streams. It contains a collection of offline and online machine learning algorithms for classification, clustering, regression, outlier detection, concept drift detection and recommender systems, as well as tools for evaluation.

The tool is designed to deal with the challenging problems of dealing with concept drift and scaling up the implementation of state of the art algorithms to real world dataset sizes, as well as making the algorithms comparable in benchmark streaming settings.

MOA, despite focusing on mining evolving data streams, supports bi-directional interaction with WEKA (Waikato Environment for Knowledge Analysis), an award-winning open-source workbench containing implementations of several batch machine learning methods.

It is written in Java, whose main benefits are its portability, the strong and well-developed support libraries, the language's widespread use, and features such as automatic garbage collection which help reducing programmer burden and error.

The tool's visualization component allows to visualize the stream, as well as the clustering results, choose dimensions for multi-dimensional settings, and compare experiments with different settings in parallel.



Measure	Current		Mean	
CMD	4,75	41,28	7,27	16,15
F1	0,27	0,45	0,33	0,62
Precision	0,84	0,72	0,89	0,80
Recall	0,35	0,62	0,37	0,73
SSQ	0,00	0,06	0,00	0,09

Figure 2.5: MOA's Graphical Interface

2.6.2 Renjin

Due to its mathematical power and simplicity, as well as graphical capabilities, the R software environment ended up being very used for data processing and visualization.

Because of that, the desire to be able to make use of its principles on a project which could be used in the android context emerged.

Renjin turned out to be a direct solution to the problem. Built on the JVM, it allows R code to interact directly with JVM libraries and data structures, without the need for inter-process communication.

Being an R implementation in Java, it can be seen as a simple java library possessing all the great principles and simplicity which turned R into such a famous language.

Despite a few instabilities, it was capable of serving our needs, and was used throughout the entire project.

2.6.3 Android

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets.

Android's source code is released by Google under open source licenses. This has encouraged a large community of developers and enthusiasts to build extremely rich and innovative applications, as well as to use the open-source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices originally shipped with other OS.

The Android SDK provides the tools and APIs necessary to develop applications on the Android platform using the Java programming language. It includes a debugger, libraries, emulators, documentation, sample code, and several tutorials. Developers have full access to the same framework API's used by the core applications.

Since both Android and MOA have Java as their foundation's programming language, it becomes easier to apply both together consistently.

2.7 HAR Key Studies

We now have a clear understanding of the main concepts behind machine learning applied to data streams, particularly in respect to online semi-supervised learning.

However, before we initiate the empirical work, it would be wise to further explore existing projects that strived to achieve the same end.

More than just a source of inspiration, this would allow us not only to acknowledge what worked properly, but also what did not, ultimately assisting us into formulating a good strategy to tackle the problem without following the same faulty footsteps.

Human Activity Recognition is not new. Several approaches have been attempted, from using external sensors, fixed in predetermined points of interest, to using cameras for video analysis. However, we will emphasize work that makes use of wearable sensors, in particular the modern day smartphone sensors. We will use Lara's survey on HAR [LL13] as a basis source.

eWatch [MRSS06] is an online HAR system which makes use of a sport watch device. The watch included four sensors, namely an accelerometer, a light sensor, a thermometer and a microphone. It was very energy efficient.

This approach used a C4.5 decision tree and time-domain feature extraction. The overall accuracy was up to 92.5% for six ambulation activities. The execution time for feature extraction and classification is less than 0.3 ms, making the system very responsive.

However, it achieved less than 70% for activities such as descending and ascending, and the data was collected under controlled conditions, with a lead experimenter supervising and giving specific guidelines to the subjects on how to perform the activities.

Vigilante is a mobile application for real-time human activity recognition under the Android platform. It makes use of both a smartphone and a chest sensor strap. It measured acceleration, heart rate, respiration rate, breath waveform amplitude and skin temperature, among others.

Statistical time and frequency-domain features were extracted from acceleration signals while polynomial regression was applied to physiological signals via least squares. The C4.5 decision tree classifier recognized three ambulation activities with an overall accuracy of 92.6%. The application can run for up to 12.5 continuous hours with a response time of no more than 8% of the window length.

Vigilante was tested completely online, to provide more realistic results. It was only moderately energy efficient because it required permanent Bluetooth communication between the chest strap and the smartphone.

ActiServ [BBG+10] was implemented on the Neo FreeRunner phone. It made use of a fuzzy inference system to classify ambulation and phone activities based on the signals given by the phone's accelerometer only, making it a very energy efficient and portable system.

The overall accuracy varies between 71% and 97%. However, in order to reach the top accuracy level, the system requires a runtime duration in the order of days. When the algorithms are executed to meet a real-time response time, the accuracy drops to 71%. Still, with personalization, in other words, subject-dependent analysis, ActiServ could reach up to 90% accuracy.

From the reported confusion matrices, the activities walking and cycling were often confused, while standing and sitting could not be differentiated when the cellphone's orientation was changed.

COSAR [RB11] is a framework for context-aware activity recognition using statistical and ontological reasoning under the Android platform.

The system was capable of recognizing ambulation activities, as well as brushing teeth, strolling and writing on a blackboard.

It gathers data from two accelerometers, one in the smartphone and another on the individual's wrist. It also used the smartphone's GPS, making it moderately energy efficient.

COSAR uses a concept of potential activity matrix to filter activities based on the user's location. For instance, if the user is in the kitchen, he is probably not cycling. Another contribution is the statistical classification of activities with a historical variant. For example, if the predictions for the last five time windows were jogging, jogging, walking, jogging, jogging, the third window was likely a misclassification.

The overall accuracy was roughly 93%. In some cases, standing still was confused with writing on a blackboard, as well as hiking up with hiking down.

Kao, Lin and Wang [KLW09] presented a system that made use of a triaxial accelerometer placed on the user's dominant wrist. It applied time domain features and the Linear Discriminant Analysis to reduce the dimension of the feature space. Then, a Fuzzy Basis Function learner, using fuzzy If-Then rules, classifies the activities.

The overall accuracy was of 94.71%, reached for seven activities: brushing teeth, hitting, knocking, working at a PC, running, walking and swinging.

The system reports an average response time of less than 10 ms, supporting its feasibility. However, all the computations are done in an embedded system that should be carried by the user as an additional device, a disadvantage in terms of portability, comfort and cost. Moreover, the size of the time window was chosen to be 160 ms, causing accidental movements when swinging or knocking to be confused with running, for instance.

Blum and Mitchell's system [MB98] made use of en-co-training. It was tested with ten ambulation activities and compared to three other fully supervised classifiers (KNN, Naïve Bayes and a Decision Tree).

The maximum error rate improvement reached by en-co-training was from 17% to 14% when 90% of the training data was unlabeled. If 20% or more of the training data was labeled, the error rate difference between en-co-training and the best fully supervised classifier didn't exceed 1.3%.

From this analysis, we can conclude that the existing online human activity recognition systems all have their own benefits as well as drawbacks. It is a challenge to address all the aspects of a good overall system. To make a system everybody can use we have to account for key points such as portability and obtrusiveness, energy consumption, overall system cost and processing needs, all aspects which are affected by how broad is the set of activities the system wishes to recognize.

However, this shouldn't be seen as discouraging, and rather as an opportunity for improvement. There are many strategies left to explore, as it is the one intended by this project. Only with unscathed persistency can the dilemma of Human Activity Recognition at the desired level and scale be attained.

Chapter 3

Methodologies

3.1 Environment

In an initial phase of the empirical work, careful consideration of the possible technologies to be used and experiences to be performed was not taken lightly, since a good working environment is essential to achieve great results and to maintain a productive work flow.

The main project was built using the Java language, since it is the dominant development tool for the android platform. The IDE of choice was Eclipse, for its well known simplicity, spread and fame within the java community.

It was decided very early on that MOA and Renjin were to be promptly included in the project. These powerful tools would make the core of the project. The MOA source code was cloned and imported into the workspace as a Maven project. The Renjin Engine was imported as a simple jar library.

With this, a very capable, yet manageable working environment was created, which allowed for the use of complex incremental machine learning algorithms with MOA, while Renjin facilitates all the necessary mathematics involved in the processing of the data stream.

Aside from the main project, R was also used as a visualization tool. Several R scripts that allowed for data plotting and analysis were written, which helped understanding the dataset and its perks.

3.2 Dataset

Without a doubt one of the most influential aspects about a successful machine learning project is a good dataset. Unfortunately, there is a limited collection of activity datasets with easy access and availability to the public.

The chosen dataset from within the possible solutions was created by Smartlab, and consists of experiments carried out by a group of 30 volunteers, within an age bracket of 19-48 years.

Each performed a protocol of activities composed of six basic activities: Standing, Sitting, Laying, Walking, Walking Downstairs, Walking Upstairs. The transitions between the static postures also count as activities: Stand-To-Sit, Sit-To-Stand, Sit-To-Lie, Lie-To-Sit, Stand-To-Lie, Lie-To-Stand. As such, there are a total of 12 activities.

The recording was performed at a constant rate of 50Hz by a Samsung Galaxy S II's accelerometer and gyroscope, which is very important since the project should be able to work with readings from a smartphone. Unfortunately, the phone was not located in the user's pockets, but on a waist belt, which compromised the desired unobtrusiveness. However, as a proof of concept that it is possible for an application with a generic model to improve itself from its user's data, this detail was not critical and was ignored.

Although the dataset comes with a processed version, only the unprocessed version composed of the original raw inertial signals from the smartphone sensors was utilized, since it provided more freedom of choice for custom features and data summarization.

Each person repeated the activity routine twice, totaling around 15 minutes of data per person. As such, the full dataset possess a few hours of activities, which ended up being less than the desired, but enough to achieve results.

3.3 Core Architecture

The project was split into several incremental experiments, each with specific goals. Some side experiments regarding visualization and data processing were also performed in parallel to the main experiments.

In the very first experiments, data was pre-processed into ARFF files, which is MOA's format of choice. Later, however, to make the experiments more realistic in terms of technical feasibility, data was processed in runtime as it would in a final application.

Although several classifiers were tested, the three main incremental algorithms used were Naive Bayes, Very Fast Decision Trees and K-Nearest Neighbors, especially as an ensemble.

Various Semi-Supervised Learning approaches were attempted around Self-Training and Co-Learning, which will be presented in more detail in Chapter 4.

Last but not least, the testing mechanism was a very important aspect of this project, because it was fundamental to understand the actual progress and accuracy being achieved throughout the entire work. As such, more than one testing mechanism was employed, depending on the context of the experiment, namely K-Fold Cross Validation, Prequential Evaluation, and even simple data splits into train and test, aiming at properly validating each section of the project.

Chapter 4

Empirical Research on HAR

4.1 Visualization

Visualizing what we are doing and what we want to do is of utmost importance to be able to fully understand it. As such, before any main experiment was performed, visualizing the acquired dataset was a top priority, especially to study which could be the most logical features to be later extracted from the data.

To achieve data visualization, R was used. The script allowed to analyze the accelerometer and gyroscope data of each activity interval, as plots of their triaxial readings. The results for each activity can be found in Appendix A.

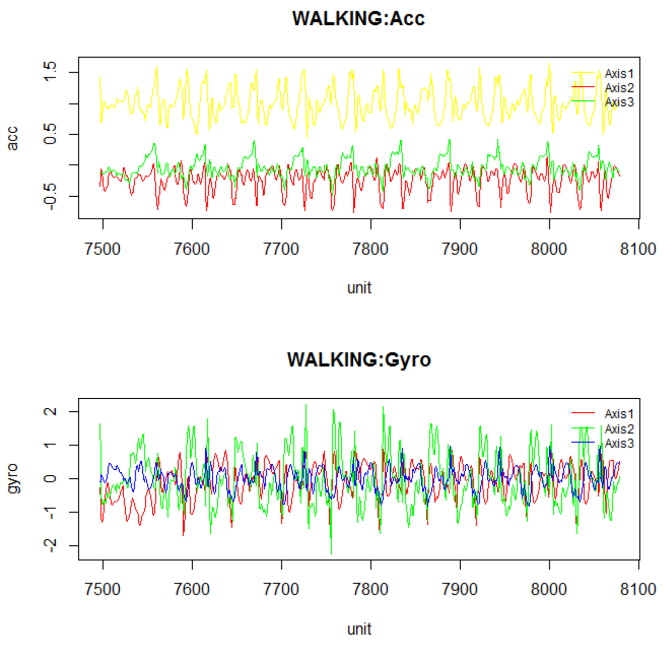


Figure 4.1: Plot of the sensor data which represents the activity "Walking". Each unit represents a single sensor reading, or in other words, a data point.

The analysis of the activity plots was very helpful to proceed with the summarization of the data. Since each activity seems to have its own mean values and well defined patterns, tests around the use of means, medians and deviations were performed. For instance, it was clear that the activity "walking", having several frequent spikes along the acceleration readings, would possess a high standard deviation value when compared to activities like "standing", with almost no standard deviation at all. These tests ended up working quite well, and will be presented with more detail in the first experiment.

These plots were used throughout the whole project, and proved to be very useful in explaining and solving many obstacles and doubts that occasionally surfaced related to the data's constitution.

4.2 Experiments

The experiments will be presented in the exact order they were performed. This order was important to tackle the final problem incrementally, adding one layer of complexity to the solution at a time.

4.2.1 Perfect Segmentation Supervised Learning

4.2.1.1 Goals

Perfect Segmentation is what we call knowing exactly when an activity starts and where does it end. Perfect Segmentation is virtually impossible in a realistic context, because it is impossible to separate a stream of sensor data into their respective activities automatically. If that was possible, there would be no need for human agents to spend so much time labeling collected data.

However, the goal of this first experiment was not to be realistic, but to serve as a testbed for finding the features that better identify the activity data. Also, it was a good way of finding out the maximum accuracy that could be expected, since it was very unlikely that higher accuracy could be achieved in a realistic context.

4.2.1.2 Methodology

This very simple initial experiment consisted of splitting the dataset by a split factor (usually 50%), creating a train set and a test set, but taking into account that each set must have a fair amount of each activity, to avoid a training set to have no samples of a particular activity. Then, these sets would be pre-processed into ARFF files and fed to a classifier.

4.2.1.3 Results

Numerous combinations of features were experimented, mostly inspired on the already mentioned HAR Studies in Section 2.7.

However, the best result was a combination of 20 features, 10 from the accelerometer and 10 from the gyroscope:

Arithmetic Mean of the X, Y and Z axis of both sensors, individually and also together, resulting in 4 features for each of the sensors.

Standard Deviation of the X, Y and Z axis of both sensors, individually, resulting in 3 features for each of the sensors.

Pearson Correlation of axis X and Y, Y and Z, and X and Z, for both sensors, resulting in 3 features for each of the sensors.

With these features, we achieved an accuracy of 87.5% with Naive Bayes and Very Fast Decision Trees, and some very consistent 100% with KNN.

4.2.1.4 Conclusions

Although incremental algorithms are usually not as powerful as their batch implementations, this experiment proves that even with just Supervised Learning, it would be possible to achieve great results if the activities were well separated. Although it is not a realistic situation, these features have demonstrated to characterize the data quite well.

4.2.2 Perfect Segmentation Semi-Supervised Learning

4.2.2.1 Goals

After finding a good set of features, it was necessary to understand what to expect in terms of accuracy boost with the Semi-Supervised approach. Therefore, this second experiment served as an initial familiarization with this technique, in terms of both results and implementation. Perfect Segmentation was still used, since the experiment doesn't aim at realism, but at testing limits and data dynamics.

4.2.2.2 Methodology

The first implemented Semi-Supervised Learning algorithm was a combination of Democratic Co-Learning [ZG04] and Tri-training [ZL05]. As such, a new type of classifier was produced, an ensemble of three classifiers: Naive Bayes, VFDT and KNN.

After the initial supervised training, further training with an unseen instance was only performed if most classifiers agreed on the label they would classify the instance with. Since our ensemble had three classifiers, if two agreed, than that new instance would be used with the agreed label for training all the classifiers. If all disagreed, the instance would be discarded.

The testing was done by means of Leave-One-Out Cross-Validation, with the particularity that what is left out is not an instance, but a whole user. The reasoning behind this is due to the fact that an important goal of this entire project is to make the application adapt to a specific user. Therefore, the most logical validation for this experiment is to leave 1 user out of the 30, train a generic model in a supervised fashion with the remaining 29, test the model accuracy with the user left out, then train further with that user's data by means of Semi-Supervised Learning, to finally test the model again with the same data used for the first test.

Since the user's data can't be used for training if it has already been seen, either the user's data is split in two, for training and testing, or we use Prequential Evaluation. In this particular experiment, the data from the user was split, which might not have been optimal, but it was good enough to take the desired conclusions. However, later experiments made use of Prequential Evaluation.

4.2.2.3 Results

With this experiment, the Democratic Ensemble Classifier was capable of achieving a Cross-Validation accuracy average of 89.15% before the unlabeled data was presented, and 89.49% after the unlabeled data, that is, an increase of 0.34%.

4.2.2.4 Conclusions

Taking into account the fact that a single individual's data is only around 15 minutes long, and the data was split in half for train and test, the results are actually highly motivating.

This experiment proves that Semi-Supervised Learning can indeed be used to significantly improve a model's accuracy. As such, it is now time to start looking at the data as a stream.

4.2.3 Fixed-Length Sliding Window Supervised Learning

4.2.3.1 Goals

Now that we have a general idea of what to expect in terms of results, it was time to start working on realistic case scenarios. In this experiment, the data is handled as a stream. As such, a fixed-length sliding window was used to iterate the data in the order it was recorded.

4.2.3.2 Methodology

The window size was set to 200, due to not only showing up in several of the past HAR works, but because it indeed presented the most consistent results. Since the data was recorded at a frequency of 50Hz, this means that a window possesses 4 seconds of data.

Both non-overlapping and overlapping sliding windows were tested, with the overlapping having an overlap value of 70%, meaning that a window contains 70% of the data of the previous window.

Since now there is no Perfect Segmentation, a method of testing the accuracy of a model's classifications of a window had to be decided. Initially, the method used was that a window's label was defined by the activity most present. This meant that if a window has 90% "Walking" and 10% "Standing", the window should represent "Walking".

However, it turned out to be intolerant, since if an activity is just slightly less present in a window, for example 49%, it should still be a valid classification. As such, an updated method of validation stated that an activity should be a valid label if it appeared in a significant amount, such as at least 30%.

In the end, even this option was discarded, because in a realistic, practical context, if a window contains an activity at all, it should be a valid classification. If i just finished up "standing" and started "running", even if there is only 10% of "running" in the window, if the model classifies it as "running" it should be perfectly valid, and therefore this was the final decided method of testing classifications.

Other than the already employed classifiers, a variant of the previous ensemble classifier was created: Confidence Ensemble Classifier.

In MOA, every classifier has the *getVotesForInstance* method, which returns a list with all the votes the classifier assigned to each activity. As such, for each classifier in the ensemble, the votes were collected, equally scaled, and added. In the end, the final classification is, simply, the most voted activity.

This differs from the Democratic Ensemble Classifier because if a classifier is 99% certain of an activity label, but the other two agree on a different activity with just 30% certainty, then it is not absurd to infer that the first classifier's opinion should be taken into account, despite its numeric disadvantage. Also, this classifier always presents a classification, as opposite to the Democratic, which does not provide a classification when all the classifiers disagree.

4.2.3.3 Results

Several cases were tested in this experiment. Both ensembles and their individual classifiers were tested in Non-overlapping and Overlapping Windows. The results are presented in the table below.

%	Non-Overlapping	Overlapping
Naive Bayes	81.22	82.11
VFDT	81.22	83.33
KNN	82.64	82.51
Democratic	81.22	84.33
Confidence	82.05	85.41

4.2.3.4 Conclusions

From the analysis of the obtained results, we can conclude that Overlapping Windows are significant and consistently better than Non-Overlapping Windows.

Another interesting observation is how the ensemble classifiers are being able to provide very competitive results in contrast to the individual classifiers. The best result was indeed obtained from the new Confidence Ensemble Classifier, with an accuracy of 85.41%. As this is now a realistic context, it is a satisfying result, and a great basis for the Semi-Supervised Learning approach.

4.2.4 Dynamic Data Segmentation

4.2.4.1 Goals

This experiment deviated from the path that the previous experiments were taking. In this case, an attempt at discarding fixed-length windows was performed by implementing the Dynamic Data Segmentation algorithm [KLG11]. Since 100% accuracy was achieved when the activities were perfectly segmented, trying to more accurately separate the stream of data was a logical and worthy effort.

4.2.4.2 Methodology

The implementation of the algorithm was initially done in R, not only due to its simplicity but also for visualization. Later, it was implemented in Java, reducing the run time of the experiment from several minutes to a few seconds.

Although the constant C of the algorithm was calculated with the purposed method, since it was not giving very good results, the program tested several combinations of N (number of previous data points used to calculate the threshold) and C, in order to find the best possible values.

4.2.4.3 Results

Unfortunately, the results were far worse than expected. Using the same features than up until now, with the Confidence Ensemble Classifier, an accuracy of only 67.2% was achieved.

In an attempt of optimizing the results, we experimented with the same features used in the article. Although the results were better, the maximum accuracy achieved was of 75.1%, which was much lower than desired.

4.2.4.4 Conclusions

There are many reasons why this approach might not have worked as intended. The resulting windows presented many inconsistencies.

For instance, as we can see on Fig 4.2, an activity like "Standing" could be split into intervals ranging 8 data samples up to 50.

The summarization of these windows would therefore result in unpatterned metrics, which are prone to confusing the learning algorithms.

Also, the very objective of implementing this technique would be to separate two consecutive activities. However, since the algorithm is based on finding sudden descending pikes of acceleration, many situations in which a segment possesses two activities in almost the same quantities happen, simply because the activities do not contain that event in between. Fig 4.3 is an example of a window with almost 50% of both Lie_To_Sit and Sitting, and where the change between one and the other is smooth enough not to be segmented into two windows by the algorithm.

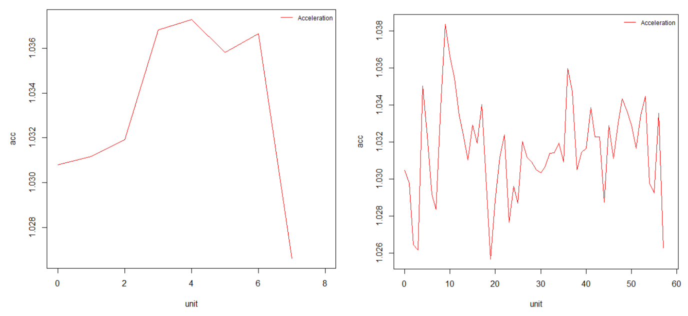


Figure 4.2: Two segments of the activity "Standing" with very distinct sizes.

It is also hard to define what is the ideal scenario about this kind of dynamic-length sliding window algorithm. If the segmentation was to only happened after an activity ended and another started, in a real case scenario, this would mean we would only be able to process the window and know what activity we've been doing once we actually finished in it and moved onto the next one. This is an unacceptable user experience. If, however, the goal was to segment each activity in every acceleration drop, we would always acquire windows such as Fig 4.2's left image, which would likely result in the windows being too small to provide quality features.

Still, this is mostly speculation, and we actually believe that Dynamic Data Segmentation might have a strong role in solving the HAR challenge. The delayed response could always be fixed by presenting an estimate of the activity being done after the window has a minimum size. In the end, user experience can always be tweaked into feeling right, so it is always worth to further explore this promising approach. Still, since it did not provide satisfying results in our experiments, we embraced Overlapping Fixed-Length Sliding Windows for the remaining of the project.

4.2.5 Fixed-Length Sliding Window Semi-Supervised Learning

4.2.5.1 Goals

After all the previous checkpoint experiments, we are now finally ready to tackle head-on the concept of applying Semi-Supervised Learning to improve a generic model's accuracy.

As such, the goal of this experiment was, fundamentally, the goal of the entire project: To understand if it is possible to use unlabeled data, acquired from the application's final user, to improve the generic model which composes the initial state of the application.

4.2.5.2 Methodology

As in the previous experiments, a sliding window of size 200 with 70% overlapping was used. Testing was performed with Leave-One-Out Cross-Validation, with Prequential Evaluation, meaning that each window of data from the user that was left out would first be used for testing (the

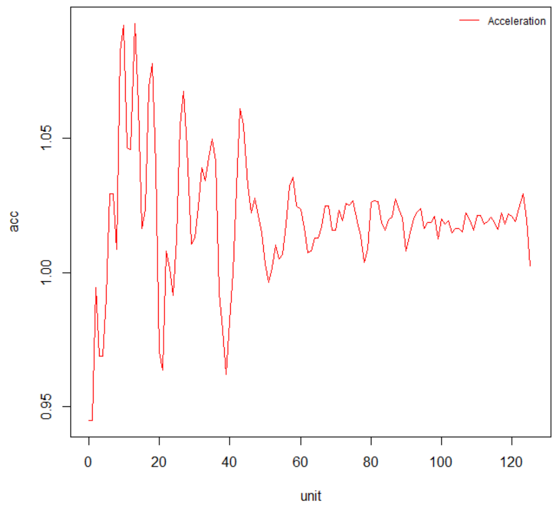


Figure 4.3: Segment containing both the activity "Lie_To_Sit" and "Sitting" in similar quantities.

model would try to classify the window), and only then for training, if the window was considered a good training sample. This technique allowed to use the data to its full potential.

This was by far the most technically exhausting experiment, as it was troublesome to achieve positive results. We had to start this experiment fully aware that it was a possibility this approach could simply fail at providing better results. One of the biggest obstacles that were faced was that several times, a window that was considered a good training sample was labeled incorrectly, which meant the model was being wrongfully trained. Contrary to the previous Semi-Supervised Learning experiment with Perfect Segmentation, positive results were not achieved immediately.

After multiple experiments resulting in a worse accuracy after the use of the unlabeled data, positive results were attained by resorting to a very specific condition for a sample to be considered train material: An extremely high degree of certainty.

As such, a new instance was only used for further training if every classifier composing the ensemble had at least 99.9% of their votes in the same activity. It might seem too restrictive that a minor disagreement is enough for an instance to be discarded, but in the end, it was concluded that it is more desirable for an instance to be discarded than to be wrongfully used.

This method of determining if an instance should be used for further training or not allows us to also use the classifiers composing our ensembles independently, since each classifier is capable of voting. Therefore, although each classifier and ensemble has their own ways of classifying an instance, in this experiment they all used the same method to determine trainable instances.

4.2.5.3 Results

The following table shows the results of every test performed in this experiment. For each classifier or ensemble, the model accuracy was tested before and after the unlabeled data was applied by the

Semi-Supervised methods. The table also contains a column with the difference between the pre and pos accuracies, for easier interpretation.

	Average		
%	SL	SSL	Diff
Naive Bayes	82.58	82.76	0.18
VFDT	83.56	83.47	-0.09
KNN	76.29	72.72	-3.57
Democratic	82.89	83.03	0.14
Confidence	84.55	84.63	0.08

As can be seen from the result table analysis, the ensembles remain the classifiers with the most consistent results. The Democratic Ensemble was capable of achieving an average improvement of 0.14%, and despite the Confidence Ensemble providing a lower average improvement (0.08%), it's final accuracy is the highest (84.63%).

Naive Bayes also behaved surprisingly well, with an average accuracy gain of 0.18% for a final accuracy of 82.76%.

VFDT and KNN did not do so well as independent classifiers. In average, the SSL approach reduced the model's performance. However, when working as an ensemble, their view of the data was beneficial at achieving consistently positive results.

4.2.5.4 Conclusions

This experiment was very gratifying as it proves that Semi-Supervised Learning can be used in practice to improve a model's accuracy, achieving a better HAR system.

The small amount of data per person (15 minutes) is a strong obstacle that might have forbidden higher increases in performance of being achieved. However, it is indeed an undeniable fact that increases were achieved, with nothing but some unlabeled data.

It is very fascinating to think that by just processing a stream of unlabeled data extracted from a smartphone's sensors, a generic model can improve itself, and classifications that were once incorrect become accurate.

4.2.6 SSL Using My Own Data

4.2.6.1 Goals

With the desire to take one step further into proving that Semi-Supervised Learning has a realistic place in achieving accurate and autonomous HAR systems, the author himself decided to record some of his own data in similar conditions to that of the original dataset.

The goal of this experiment was to prove that a generic model, fully trained in a dataset built in lab conditions, can improve its performance and adapt to anyone, even to the author of this thesis.

4.2.6.2 Methodology

With the help of a waist belt, a Smartphone Galaxy S3 and a stopwatch (for ease of labeling in order to find the accuracy improvements), the author recorded himself performing a routine containing the same activities used in this project:

Standing -> Sitting -> Laying -> Sitting -> Standing -> Laying -> Standing -> Walking -> Walking Upstairs -> Walking Downstairs

The routine was repeated three times, producing a total of roughly 50 minutes of data.

After going through a labeling process, the data was put to test. Classifiers were trained in a supervised fashion in the entirety of the original dataset, resulting in generic models. These models were tested in the user's data to acquire an initial accuracy value.

Then, the models employed the previous Semi-Supervised techniques to further train using the new, recorded data. Prequential Evaluation was used in order to acquire understandable metrics and make the most out of the author's data.

4.2.6.3 Results

Due to the fact the original dataset and the author's data were collected in unequal conditions, especially in terms of the waist belt used, some inconsistencies in the data patterns were produced. These inconsistencies tended to confuse most classifiers, and because of that, the threshold for an instance to be considered training material had to be increased from 99.9% to 99.99%. With this certainty, Naive Bayes was the classifier which better adapted to these inconsistencies, producing an accuracy increase from 60,17% to 60,25%.

100% certainty threshold was also tested for this experiment, with the Confidence Ensemble Classifier providing an accuracy boost from 62.18% to 62.22%.

However, in the end, all classifiers presented very low overall accuracies.

4.2.6.4 Conclusions

We can conclude from this experiment that even small variations in the data gathering conditions tend to affect the model's accuracy enormously. Most results from this experiment were adverse, due to the fact that a good generic model is essential as a base for the model to be capable of training itself. The role of Semi-Supervised Learning in these systems shouldn't be to turn a bad classifier into a good one, but to turn a good classifier into a better one.

Despite the results, Semi-Supervised Learning was still capable of improving some model's accuracy, especially in the case of Naive Bayes, which presented consistent improvements. It is very interesting to think that the 40 minutes of activities the author has performed were able to help a classification model improve its own accuracy.

This experiment was very important to understand both the role and the limitations of Semi-Supervised Learning. It also helped us conceive some of what we believe should be the next steps to be taken in this field, steps which will be purposed in depth in the next chapter.

Chapter 5

Conclusions and Future Work

5.1 Final Conclusions

The goal of this thesis was to tackle the field of Human Activity Recognition with a novel approach based on Semi-Supervised Learning, as a method of increasing a generic classification model's accuracy autonomously.

The empirical research was capable of demonstrating that SSL can indeed provide very consistent improvements to a generic model, adapting it to a specific user. Accuracy gains of up to 0.18% were achieved with just 15 minutes of unlabeled data.

Ensemble Learning was employed extensively throughout this project, and has provided with very positive and consistent results, constituting a powerful tool to solve not only HAR but many Machine Learning issues.

It was also concluded that SSL works the best when the base generic model has a decent initial accuracy, since a competent classifier is more qualified for self-training. Therefore, SSL excels at making good classifiers even better.

The fact that classifiers were capable of improving themselves through unlabeled data opens a future where applications will be capable of learning autonomously and grow more proficient at performing their determined task. With the development of novel applications capable of adapting to their specific user in order to better satisfy their needs, the way human beings interact with technology may very well evolve two steps further.

However, a lot of research is still essential to even think about turning this technology into an everyday tool. At the moment, very little examples of user-friendly applications that employ these techniques exist. Nevertheless, with the continuous efforts and advancements of the computer science community, this supposedly distant future is turning more and more into a reality each year.

5.2 Future Work

Throughout the course of this project, we felt that there were several aspects that should be addressed but that we could not cover as we had to keep a straight focus on our specific goals. However, in this section we will cover our view of what some future approaches to HAR should definitely consider.

The dataset that was chosen for this project was created by Smartlab and contained a total of 30 volunteers, each performing the same routine of activities twice, averaging 15 minutes per individual. Smartlab did a wonderful with their contribution to the computer science community with this dataset, but unfortunately it was in no way optimal for this specific project, and was chosen due to the lack of additional choices.

Its main inadequacy to the project was due to the fact that it focused on proving that SSL improved a generic model for a specific user. This means that while the generic model can be trained from the data of several individuals, the data used for testing and Semi-Supervised training must come from a single user. As such, it was especially ideal that we had recordings for each individual longer than 15 minutes. There is a great need for the creation of a larger dataset, available to the community, with much bigger time frames per person, such as days or even weeks. We realize that the costs of producing such a dataset are not to be taken lightly, but it would be a key step to produce more realistic and thoroughly tested applications, thus pushing HAR to the next level.

The dataset was also recorded using a waist belt, which ends up being very obtrusive. It should be stated that the author did not feel particularly comfortable recording its own data outdoors with a belt strapped around the waist, and the same feeling should be expected from every ordinary user. As such, it would be ideal that the new dataset had the smartphone located in the user's front pocket.

Another aspect of HAR that should be further researched is the fact that orientation matters. Very different data is produced depending on the smartphone's orientation inside the belt or a pocket, which will likely confuse the classification model and render the application useless. This is unacceptable in terms of user experience.

On the last experience, this issue was particularly felt, since a small variation in the belt and recording conditions was enough to lower the average classification accuracy from 80%'s to 60%'s, a huge performance drop that should not be allowed to happen.

As such, a method for converting sensor data from the smartphone's orientation to a generic orientation should be developed, so that as long as the smartphone is located in the pocket, the direction it is facing does not result in unusable data. The conference paper entitled "Correcting Smartphone Orientation for Accelerometer-Based Analysis" [TLB13] is an attempt at solving this issue, and it should serve as a basis for additional attempts, especially applied to HAR and Semi-Supervised Learning. We believe it is a key step in turning this technology into an everyday tool.

References

[BBG$^+$10] M. Berchtold, M. Budde, D. Gordon, H. R. Schmidtke, and M. Beigl. Activity recognition service for mobile phones. *In IEEE International Symposium on Wearable Computers (ISWC)*, 2010.

[BDM02] Brian Babcock, Mayur Datar, and Rajeev Motwani. Sampling from a moving window over streaming data. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 633–634, 2002.

[BI04] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1 – 17, 2004.

[DH00] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*, pages 71–80, 2000.

[FMC$^+$10] Kilian Förster, Samuel Monteleone, Alberto Calatroni, Daniel Roggen, and Gerhard Tröster. Incremental knn classifier exploiting correct-error teacher for activity recognition. *Proceedings - 9th International Conference on Machine Learning and Applications, ICMLA 2010*, pages 445–450, 2010.

[GZ00] Sally Goldman and Yan Zhou. Enhancing supervised learning with unlabeled data. *PhD Proposal*, 1, 2000.

[KLG11] S Kozina, M Lustrek, and M Gams. Dynamic signal segmentation for activity recognition. *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1–12, 2011.

[KLW09] Tzu Ping Kao, Che W. Lin, and Jeen Shing Wang. Development of a portable activity detector for daily activity recognition. *IEEE International Symposium on Industrial Electronics*, (ISlE):115–120, 2009.

[LL13] Oscar D. Lara and Miguel a. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013.

[MB98] Tom Mitchell and Avrim Blum. Combining labeled and unlabeled data with co-training. *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[MRSS06] Uwe Maurer, Anthony Rowe, Asim Smailagic, and Daniel P. Siewiorek. ewatch: A wearable sensor and notification platform. *Proceedings - BSN 2006: International Workshop on Wearable and Implantable Body Sensor Networks*, 2006:142–145, 2006.

[NG00] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. *Proceedings of the Ninth International Conference on Information and Knowledge Management - CIKM '00*, pages 86–93, 2000.

[OR01] Nikunj C. Oza and Stuart Russell. Online bagging and boosting. In *In Artificial Intelligence and Statistics 2001*, pages 105–112. Morgan Kaufmann, 2001.

[RB11] Daniele Riboni and Claudio Bettini. Cosar: Hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, 15(3):271–289, 2011.

[TLB13] Marco D. Tundo, Edward Lemaire, and Natalie Baddour. Correcting Smartphone orientation for accelerometer-based analysis. *MeMeA 2013 - IEEE International Symposium on Medical Measurements and Applications, Proceedings*, (May):58–62, 2013.

[Vit85] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

[ZG04] Y. Zhou and S. Goldman. Democratic co-learning. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 594–602. IEEE Comput. Soc, 2004.

[ZL05] Zhi Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541, 2005.

Appendix A

Graphics

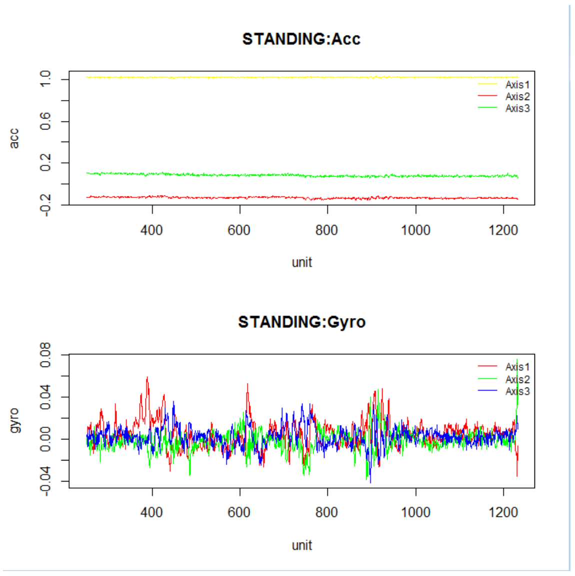


Figure A.1: Plot of the sensor data which represents the activity "Standing".

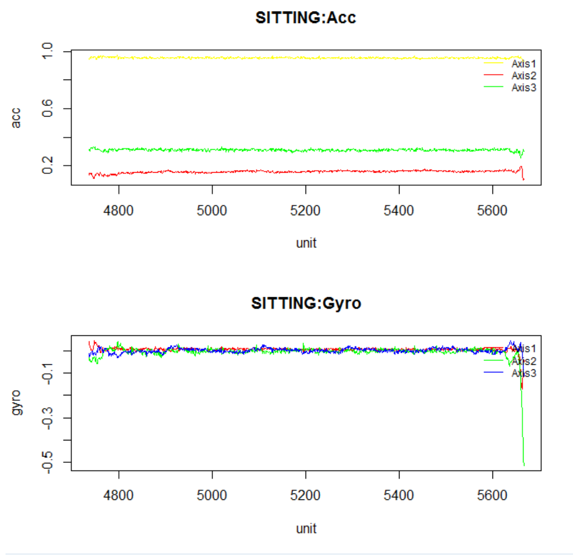


Figure A.2: Plot of the sensor data which represents the activity "Standing".

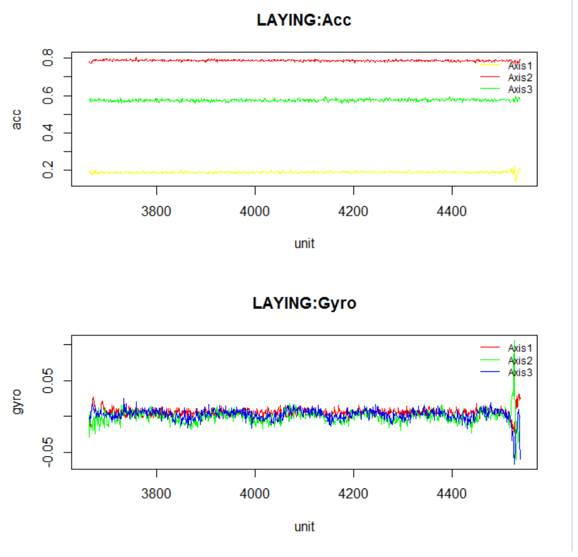


Figure A.3: Plot of the sensor data which represents the activity "Laying".

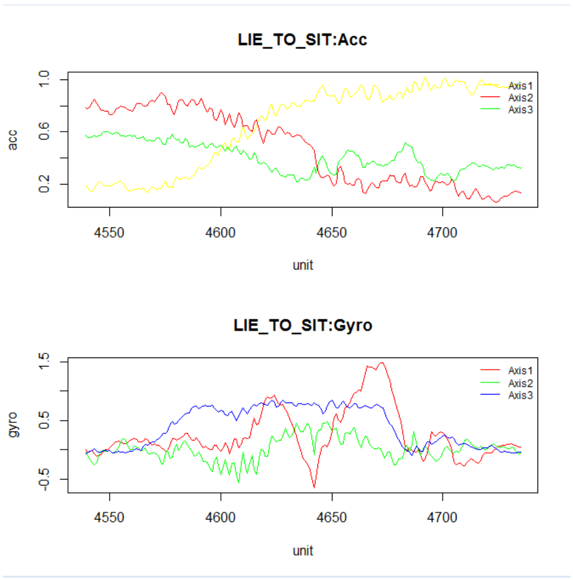


Figure A.4: Plot of the sensor data which represents the activity "Lie_To_Sit".

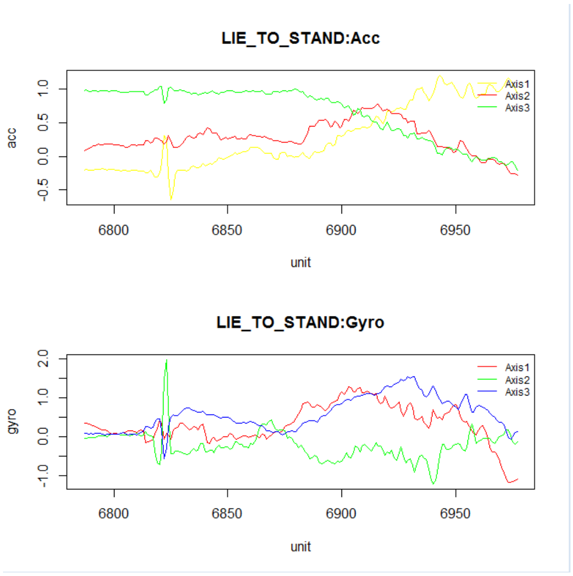


Figure A.5: Plot of the sensor data which represents the activity "Lie_To_Stand".

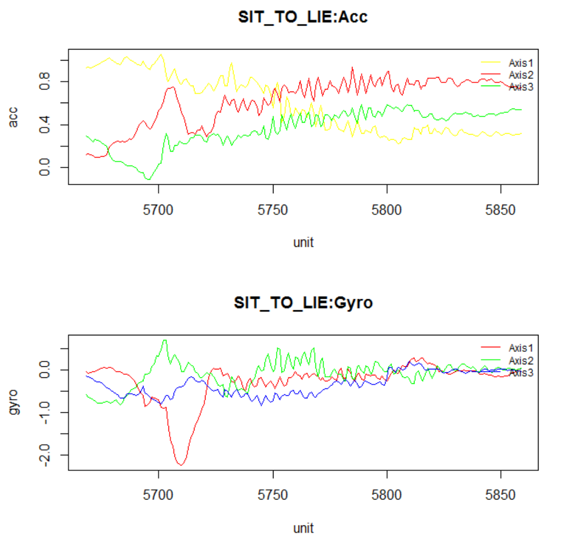


Figure A.6: Plot of the sensor data which represents the activity "Sit_To_Lie".

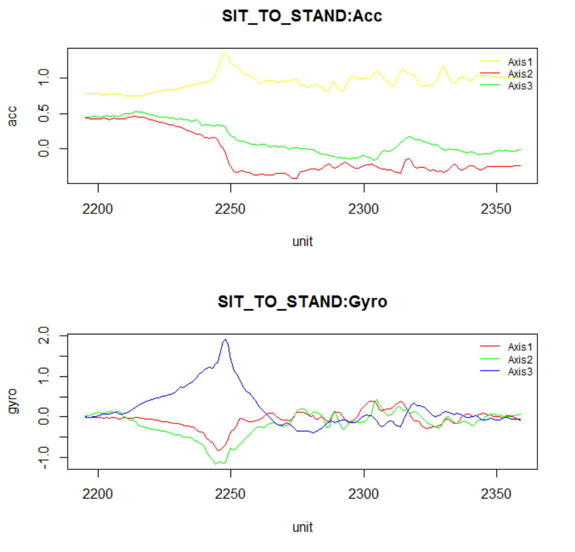


Figure A.7: Plot of the sensor data which represents the activity "Sit_To_Stand".

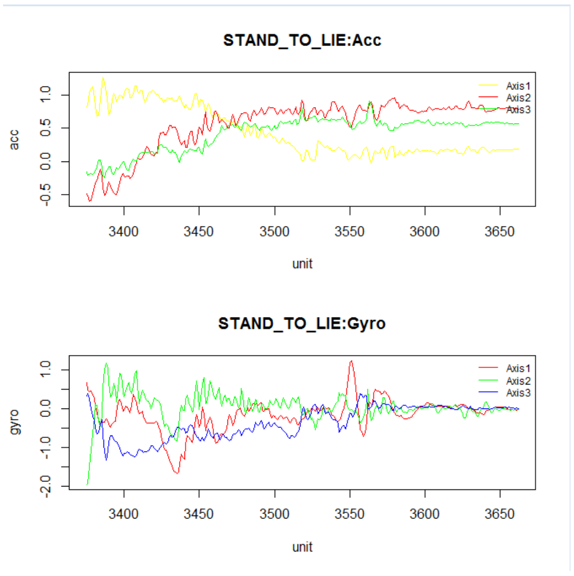


Figure A.8: Plot of the sensor data which represents the activity "Stand_To_Lie".

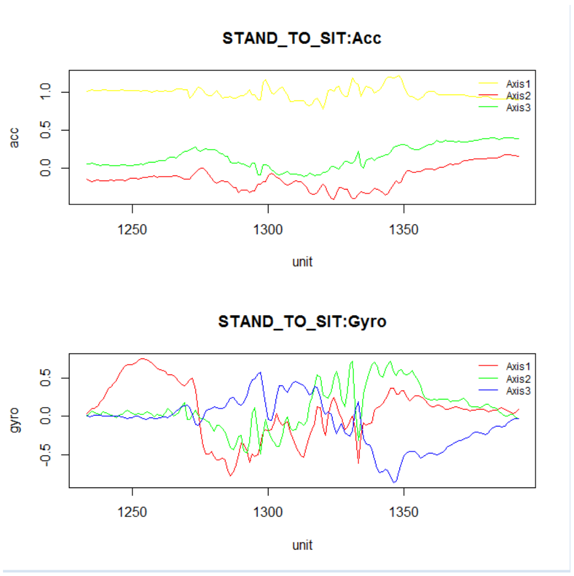


Figure A.9: Plot of the sensor data which represents the activity "Stand_To_Sit".

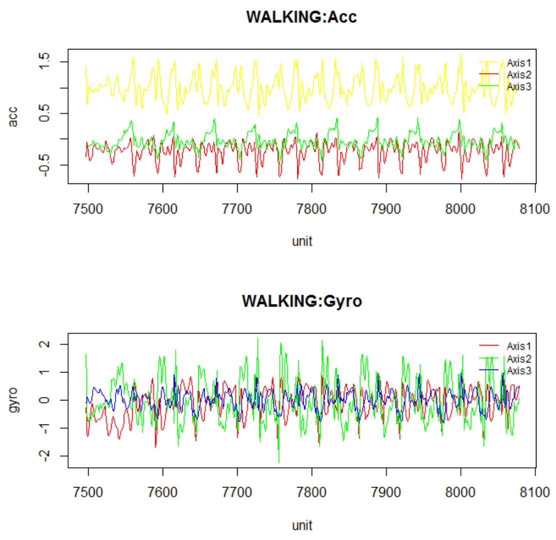


Figure A.10: Plot of the sensor data which represents the activity "Walking".

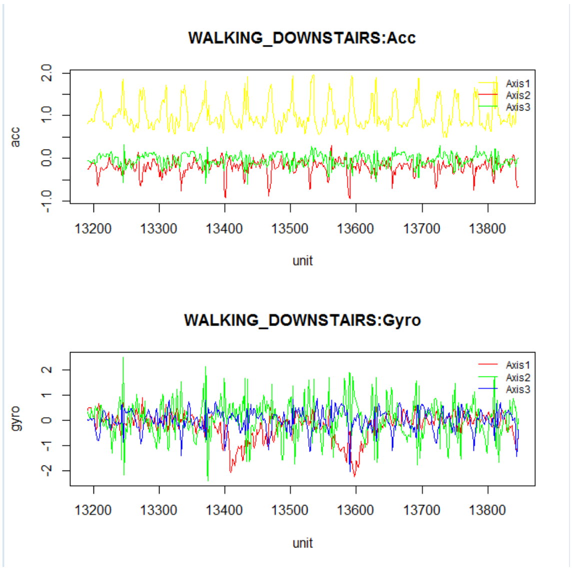


Figure A.11: Plot of the sensor data which represents the activity "Walking_Downstairs".

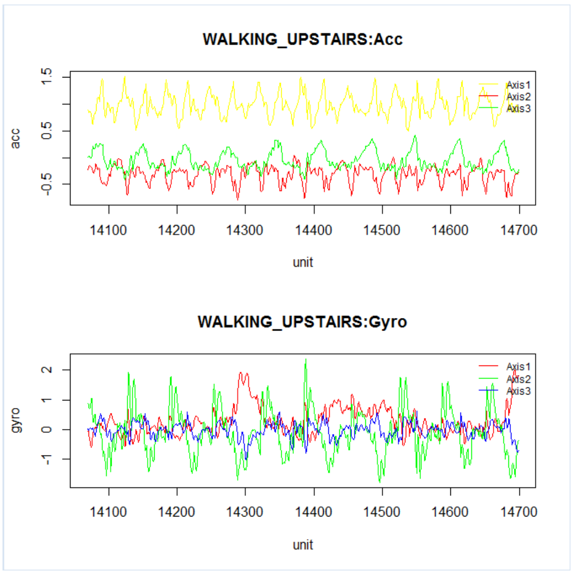


Figure A.12: Plot of the sensor data which represents the activity "Walking_Upstairs".