Sensors relevance analysis for transportation mode recognition

Supervisor:
Professor
Marco di Felice

Co-supervisor:
Dott. Luca Bedogni
Dott. Vincenzo Lomonaco

Candidate:
Claudia Carpineti

Session III
Academic Year 2015/2016

A Brunetto
che sarebbe voluto essere al mio fianco,
che avrei voluto al mio fianco.

Sommario

Identificare la mobilità di un utente attraverso la sua osservazione, o l'osservazione dell'ambiente, è un tema di ricerca di crescente interesse e con numerose applicazioni nel campo dell'*Internet of Things* (IoT). Capire come un soggetto si sta spostando dà la possibilità di offrire all'utente servizi appropriati alle sue necessità e possibilità di interazione.

La maggior parte dei lavori accademici trae vantaggio dalla diffusione dei cellulari, utilizzando i dati derivati dai sensori e le tecniche di apprendimento automatico, per inferire la mobilità dell'utente.

Una limitazione di questi lavori è l'utilizzo di dataset creati *ad hoc*. Essi, infatti, raccolgono i dati dei soli sensori di loro interesse, hanno spesso una base utente ridotta e in molti casi prevedono processi di collezione in condizioni non reali. Come conseguenza, i risultati dei lavori non sono fra loro confrontabili, rendendo di difficile individuazione le corrette strategie da mettere in atto per l'identificazione della mobilità in contesti reali.

Il primo obiettivo di questa tesi è stato la costruzione di un dataset che superasse le suddette limitazioni.

Il dataset creato è basato su tredici utenti che hanno raccolto i dati durante le loro normali attività giornaliere. Esso comprende tutti i sensori disponibili nel cellulare e distingue cinque attività: stare in macchina, in autobus, in treno, fermi e camminare.

Il dataset è stato utilizzato per la costruzione di modelli di classificazione della mobilità. Questi modelli sono il risultato dell'utilizzo di quattro algoritmi noti in letteratura (Decision Tree, Random Forest, Support Vector Machine e Neural Network) e hanno raggiunto un livello massimo di accuratezza del 96%.

Si è inoltre indagato sull'importanza dei sensori nel riconoscimento delle singole attività. A tal fine, sono stati definiti tre insiemi di dati: il primo composto dai dati derivati da tre soli sensori (accelerometro, giroscopio e microfono), il secondo composto da tutti i dati dei sensori ad esclusione di quelli derivati dal GPS (velocità) e l'ultimo dai dati di tutti i sensori disponibili.

I risultati ottenuti mostrano come all'aumentare dei sensori, aumenti l'accuratezza del modello. Non tutte le classi di attività, però, traggono lo stesso beneficio dall'aumento di informazione. Ad esempio, è stato notato che per il riconoscimento dell'attività di camminare, il modello, a prescindere di quali dati dispone, utilizza sempre accelerometro e giroscopio e aggiungendo altri sensori la sua accuratezza aumenta di poco. L'analisi di queste differenze permette di individuare quali sensori sono più utili all'individuazione di ogni singola attività.

Questi ultimi risultati suggeriscono la possibilità di scegliere il set di sensori da utilizzare sulla base delle attività da riconoscere.

Abstract

Identify user's transportation modes through observations of the user, or observation of the environment, is a growing topic of research, with many applications in the field of *Internet of Things* (IoT). Transportation mode recognition can provide context information useful to offer appropriate services based on user's needs and possibilities of interaction.

Works on transportation mode recognition take advantage of the general use of mobile, collecting data from phone's sensors and applying machine learning techniques to infer the user's transportation mode.

A strong limitation of these works is the use of *ad hoc* datasets. They collect data only from few sensors, often they have a lack in user base and in many cases the collection processes use non-real conditions. As a consequence, the results are not comparable, making difficult to choose the best strategies in real contexts.

The first aim of this thesis was to build a dataset to overcome the above limitations. Our dataset is based on the thirteen users who collected the data during their daily activities. The dataset includes all sensors available in phones and distinguishes five transportation modes: being on a car, on a bus, on a train, standing still and walking.

The dataset was used to build different models, whit different classification algorithms (Decision Tree, Random Forest, Support Vector Machine e Neural Network) reaching a maximum level of accuracy of 96%.

In order to investigate the relevance of each sensor in classification, we have defined three sets of sensors. The first set is composed of three sensors (accelerometer, gyroscope, and microphone); the second consists of all monitored sensors apart from the speed, for battery reasons.

The results show that the extension of the dataset with data derived from several sensors improves the ability of the model inference. However, accuracy gain with more sensors varies significantly for different transportation modes.

For example, to distinguish walking activity, models always uses accelerometer and gyroscope even if there are all sensors available. Analysis of these differences allows to understand which sensors are more useful to detect each activity.

These latest results suggest to choose the set of sensors to be used for recognition based on the desired transportation modes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic recognition of physical activities, commonly referred to as Human Activity Recognition (HAR), aims to identify the actions carried out by a person given a set of observations of the person itself and the surrounding environment. HAR is an emerging field of research, born from the larger fields of context-aware computing.

User transportation mode recognition can be considered as a HAR task; its goal is to identify which kind of transportation - walking, cycling, driving etc . . . - a person is using.

Transportation mode recognition can provide context information to enhance applications and provide a better user experience, it can be crucial for many different applications.

- *Device profiling* : activity information can be used to automatically customize the behavior of mobile devices. For instance, a mobile phone could be redirecting calls to voicemail if the user is jogging or switch on the ringtone while driving [17].

- *Monitoring road and traffic condition* : know how a user is moving permits to build a mechanism that allows people to know, in real-time, the traffic conditions on the routes they wish to travel and detect critical situations for the urban mobility, such as crushes and congestions [20],[19].

- *Healthcare* : detecting the transportation mode can capture the level of physical activity for users and relate this information to personal health. For medical patients, this information can enhance the quality of continuous monitoring applications. For healthy people, this information can be used to maintain their fitness generating a daily/weekly activity profile to determine if they are performing the recommended amount of exercise.

- *Traveling support* : transportation mode information can be used to give smart route recommendation based on the person's needs. For example, a navigator system can plot, not only a line from two points but can also infer what kind of actions the user has to take to reach the destination (e.g. parking) and can give support [21].

Advanced machine learning along with sensors and communication technologies is the most promising solution for user transportation mode recognition. The proliferation of smart phones that have an abundance of functionality together with sensing capability made it possible to use them for human activity recognition purposes.

Although many studies have investigated the transportation mode detection, we believe there is still space for improvement under different points of view.

This thesis focuses on the recognizing of five transportation modes, through models trained with sensors data collection. We consider daily activities as walking, being on a car or on a train or on a bus or standing still. In order to have a realistic model, we build a new dataset from thirteen volunteer subjects that annotate through a mobile application the transportation mode they are using during the day.

Our main purpose in this thesis is the study of how accuracy results obtained from dataset collected under real world conditions are different from those present in literatures. Moreover, we want to understand which sensors are important to detect specific activities, with the aim of eventually suggest to choose the best set of sensors for a specific target class.

1.1 Outline

The rest of this thesis is organised as follows.

Chapter 1 : gives an overview of context-aware computing, mobile sensor usually used to collect data and machine learning techniques used to build awareness through collected data.

Chapter 2 : takes a look on different research in transportation mode recognition. gives an overview of different approaches for the tranportation mode recognition problem. Finally, limitations of this approaches are shown.

Chapter 3 : describes the processes used to collect sensor data, pre-processing the result raw data and extract features.

Chapter 4 : includes results obtained with different approch on our dataset.

Chapter 5 : presents conclusions and discusses future work.

Chapter 2

Background

In this chapter, we give an overview of context-aware computing, mobile sensor usually used to collect data and machine learning techniques used to build awareness through collected data.

2.1 Context-awareness

Context awareness, as a core feature of ubiquitous and pervasive computing systems, has existed and been employed since the early 1990s. Many researchers have proposed definitions and explanations of different aspects of context-awareness and of context. Abowd at al. in [1], gives two important definitions:

Definition 2.1. ***Context*** *: any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical or computational object.*

Definition 2.2. ***Context-awareness*** *: the use of context to provide task-relevant information and/or services to a user.*

The focus on context-aware computing evolved with *Internet of Things* (IoT) in the last decade. IoT [3] tends to create a word where all objects around us are connected to the internet and communicate with each other with minimum human intervention. The main strength of the IoT idea is the high impact it will have on several aspects of everyday life and behavior of potential users. From the point of view of a private user, the most obvious effects of the IoT introduction will be visible in both working and domestic fields. In this context, domotics, assisted living, e-health, enhanced learning are only a few examples of possible application scenarios in which the new paradigm will play a leading role. Similarly, from the perspective of business users, the most apparent consequences will be equally visible in fields such as automation and industrial manufacturing, logistics, business/process management, intelligent transportation of people and goods.

Perera et al. in [2] identify four phases of context-aware system life cycle, described below.

1. *Context acquisition* : context needs to be acquired from various source. Techniques used to acquire context can be classified based on different characteristics.

    - *The method used to collect data from sensors* : data can be asked from software component responsible fro acquiring data (pull) or submitted from a sensor (push).
    - *The frequency with which the data are collected* : data can be collected on a specific event or periodically.
    - *Source of the context information* : data can come from different sources, from sensors hardware, through a middleware infrastructure, or from context servers.
    - *Sensor types* : there are different types of sensors that can be employed to acquire context: *(1)* physical sensors, *(2)* virtual sensors, that retrieve data from many sources and publish it as a sensor data (e.g. calendar, contact number directory, etc . . .) *(3)* logical sensors, also called software sensors, combine physical sensors and virtual in order to produce more meaningful information.

- *Acquisition process* : there is three ways to acquire context: *(1)* sense: the data is sensed through sensors; *(2)* derive: the information is generated by performing computational operations on sensor data (calculate the distance between two sensors using GPS coordinates); *(3)* manually provide by the user.

2. *Context modeling* : the collected data needs to be modeled and represented according to a meaningful manner.

3. *Context reasoning* : modeled data needs to be processed to derive high-level context information from low-level raw sensor data. In literature it has been proposed the use of different context reasoning decision models, most of them originated and employed in the fields of artificial intelligence and machine learning (decision tree, neural network . . .).

4. *Context dissemination* : both high-level and low-level context information needs to be distributed to the consumers who are interested in context.

2.2 Sensors

A sensor measures different physical quantities and provides corresponding raw sensor readings which are a source of information about the user and their environment.

Due to advances in sensor technology, sensors are getting more powerful, cheaper and smaller in size. Almost all mobile phones currently include sensors that allow the capture of important context information. For this reason, one of the key sensors employed by context-aware applications is the mobile phone, that has become a central part of users lives. In the past few years, smartphones remarkably started to carry sensors like Global Position System (GPS) receiver, accelerometer, gyroscope, microphone, camera, etc . . .

The above trend, enabled by the proliferation of many sensors in the possession of common individual, creates an unprecedented potential for building services that leverage massive amounts data collected from willing users.

Above will discuss the characteristics of the most popular sensors which most smartphones have.

Accelerometer : the accelerometer sensor measures proper acceleration [1] forces (m/s^2) applied to the device. Multi-axis models of accelerometers are available on almost all phones, they return acceleration as a 3D-vector quantity. Note that the force of gravity is always influencing the measured acceleration. For this reason, when the device is sitting on a table (and obviously not accelerating), the accelerometer reads a magnitude of $g = 9.81m/s^2$. Similarly, when the device is in free-fall and therefore dangerously accelerating towards to ground at $9.81m/s^2$, its accelerometer reads a magnitude of $0m/s^2$.

Gyroscope : the gyroscope measures the rate of rotation in rad/s around a device's x, y, and z-axis. Gyroscope return 3D-vector includes entries whose values contain current angular velocity about the corresponding axes. Usually, is calibrated to give a reading of zero when the device is kept on a plane horizontal surface. Is used to automatically determine the orientation in which the user is holding the phone and use that information to automatically re-orient the display between a landscape and portrait view or correctly orient captured photos during viewing on the phone.

Magnetometer : the magnetometer sensor measures the strength and perhaps the direction of geomagnetic fields for all three physical axes (x, y, z) in μT(microtesla). The geomagnetic field is defined as the magnetic

[1]proper acceleration is the physical acceleration experienced by an object. Thus it is acceleration relative to a free-fall, or inertial, observer who is momentarily at rest relative to the object being misurate. Proper acceleration contrasts with coordinate acceleration, which is dependent on choice of coordinate systems and thus upon choice of observers.

force field that surrounds the earth. It is attributed to the combined effects of the planetary rotation and the movement of molten iron in the earth's core. The 3 axis magnetometer usually found in mobile phones, have 3 such components oriented such that each one is orthogonal to the other two, one each in the x, y and z directions in the device's local coordinate system.

The direction and magnitude of the earth's field change with location, latitude in particular. For example, the magnitude is lowest near the equator and highest near the poles. Some hard-iron interference, meaning the presence of permanent magnets (e.g. magnets in the speaker of a phone) in the vicinity of the sensor also affects the accuracy of the reading. The presence of electronic items, laptops, batteries, etc also contributes to the soft-iron interference. Flight Mode option in mobile phones might help in decreasing the electro magnetic interference. In addition to the above spatial variations of the geomagnetic field, time-based variations, like solar winds or magnetic storms, also distort the magnetosphere or external magnetic field of the earth.

Proximity : the proximity sensor measures the distance (in centimeter) from the sensor to the closest visible surface. The precise distance value reported by different devices can be different, due to differences in the detection method, sensor construction, etc. Moreover, some proximity sensors might be only able to provide just a boolean to indicate if there is an object which is near, more like presence detection, than an absolute value for the distance.

Since most proximity sensors detect electromagnetic radiation (e.g., an infrared light or a magnetic field), certain material properties can interfere with the sensor's ability to sense the presence of a physical object. Things that can interfere with a sensor include, but are not limited to, the material's translucency, reflectiveness, color, temperature, chemical composition, and even the angle at which the object is reflecting the radiation back at the sensor. As such, proximity sensors should not be relied on as a means to measure distance. The only thing that can be deduced from a proximity sensor is that an object is somewhere in the distance between the minimum sensing distance and the maximum sensing distance with some degree of certainty.

Ambient Light : the ambient light sensor provides information about ambient light levels, as detected by the device's main light detector. The output is current light level or illuminance, that is a value that represents the ambient light levels around the hosting device. Its unit is the lux (lx). Light level allows phones to automatically sets the screen brightness based on surrounding light and conserves battery life.

Microphone : microphone is a very common sensor; it is usually used for recording sound. This sensor can give very interesting information even when using minimal processing, such as noise level, type of input (noise, music, speaking), etc.

GPS : a GPS module connecting with satellite gives an accurate position result. A GPS receiver receives multiple signals from different satellites at different times, when it receives a transmission, based on the time it takes to get the packet it can determine how far it is from the satellite. Once the GPS module collects enough satellite data to calculate an accurate position, it has a valid location (a fix) that it can report. GPS works well once phone finds three or four satellites, but that may take a long time, or not happen at all if is indoors or in an "urban canyon" of buildings that reflect satellite signals. GPS return coordinates (usually expressed as longitude/latitude), the accuracy of returned coordinates, altitude, and the estimated speed.

2.3 Machine learning

Once the data from sensors are collect, machine learning techniques are used to model and derive context information from low-level raw data.

According to [4], machine learning is defined as:

Definition 2.3. *A computer program which learns a problem from experience E with respect to some class of tasks T and performance measure P. The performance at tasks in T, as measured by P, could improve with experience E.*

2.3.1 Supervised and unsupervised learning

Machine learning tasks are classified into two main categories: supervised and unsupervised learning. Supervised learning is the machine learning approach of inferring a function from supervised training data. The training data consisist of a set of of observations called *instances*, (x, y). An instance is a single observation of the world from which a model will be learned, or on which a model will be used (e.g., for prediction). From praticl point of view, an instance is a tuple (x, y) composed by a vector of *attributes* or *features* (x), which carachterize the data, and appropriate output (y). Each feature may be boolean, discrete with multiple values, or continuous. Learning in this context means finding the mapping function $f : x \rightarrow y$. The goal is to generalize from the training instances (o observations) that will enable novel objects (which we have never seen before) to be identified corresponding to appropriate output.

Unsupervised learning, instrad is closely related to pattern recognition. In unsepervised cases, the dataset does not include a known outcome. Often the goal in unsupervised learning is to decide which objects should be grouped together. The purpose of this learning technique is to find similarity among the groups or some intrinsic clusters within the data. The "correct" number of clusters depends on prior knowledge or biases associated with the dataset to determine the level of similarity required for the underlying problem. Theoretically, we can have as many clusters as data instances, although that would defeat the purpose of clustering. Unsupervised techniques can be used to identify the output for later use supervised classification algorithms.

2.3.2 Classification task

Learning how to classify objects to one of a pre-defined set of categories or classes is a basic characteristic of intelligence that has been of keen interest to researchers for many decades. The ability to perform classification and to be able to learn how to classify gives people and computer programs the power to make decisions. The efficacy of these decisions is affected by performance on the classification task.

In machine learning, the classification task is commonly solved through *supervised learning* techniques. For example, in medical diagnosis problems, the features x might include the age, weight, and blood pressure of a patient, and the class label y might indicate whether or not a physician determined that the patient was suffering from heart disease.

Notice that the classification task is to determine a class with the only information over the training examples. Therefore, inductive learning algorithms can at best guarantee that the output hypothesis fits the target concept over the training data.

The classification problems can be distinguished on the basis of the characteristics of the classes as we see below.

Binary classification : a classification task with two classes. An example of binary classification can be predict if email is spam or not spam.

Multiclass classification : a classification task with more than two classes. Multiclass classification makes the assumption that each sample is assigned to one and only one label. An example can be classify a set of images of fruits which may be oranges, apples, or pears.

Multilabel classification : a classification task where each sample can have a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant to a document.

The multiclass classification problem can be decomposed into several binary classification tasks that can be solved efficiently using binary classifiers [6]. There are many ways to reduce a multiclass problem to multiple binary classification problems. For all of these methods, after the binary classification problems have been solved, the resulting set of binary classifiers must then be combined in some way.

The simplest approach is *One-versus-all*, OVA approach ([7], [8]). OVA reduce the problem of classifying among K classes into K binary problems, where each problem discriminates a given class from the other $K-1$ classes. For this approach, we require $N = K$ binary classifiers, where the $k-th$ classifier is trained with positive examples belonging to class k and negative examples belonging to the other $K-1$ classes. When testing an unknown example, the classifier producing the maximum output is considered the winner, and this class label is assigned to that example.

Another scheme for multiclass classification is the *All-versus-all*, AVA scheme ([9], [10]). In this approach, $\binom{N}{2}$ binary classifiers are trained; each classifier separates a pair of classes.

2.3.3 Dataset

Supervised learning, is based on induction and user examples instead of general axioms as in deduction. In this scenario, users provide examples along with the associated classes. As we said in 2.3.2, dataset X consist of a collection of instances x (users examples) with a number of features f and an associated class c. A common procedure to train a model is random divide instances into the *training set* and *test set*. The general idea is this: use the training data to enable a classification rule to be set up, and use the test set as a second independent sample of new observations in order to get an unbiased performance evaluation on examples the classifier has neve seen before. The predicted and true classifications on the test data give an unbiased estimate of the error rate of the classifier.

Features selection

Type of training experience from which system will learn is the first design choice. The type of training experience available can have a significant impact on success or failure of the learner. If the data is inadequate or irrelevant then the concept descriptions will reflect this and misclassification will result when they are applied to new data. If, however, the data is suitable for machine learning, then the task of discovering regularities can be made easier and less time consuming by removing features of the data that are irrelevant or redundant with respect to the task to be learned. This process is called *feature selection*. The benefits of feature selection for learning can include a reduction in the amount of data needed to achieve learning, improved predictive accuracy, learned knowledge that is more compact and easily understood, and reduced execution time. There are a number of different definitions in the machine learning literature for what it means for features to be “relevant”. The reason for this variety is that it generally depends on the question: “relevant to what?”. More to the point, different definitions may be more appropriate depending on one’s goals. Here, we describe *incremental usefulness* and discuss its significance.

Definition 2.4. *Given a sample of data S, a learning algorithm L, ad a features set A, feature x_i is incrementally useful to L with respect to A if the accuracy of the hypothesis that L produces using the feature set $x_i \cup A$ is better than the accuracy achieved using just the feature set A.*

This notion is especially natural for feature-selection algorithms that search the space of feature subset by incrementally adding or removing features to their current set. There are a variety of natural extensions one can make to the above definition. For instance, one can consider a relevant *linear combination* of features, rather than just relevant individual features.

Data pre-processing

Collecting data is the first step towards preparing it for modeling, but it is sometimes necessary to run the data through a few pre-processing steps depending on the composition of the dataset.

Categorical features Many machine-learning algorithms work only on numerical data, integers, and real-valued numbers. The simplest machine learning datasets come in this format, but many include other types of features, such as categorical features [2], and some might include missing values or be in need of other kinds of data processing before being ready for modeling.

Categorical features are the most common type of non-numerical feature. Some machine learning algorithms deal with categorical features natively, but generally, they need data in numerical form. It is possible to encode categorical features as numbers (one number per category) but we cannot use this encoded data as a true categorical feature introducing an (arbitrary) order of categories. Recall that one of the properties of categorical features is that they are not ordered. What we can do instead is to convert each of the categories in the categorical feature to a feature with 1's or 0's wherever the category appeared or not. In essence, we convert the categorical feature into binary features that can be used by machine learning algorithms that only support numerical features.

Missing data Tabular datasets often contains data cells as *NaN*, *None* or similar. These represents missing data and are usually artifacts of the data collection process; for some reason, a particular value could not be measured for a data instance. There are two main types of missing data, which we

[2]Categorical feature is a feature that can take on one of a limited, and usually fixed, number of possible values. The values in a categorical variable exist on a nominal scale: they each represent a logically separate concept, cannot necessarily be meaningfully ordered, and cannot be otherwise manipulated as numbers could be. An example of categorical feature is the blood type of a person: A, B, AB or O.

need to handle in different ways. In one case the fact that the data is missing carries information in itself and could be useful for the algorithm. In another case the data is missing simply because the measurement was not taken and there is not information around the reason for the unavailability of the information. Considering the case of meaningful missing data. There is information in the data being missing, it would be useful that the algorithm is able to use this information to potentially improve the prediction accuracy. To transform missing data into information is usual to convert them into the same format as the column in general. For numerical columns this can be done by setting missing values to -1 or -999, depending on typical values of non-missing values. Simply pick a number in one end of the numerical spectrum that will denote missing values, and remember that order is important for numerical columns. For a categorical column with meaningful missing data, is possible to create a new category called "missing", "None" or similar and then handle the categorical feature in the usual way.

In the case of missing data where the lack of information carries no information, is not possible to simply introduce a special number or category because we might introduce data that are flat-out wrong. Some machine learning algorithms will be able to deal with these truly missing values by simply ignoring them. If the algorithm is not capable or processing missing values, a pre-processing phase is required. Such phase will replace the missing value with an "imputed" value. There are many ways to impute missing data, but there is, unfortunately, no one good solutions for all cases. The easiest and most undesirable way is to simply remove all instances for which there are missing values. This will not only decrease the predictive power of the model but also introduce biases in case the missing data are not randomly distributed. Another simple way is to assume some temporal order to the data instances and simply replace missing values with the column value of the preceding row. With no other information, we are making a guess that a measurement hasn't changed from one instance to the next. Needless to say, this assumption will often be wrong. It is usually better to use a larger portion of the existing data to guess the missing values. To avoid biasing the model is possible to replace missing column values by the mean value of the column. With no other information, make a guess that on average will be closest to the truth. The mean is sensitive to outliers, so depending on the distribution of column values we may want to use the median instead. These are widely used in machine learning today and work well in many cases.

2.3.4 Supervised learning algorithms

Decision trees

The decision tree is one of the more widely used classifiers in practice because the algorithm creates rules which are easy to understand and interpret. With this method, the learned function is represented by a decision tree. Learned trees can also be represented as sets of if-then rules.

Decision trees classify instances by sorting them down the tree from the root to leaf nodes, which provides the classification of the instance. Each node in the tree specifies a test of some feature of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root node of the tree, testing the feature specified by this node, then moving down the tree branch corresponding to the value of the feature in the given example. This process is then repeated for the subtree rooted at the new node. An example of a decision tree for concept *play tennis* is in Figure 2.1. Outlook, humidity, and the wind are features of the dataset, branches represents test on features, and leaf node represent class labels (in this cases *yes* or *no*).

Decision trees represent a disjunction of conjunctions of constraints on the feature values of instances. Each path from the root to a leaf corresponds to a conjunction of attribute tests, ant the tree itself to a disjunction of these conjunctions.

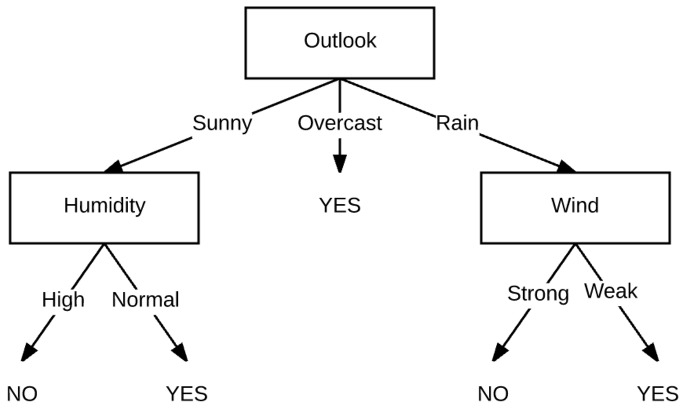


Figure 2.1: Example of decision tree for concept *play tennis*.

Random forest

Random forest is an ensemble learning method for classification, operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) of the individual trees.

Leo Breiman in [11] describes a method of building a forest of uncorrelated trees using a procedure that combines two different techniques. First technique **B**ootstrap **agg**regat**ing** ("bagging"), that is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification. It reduces variance and helps to avoid overfitting. Second technique is random selection of features, in order to construct a collection of decision trees with controlled variance. His proposal is the basis of the modern practice of random forest randomized node optimization and bagging.

Support Vector Machines

Support Vector Machines (SVM) [5] have long been recognized as being able to efficiently handle high-dimensional data. Originally designed as a two-class classifier, it can work with more classes by making multiple binary classifications (one-versus-one between every pair of classes).

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. The algorithm works by classifying instances based on a linear function of the features. Is possible to consider the algorithm outputs as optimal hyperplane which categorizes new examples. The operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. This distance is called *margin* in SVM's theory. The optimal separating hyperplane maximizes the margin of the training data, an example is in Figure 2.2 .

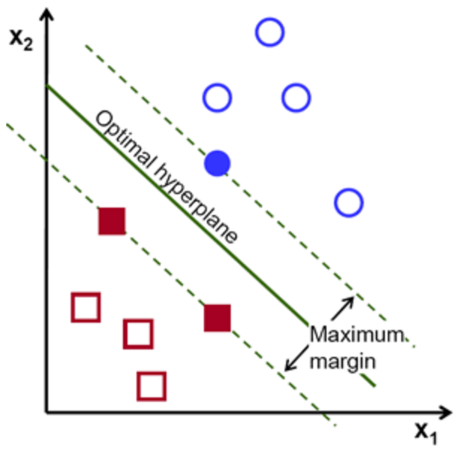


Figure 2.2: Example of SVM hyperplane. The optimal hyperplane, is the one that gives the largest minimum distance to the training examples

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using a kernel function to raise the dimensionality of the examples.

Artificial Neural Networks

An artificial neural network (ANN) is an interconnected group of nodes intended to represent the network of neurons in the brains. They are widely used in literature, because of their ability to learn complex patterns.

The study of ANNs has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons. In rough analogy, artificial neural networks are built out of a densely interconnected set of simple units, where each unit takes a number of real-valued inputs (possibly the output of other units) and produces a single real-valued output (which may become the input to many other units).

The artificial neural network is comprised of nodes (shown as circles in Figure 2.3), an input layer represented as $x_1, ..., x_n$, an optional hidden layers (one in Figure), and an output layer y. The objective of the ANN is to determine a set of weights w (between the input, hidden, and output nodes) that minimize the mean squared error (MSE) [3].

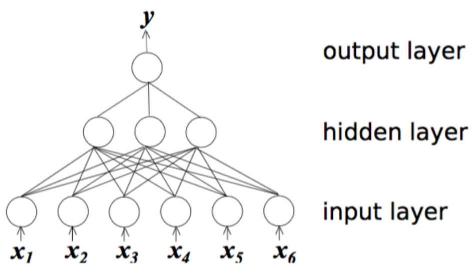


Figure 2.3: Example of multilayer artificial neural network.

The difficulty of using artificial neural networks is finding parameters that learn from training data without over fitting (i.e. memorizing the training data) and therefore perform poorly on unseen data. If there are too many hidden nodes, the system may overfit the current data, while if there are too few, it can prevent the system from properly fitting the input values.

2.3.5 Performance metrics

Empirically evaluating the accuracy of hypotheses is fundamental to machine learning. In testing the accuracy of a classification rule, it is widely

[3]Mean square error (MSE) measures the average of the squares of the errors, that is the difference between the estimator and what is estimated. The MSE is a measure of the quality of an estimator it is always non-negative, and values closer to zero are better.

known that error rates tend to be biased if they are estimated from the same set of data as that used to construct the rules. At one extreme, if a decision tree for example is allowed to grow without limit to the number of leaves in the tree, it is possible to classify the given data with 100% accuracy, in general at the expense of creating a very complex tree-structure. In practice complex structures do not always perform well when tested on unseen data, and this is one case of the general phenomenon of *overfitting* data.

When evaluating a learned hypothesis we are most often interested in estimating the accuracy with which it will classify future instances, so we use *test set* (built as described in 2.3.3).

Confusion matrix and accuracy

A *confusion matrix*, also called a contingency table, is a visualization of the performance of a supervised learning method. Multiclass classification with n classes requires a confusion matrix of size $n \times n$ with the rows representing the specific actual class and the columns representing the classifiers predicted class. In a confusion matrix, TP (true positive) is the number of positives correctly identified, TN (true negative) is the number of negatives correctly identified, FP (false positive) is the number of negatives incorrectly identified as positive, and FN (false negative) is the number of positives incorrectly identified as negatives. An example of a confusion matrix can be seen in Table 2.1.

		Predicted	
		+	-
Actual class	+	TP	NF
	-	FP	TN

Table 2.1: Example of confusion matrix.

From the confusion matrix, it is relatively simple to arrive at different measures for comparing models. An example is *accuracy*, which is a widely used metric and is easy to interpret. From equation 2.1, accuracy is the total number of correct predictions made over the total number of predictions made. While accuracy is a popular metric, it is also not very descriptive when used to measure the performance of a highly imbalanced dataset. A model may have high levels of accuracy, but may not obtain high levels of identification of the class that we are interested in predicting.

For example, if attempting to identify large moves in a stock which are comprised of 99% small moves and 1% large moves, it is trivial to report a model has an accuracy of 99% without additional information. A classifier could also have 99% accuracy by simply reporting the class with the largest number of instances (e.g. the majority class is small moves). In an imbalanced dataset, a model may misidentify all positive classes and still have high levels of accuracy; pure randomness is not taken into account with the accuracy metric. Accuracy's complement is the *error rate* $(1 - Accuracy)$ and can be seen in equation 2.2.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

$$ErrorRate = \frac{FP + FN}{TP + FP + TN + FN} \tag{2.2}$$

Chapter 3

State of the art

Machine learning methods are often applied for detecting transportation mode. With this approach, systems learn from a training dataset and use the gained knowledge to automatically deal with different datasets having similar characteristics. A lot of techniques in this category are used in transportation mode recognition. Some examples are: neural network, decision tree, support vector machine, random forest, etc... Studies differ mainly in the source of data that can be retrieved from various sources, such as environmental, wearable sensor or smartphone sensors.

A typical activity recognition system based on machine learning, follow these steps:

1. data is collected from various sensors and stored in a log of raw data;
2. meaningful features in a fixed size window are generated from raw data;
3. machine learning model is trained on features;
4. model is used to predict a user's activity for specific feature values.

Some existing works focus on using location information coupled with external data such as GSM cell tower and WiFi access point density [13], [14], [15]. In these cases accuracy is strongly strongly related with GSM and WiFi coverage where the user is located. Others works focus on dedicated wearable sensors in different body parts [12]. However users don't always feel comfortable wearing sensors consequently, it became necessary to find new ways to collect the data.

The rapid adoption of mobile phones comes as an opportunity to collect larger dataset without changing user habits. Today's smartphones are programmable and equipped with a growing set of cheap powerful embedded sensors, such as an accelerometer, gyroscope, Global Positioning System (GPS), microphone, and camera, etc ... The non-intrusiveness and the presence of numerous sensors make the mobile phone the most proposed vehicle for collecting data in transportation mode recognition.

Research on transportation mode recognition that collect data through phones, can use data from GPS [21], accelerometer [12], [18], [16],[22], sometimes a combination of the two [23], or accelerometer data combined with other sensor data [17], [24]. Table 3.1 summarize some relevant works on the topic by looking at five dimensions:

- type of activity modes that can be inferred;
- sensors being used;
- the user base of the dataset;
- hours of data collected;
- accuracy of the classification.

The methodologies that are examined build their model on different data and try to recognize different activities so results are hardly comparable. Each different activity sets that the study tries to recognize, bring a totally different pattern recognition problem. For example, discriminating among walking, running, being still and using a vehicle, turns out to be much easier than incorporating more complex activities such as driving, being on a bus, or being on a tram. Another factor makes it difficult to compare this studies is the quality of the training data, the procedure followed by the individuals while collecting data is critical in any HAR. It's important to train and test activity recognition system on data collected under naturalistic circumstances, because laboratory environments may influence activity patterns. The number of individuals and their physical characteristics are also important factors in any HAR study. A comprehensive study should consider a large number of individuals with diverse characteristics in terms of gender, age, height, weight, and health conditions.

Clearly, most of the works in Table 3.1 have some weaknesses in the quality of the data. Datasets used to build and then evaluate models often lack users, they often consider a time span that is too narrow, or are built on special conditions (e.g. all users follow the same paths). In some cases, the lack is also on the environment; in [13], [14], [15], the proposed models operate well in urban environment, but it certainly would not do just as well in a rural area. Ideal test conditions or reduced training and test set can overestimate their accuracy.

3.1 Work based on GSM

Techniques based on external data show how a cell phone can infer, with good accuracy, the mode of travel by monitoring the fluctuation of GSM signal strength levels and neighboring cell information. Anderson et al. in [13] and Sohn et al. in [14] distinguish between walking, traveling in a motor car, and remaining still with data available on a typical GSM cell phone. Both works take advantage of the fact that GSM signal strength levels and neighboring cell information change minimally when the cell phone is static. Their variation is reflective of the current speed of travel, not of the current mode of travel.

Anderson et a.l in [13] also notice that their model performs well in all classes except for traveling by car. This activity was easily confused with other activities, for example, traffic congestion can be confused with remaining still and waiting at traffic lights with walking. They address this issue by refining the network to include additional information about the task. For

	Activities	**Sensors**	**Users**	**Hours**	**Accuracy**
[13]	still, walking, vehicle	GSM	1	9	82%
[14]	still, walking, vehicle	GSM	3	323	85%
[15]	still, walking, vehicle	GSM, WiFi	2	13	83%
[21]	walking, driving, bus bicycle	GPS	65	2000	75,6%
[23]	still, walking, vehicle running, bicycle	GPS, accelerometer	16	120	93,6%
[12]	still, walking,running, bike, reading, eating ...	accelerometer	20	34	84%
[18]	running, still, jumping, walking	accelerometer	11	1,5	97,5%
[16]	walking, jogging, sitting, descending stairs, ascending stairs, standing	accelerometer	29	13	91.7%
[22]	walking, still, bus, train, metro, tram, car	accelerometer	16	150	80,1%
[24]	still, walking, driving, train, bike, city bus, national bus.	accelerometer, gyroscope	8	6,5	93,9%

Table 3.1: Main characteristics of related work on transportation mode recognition: target transportation mode/activity, type of sensors used for classification, the number of users who have contributed to the creation of the dataset, data collected in terms of hours, and accuracy of the results. Note that the accuracy reported in the table refers to model trained and tested on the dataset used by individual studies, then on completely different datasets.

instance, they remove changes to the mode of travel that occur for short periods of time by using the knowledge that a typical a car journey is not ended and a new one started in a 15-second time period. Based on that, short periods of low signal strength fluctuation can be ignored when placed between periods of high fluctuation. These *task-based* approach increase performance for all classes: accuracy in vehicle class grows from 36% to 80%.

Sohn et al. in [14] reach their best result, in term of accuracy, with a two-stage classification scheme. The first stage classifies an instance as stationary or not stationary. If the instance was classified as not stationary, a second classifier would determine if the instance was walking or driving. Both classifiers were trained using a boosted logistic regression technique [26].

3.2 Work based on GSM and WiFi

After two years Mun et al. in [15], argument that GSM data can perform better with the insertion of features derived from WiFi connectivity for train a classification model. They built a mobility classification model using features derived from GSM and WiFi beacons data [1] to detect the same activity as in [13] and [14] (still, walking, vehicle). Combining two complimentary data sources (GSM and Wifi) gives an accuracy of 83% (accuracy of using only GSM 79% and accuracy of using only WiFi 75%). Both GSM and WiFi based system work well for *coarse-grained* transportation mode classification, such as determining the difference between still and motorized transport, but are not useful for *fine-grained* classification, as to distinguish driving a car from taking a train. Also, features related to GSM and WiFi beacon are strongly dependent on environment characteristics. We reasonably expect a different result on rural environment.

[1]The beacon frame, which is a type of management frame in IEEE 802.11, provides the "heartbeat" of a wireless LAN. Beacon frames are transmitted by the access point periodically to announce the presence of a wireless LAN.

3.3 Work based on GPS

Some researchers have focused on the use of GPS data [21] [23]. In [21], Yu et al. automatically infer transportation modes, including driving a car, walking, taking a bus, and riding a bicycle from raw GPS logs based on supervised learning. The work follows a four-step methodology: *(1)* partition each GPS trajectory into separate segments of different transportation mode; *(2)* from each segment identify the features with poor correlation with the speed; *(3)* build inference model with decision tree; *(4)* conduct a graph-based post-processing algorithm to improve the inference performance of the model.

To increase the inference performance in post-processing algorithm the authors use knowledge derived from a transition matrix among transportation mode based on real data. This matrix summarizes data collected by sixty-five people over a period of ten months during which subjects annotate every change on transportation mode. From the transition matrix they observe some relevant users transportation behaviors that can be used to improve accuracy. For example, in almost all cases, driving, taking a bus, and riding a bicycle, transfer to walking before changing to one another. During post-processing they consider the posterior probability [2] of each segment. If the probability of a segment with inference, for example $segment[i]$, is less than a fixed threshold T_1 the inference of $segment[i-1]$ can be used to revise the prediction of $segment[i]$. Without post-processing, they have reached a 71,5% which increase to 75,6% when adding a post-processing phase.

A scenario of use of the post-processing is shown in Figure 3.1. In the example during $segment[i]$ preliminary inference give *Bike* as the most probable activity (40%), suppose T_1 equal to 0,60, during post-processing the authors use result on previous segment, Driving. To choose what kind of activity refer to $segment[i]$ they calculate posterior probability of $segment[i]$ being different transportation modes conditioned by the transportation mode

[2]The posterior probability of a random event is the conditional probability that is assigned after the relevant evidence is taken into account.

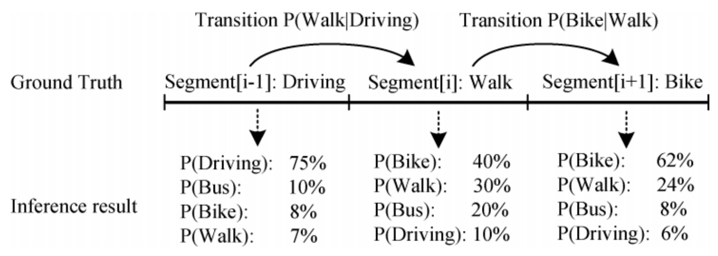


Figure 3.1: Example of post-processing on GPS trajectory. From [21].

of $segment[i-1]$ according to equations 3.1 and 3.2 (an equation for each activity of activity set). After the calculation, authors use the transportation mode with maximum probability.

$$Segment[i].P(Bike) = Segment[i].P(Bike) \times P(Bike|Driving) \quad (3.1)$$

$$Segment[i].P(Walk) = Segment[i].P(Walk) \times P(Walk|Driving) \quad (3.2)$$

GPS-only solutions, like GSM and WiFi solutions, yield good performance in terms of discerning between motorized and not-motorized transportation (*coarse-grained* classification), but might fail in classifying motorized modes with similar speed (*fine-grained* classification).

When considering these works is good to also keep in mind that GPS-based system can be very efficient when GPS signals are available, but they suffer from some important limitations. Besides the high power consumption, GPS receivers depend on unobstructed view to satellites which presents problems in many common situations such as moving underground or inside a station.

3.4 Work based on inertial sensors

To overcome some of the limitations of the GPS-based approaches, more recent studies are leveraging accelerator data and, sometimes, combine that data with data from others inertial sensors embedded in the phone.

Compared to GPS, an accelerometer requires much lower power consumption allowing to monitor the transportation behavior continuously. Furthermore, accelerometers measure user's movements directly and therefore do not depend on any external signal sources, as GSM or WiFi beacon.

Using accelerometer data requires considering the orientation of the sensor. Accelerometer output data tracks acceleration along the three axes (x, y, z).When the orientation of the sensor changes, the coordinate system will rotate accordingly ant the readings at the three axes will change. This is problematic because we expect to find in data representative patterns to distinguish between activities, instead, a non-fixed accelerometer can introduce high fluctuation in data as a consequence of the orientation changes. Then, once accelerometer data are collected, features extraction is strongly related to sensor orientation, if is fixed the identification of pattern is easiest, but things get complicated when the data collection is done without constraints on this characteristic. To overcome this issue it is important to define an *orientation-independent metric.*

Sun et al. in [25] propose an orientation independent sensor reading dimension, which can relieve the effect of the varying orientation on the performance of the activity recognition. Their aim is to distinguish between standing still, walking, running, bicycling, ascending stairs, descending stairs and driving a car assuming that the mobile phone is freely placed in one of the pockets (tests were conducted using front or rear pockets of the trousers, or front pockets of the coat). Under this assumption, the accelerometer sensor inside the phone will take the position and orientation associated with the moving pocket. They pre-process accelerometer data adding to the 3-D vector (acceleration along three axes) a fourth component, the *acceleration magnitude.*

Magnitude is a measure of the quantity of acceleration and has no direction and it is insensitive of the orientation of the mobile phones. For each acceleration vector, its magnitude, that reflects the module of the sensor vector, is defined as: $magnitude(v) = \mid v \mid = \sqrt{v, x^2 + v, y^2 + v, z^2}$.

In [25], data collected from seven subjects for a total of forty-eight hours of registrations, are used to build two support vector machine models, one with and one without the acceleration magnitude. In order to verify the classification contributions of the acceleration magnitude, the study compare result on the test set, founding that the use of magnitude increase classification accuracy.

One of the most exhaustive work, that uses the accelerometer to collect the data, is by Bao and Intille [12]. Their work is based on data acquired using five specific biaxial accelerometers worn simultaneously on different parts of the body with fixed orientation. In their study twenty subjects wore the accelerometers as they perform twenty activities, including daily activities with a different range of intensity such as walking, sitting, standing still, watching TV, running, bicycling, eating or drinking, brushing teeth, etc ...

Data generated by the accelerometers, divided in a 50%-overlapping sliding window of 6.7 seconds, are used to perform features extraction. Bao and Intille extract different features *(1)* mean; *(2)* energy, calculated as the sum of the squared discrete Fast Fourier Transform (FFT) component magnitudes of the signal; *(3)* frequency-domain entropy, calculated as the normalized information entropy of the discrete FFT component magnitudes of the signal; *(4)* correlation features, calculated between all pairwise combinations of axes on different hoarder boards.

Features are used to train a set of classifiers. Decision tree showed the best performance, recognizing activities with an overall accuracy of 84%. Some of these activities exhibit similar or identical body acceleration (watching TV, sitting) so, as expected, there are significant differences in accuracy between the different activities (ranging over from 43.58% to 96.42%).

Bao and Intille train and test classifiers using two protocols:

- *user specific mode* : only a particular user's data is used for training and testing purposes;
- *leave one user out mode* : the classifier is trained with all but one user (fifteen out of sixteen) and tested with the user not in the training set.

Results on test set show how the accuracy was significantly higher for the *leave one user out mode* validation process. This indicates that the effects of individual variation in body acceleration may be dominated by strong commonalities in activity pattern between individuals. This suggests that real-work activity recognition system can rely on classifiers that are pre-trained on large activity data sets to recognize some activities, simplifying the deployment of this kind of system. This result is one of the major contributions given by this work. The other major contribution is the demonstration that the utilization of multiple devices might improve the accuracy of the activity recognition process reducing the classification errors caused by random noise.

Reddy et al. in [23], use accelerometer combined with GPS information to train their model. The dataset built by data from sixteen individuals, perform each activity for fifteen minutes, with six phones attached simultaneously, positioned on the arm, waist, chest, hand, pocket and in a bag with orientations set according to their preference. Collected data are divided into one-second of a non-overlapping window. For accelerometer data, they compute magnitude and extract various features including the mean, variance, energy and the Discrete Fourier Transform (DFT) energy coefficients. Variance along with DFT energy coefficients was selected as the feature set using Correlation-Based Feature selection (CFS). CFS uses heuristic "merit" function that finds the subset that is predictive of the classification groups while reducing redundancy among the features themselves. The most relevant accelerometer derived features for the classification after CFS are: variance and three different DFT coefficients. They perform *two-stage classification*, combining *decision tree* with Discrete Hidden Markov Model (DHMM) reaching an overall accuracy of 93,6%.

Authors investigated how phone placement affects transportation mode accuracy, training model on seven different datasets. First derived from all six positions and each of the remaining six derived from data of a single cell. The result shows that a generalized classifier accuracy is slightly below the accuracy of the others. The average accuracy decrease for the generalized classifier of 1,1%. Thus a generalized classifier can be created so that the user can be agnostic about where to position the phone and still obtain accurate transportation mode inferences.

As in [12] also Reddy et al. in [23] test the user variation in model (*user specific mode* and *leave one user out mode*), reaching the same result, proving that it is possible to achieve good performance without requiring users to provide specific training data as long as the training set contains enough variation in terms of each activity. With the *user specific mode*, the accuracy increases by 2,2% compared to a generalized classifier that is trained and tested on all individuals. So create user specific classifiers would help in terms of classifier performance. With the *leave one user out mode*, an average accuracy of 93,6% and a minimum of 88,2% is obtained.

As seen in the works so far also Hemminki et al. in [16] use only accelerometer data, but from a single accelerometer kept in the user's pocket rather than multiple devices distributed across the body. Accelerometer data was collected from twenty-nine users as they performed daily activities such as walking, jogging, climbing stairs, sitting, and standing. Authors choose a ten-second of non-overlapping window dimension to generate forty-three features, although all features are variants of just six basic features:

- average acceleration (for each axis);
- standard deviation (for each axis);
- average absolute difference between the value of each reading and the mean values;
- average acceleration magnitude;
- milliseconds between peaks in the sinusoidal waves associated whit most activities (for each axis);
- fraction of each reading in ten bins, in which they split the range of values (maximum - minimum).

They use the resulting data to induce a predictive model for activity recognition reaching accuracy between 78.1% and 91.7%. They prove that it is possible to perform activity recognition with commonly available equipment and yet achieve highly accurate results and with no strong restrictions on the position and orientation of the sensors.

Hemminki et al. in [22] use hundred and fifty hours of transportation data collected from sixteen individuals for distinguishing between still, walking, train, bus, metro, tram or car. They divided the data using a sliding 1.2-second window with 50% of overlap. They extract a large list of features.

- *Frame-based features* : fifty-four features based on windows:
  - *statistical domain* (mean, variance, min, max, range, ...);
  - *time domain* (integral, auto-correlation ...);
  - *frequency domain* (FFT, DCT, spectral entropy ...).
- *Peak-based features* : ten features based on *peak areas* that correspond to acceleration or breaking events (intensity, length, ...).
- *Segment-based features* : seventeen features characterize patterns if acceleration and deceleration period over the observed segment (variance of peak features, peak frequency, stationary duration, stationary frequency).

In their work, they demonstrate that *peak-based* and *segment-based* features are not sensitive to the placement of the device.

They decompose main task hierarchically into sub-tasks, proceeding from a coarse-grained classification towards a fine-grained distinction of transportation mode. At the root, there is a classifier that determinate if the user is walking or not walking (*kinematic motion classifier*); if this classifier detect not walking the process progresses to next classifier (*stationary classifier*) that defines whether the user is stationary or in a motorized transport. If motorized transportation is detected, the classification proceeds to the motorized classifier which is responsible for classifying the current transportation activity into one of these modes: bus, train, metro, tram, car. Following this process, they reach 80% of accuracy.

Bedogni et al. in [24] propose the utilization of accelerometer data combined with gyroscope and GPS data. They collect data from eight people performing seven activities: standing still, walking, driving a car, being on a train, driving a bike, being on a city bus, being on a national bus.
They divide the dataset into consecutive and 5-second of a non-overlapping window and for each segment use four statistical-domains features (minimum, maximum, average and standard deviation) derived by *magnitude* dimension.

Authors try to understand which learner and with sensors data is better to use. They choose six different learners, and for each learner use every possible combination of sensor data of the training set. The best results in terms of accuracy are when they use random forest with combining all features (95,44%). At the same time, energy consuming consideration on GPS, make acceptable to lose something in accuracy to save battery on the device. In this case the use of only accelerometer and gyroscope features reaches a 93.91% accuracy.

In spite of the limitations of each of the works briefly presented here and of the relative poor way to compare them, all these studies contribute in a different way to direct future research in transportation mode recognition. Results show that it's possible to use daily device, like phones, to make *fine-grained* classification, without limitation on their position or orientation. Furthermore it seems that is not necessary to build a user-specific classification as long as the training set contains enough variation in terms of each activity. An aspect that certainly merits further study is the selection of the features. Various kind of features, especially for accelerometer data, have been investigated in activity recognition works, including mean, variance, correlation, energy, frequency-domain entropy, etc . . . To the best of our knowledge, little general analysis of the contribution of each feature on different application scenarios has been reported. Applying more features may bring benefits to the recognition accuracy in the case of computing on the powerful computers. However, when we are trying to implement these features inside the resource and power limited mobile phones, we should try to avoid the features that need complex computing since it consumes much resources and energy, which is critical to the user experience and acceptance of such applications.

Chapter 4

Dataset

To perform a classification on transportation mode using sensor data from smartphones we need a collection of raw data with good coverage of activities, users, and devices.

The end goal is to extrapolate a general model that is able to infer what kind of actions the user is performing using unseen data. In order to yield this results, that dataset needs to be large and diverse enough.

In Chapter 3 we saw some studies on transportation mode recognition task and talked about weaknesses in their dataset. Most of them collect data from subjects under artificially constrained laboratory settings. Some evaluate recognition performance on data gathered in natural settings (out-of-lab), but only use limited datasets collected from few individuals or in a particular physical environment. Research using naturalistic data collected from multiple subjects has focused on *coarse-grained* recognition. It is uncertain how prior systems will perform in *fine-grained* classification for a larger sample population under real-world conditions.

The lack in the literature of datasets that exceeds the limitations that we have listed has pushed us to collect a new dataset.

Our requirements for the dataset are:

- wide user base, with different gender, age, and occupation;

- collection occurs under real-world conditions; users during the experiments are free to use their device, carry, and move it as they want.

4.1 Data collection

In order to address the activity recognition, we collected sensors data from thirteen volunteer subjects, ten male, and three female. Volunteers had to carry a smartphone while performing a specific set of activities.

The dataset was built without any constraint on the device position. Subjects were free to to carry and use their device during the experiments in the way they wanted (refer to section 3.4 for some of the challenges around the variability of the orientation of the device). Furthermore, no restriction has been imposed even on device's characteristics (e.g. type of embedded sensors) except for the fact that it must be an Android mobile phone. The reason for using only Android devices is related to the data collection process itself and not to the quality of the data. These choices increase the realism of the experiments, but introduce additional complexity.

Table 4.1 summarize the data collected by users by looking at five dimensions: gender, age, occupation, device model, and Android version installed while they collected data.

Data collection was controlled by an Android application running on the user's phone as they performed activities. This application, through a simple graphical user interface, shown in Figure 4.1, permitted volunteers to record their name, start and stop the data collection, and label the activity being performed. We asked users to use the application during specific activities such as walking, being on a car or on a train or on a bus or standing still. Above we refer to activities with this abbreviations: $TM = \{bus, car, train, still, walking\}$.

The application registers each sensor event with a maximum frequency of 20 Hz. Events occurs every time a sensor detects a change in the parameters it is measuring, providing four pieces of information:

	Gender	**Age**	**Occupation**	**Device**	**Android Version**
$User_1$	male	30	student	LG G2	5.0.2
$User_2$	male	27	student	Sony XPERIA Z3 Compact D5803	6.0.1
$User_3$	male	30	student	Nexus 5	7.0
$User_4$	male	36	office worker	Huawei Honor 5X	6.0.1
$User_5$	male	36	stage director	RIC	6.0.1
$User_6$	male	27	resercher	Samsung galaxy s3 neo	4.4.2
$User_7$	male	32	cameramen	Samsung S7	6.0.1
$User_8$	female	32	bartender	Huawei Tag-l01	5.1
$User_9$	female	24	student	Motorola Moto G	5.1
$User_{10}$	male	22	student	Huswei P9	7.0
$User_{11}$	female	31	office worker	Nexus 5	7.0
$User_{12}$	male	31	resercher	Samsung Galaxy S6	6.0.1
$User_{13}$	male	60	retired	Nexus 5	7.0

Table 4.1: Dataset variability in terms of user's in term of age, sex, occupation, device's model and Android version while data are collected.

- the name of the sensor that triggered the event;
- the timestamp for the event;
- the accuracy of the event;
- the raw sensor data that triggered the event.

The length and content of the raw sensor data depend on the type of sensor. Each raw data registered has the following structure:

$$< timestamp, sensor_i, sensorOutput_i >$$

The application saves each sample on a `csv` file on the device. The maximum frequency of sampling (20Hz) ensures small file size suitable to be stored in

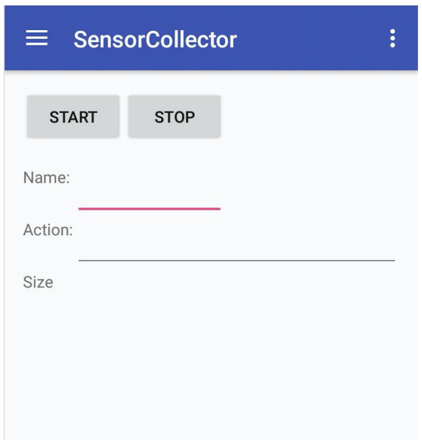


Figure 4.1: Initial screen of the application used to collect data (*Sensor collector*).

a smartphone. The file's data was collected directly from files stored on the phones via a USB connection.

In total, we gathered 226 labeled files representing the same number of activity corresponding to more than 31 hours of data: 26% of data is annotated as walking, 25% as driving a car, 24% as standing still, 20% as being on a train, and 5% as being on a bus. A detailed composition of dataset is in Table 4.2.

Result files contained raw data from twenty-three sensors. Twenty-three is the number of sensors of which we collect data (we will see that not all will be used for the model though). Instead of deciding before hands which sensors to use based on the task being performed, we choose to collect data from all sensors available on the device and then we decide which sensor is good to keep.

As we mentioned before, we only constrained the device to be and Android phone. Users were not equipped with the same model and this led to high variability in the terms of the sensors actually available. All mobile devices had an accelerometer but other sensors were less common.

	Bus	**Car**	**Still**	**Train**	**Walking**	
$User_1$	00:53:35	02:46:10	02:46:10	03:17:00	03:09:55	**12:18:00**
$User_2$	-	00:06:55	00:12:05	-	01:35:30	**01:54:20**
$User_3$	00:14:40	-	-	00:14:45	00:51:25	**01:20:50**
$User_4$	-	00:07:20	00:18:55	-	01:08:35	**01:34:50**
$User_5$	-	-	00:02:25	-	00:04:45	**00:07:10**
$User_6$	00:07:25	00:07:25	00:19:25	00:23:40	00:07:25	**01:21:30**
$User_7$	00:13:15	-	-	01:20:25	-	**01:33:40**
$User_8$	-	01:13:25	00:10:25	-	-	**01:23:50**
$User_9$	-	-	-	00:13:15	00:15:35	**00:28:50**
$User_{10}$	-	00:40:45	02:30:00	-	00:18:10	**03:28:55**
$User_{11}$	-	01:07:25	-	-	-	**01:07:25**
$User_{12}$	00:15:40	00:52:10	01:45:00	00:51:20	00:49:05	**04:33:15**
$User_{13}$	-	00:36:15	-	-	-	**00:36:15**
	01:44:35	**07:53:50**	**07:29:35**	**06:20:25**	**08:20:25**	**31:48:50**

Table 4.2: Detailed dimension of dataset in terms of time user's contribution.

4.2 Inside sensor data

As a direct result of the user's freedom in terms of device, we observe variability of a presence of embedded sensors. Different mobile phone have different type sensors, the common subset is very small related to the total number of sensors in each phone. In our study we exclude sensors that are not supported by enough devices (support is too low), and sensor that collect data that can me too misleading for the actual classification of the activity.

Considering $sensor_i$ we define its *support* in two ways:

- *user support* : numbers of user's device that have $sensor_i$ over the total number of devices;
- *activity support* : activities in labeled raw data that have $sensor_i$ over the total of all considered activities.

Greater user and activity support may mean greater representativeness of sensor data, thus increasing the possibility of recognizing a given pattern through its data. If a sensor has low user support, user specific habits on performing an action can affect very strongly the measurement thus reducing the generality of the model.

Moreover, using data derived from a sensor with a non-complete support in the activities would have an even worse effect on result model than the previous case due to the pre-processing phase. Because of this variability, our dataset is sparse, as we saw in 2.3.3, in the pre-processing phase we have to manage the *nan values* “imputing” a value. Making the right decision for imputed values with not enough information can be dangerous and result in a bad choice that compromises the model.

Accordingly to the above considerations, we choose to remove sensor with non-complete activity support and fix a threshold value for user support, below which the sensor is excluded from the dataset ($t_{sensor} = 0.60$). A detailed look at the sensor support is in Table 4.3.

Sensor	**User support**	**Bus**	**Car**	**Still**	**Train**	**Walking**
accelerometer	100%	x	x	x	x	x
sound	100%	x	x	x	x	x
light	100%	x	x	x	x	x
speed	92%	x	x	x	x	x
gravity	85%	x	x	x	x	x
orientation	85%	x	x	x	x	x
magnetic field	85%	x	x	x	x	x
linear acceleration	85%	x	x	x	x	x
gyroscope	77%	x	x	x	x	x
rotation vector	77%	x	x	x	x	x
proximity	69%	x	x	x	x	x

magnetic field uncalibrated	69%	x	x	x	x	x
game rotation vector	69%	x	x	x	x	x
gyroscope uncalibrated	69%	x	x	x	x	x
pressure	62%	x	x	x	x	x
step counter	*53%*	x	x	x	x	x
tilt detector	*38%*	x	x	x	x	x
geomagnetic rotation vector	*31%*	x	x	x	x	x
step detector	*31%*	x			x	x
qti sensor rmd	*8%*	x	x	x	x	x
qti sensor amd	*8%*	x	x	x	x	x
internal temperature	*8%*	x			x	x
gesture	*8%*		x			

Table 4.3: Sensors user and activity support in our dataset. The sensors immediately excluded from the dataset for non complete activity support or user suppor lower than 0.5 are highlighted in red.

Support is not the only reason behind the exclusion of a sensor from model training. We also have to consider the meaning of the remaining sensors, and address if they are relevant for the transportation mode recognition task. If we include in the dataset raw data from sensors that are not relevant for the task, we risk to compromise the goodness of the resulting model.

There are also other considerations that we have to do before decide if a sensor will be or not in the dataset, like type of sensor and sensor battery consumption level.

Type of sensor : *base sensors*, that relay data from a single physical sensor are preferred over *logical sensors*, and over *uncalibrated sensors*. Logical sensors generates data by processing and/or fusing data from one or several physical sensors, while *uncalibrated sensors* provide more raw results but may include some bias.

Note that base sensors are not equal to their underlying physical sensor. The data from a base sensor is not the raw output of the physical sensor because corrections (such as bias compensation and temperature compensation) are applied.

Figure 4.2 shows the most common sensors, their type and their relationship.

Sensor battery consumption level : this parameter is relevant in any mobile environment. High battery consumption leads to a lower usability.

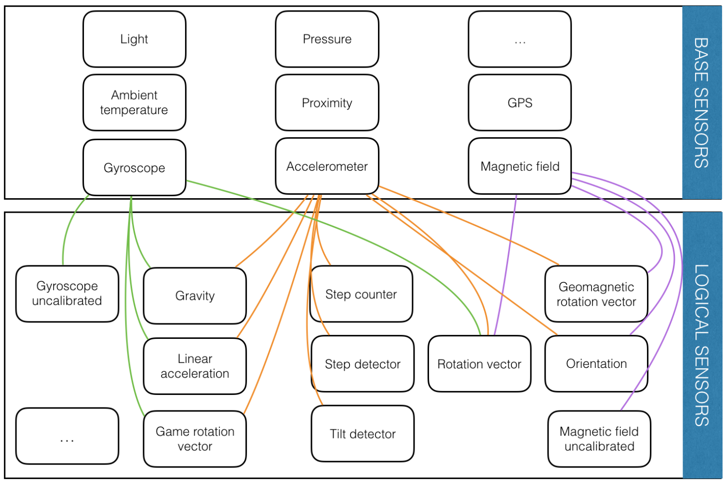


Figure 4.2: Base and logic sensors in Android phones.

Once evaluation parameters are defined we can look to the remaining sixteen sensors and make the appropriate considerations.

We first analyze **motion sensors**, that measures acceleration forces and rotational forces along three axes.

Accelerometer : measures the acceleration force in m/s^2 that is applied to a device on all three physical axes, including the force of gravity.

Gravity : measures the force of gravity in m/s^2 that is applied to a device on all three physical axes.

Gyroscope : measures a device's rate of rotation in rad/s around each of the three physical axes.

Gyroscope uncalibrated : similar to gyroscope but no gyro drift compensation has been performed to adjust the given sensor values.

Rotation vector : measures the orientation of a device by providing the three elements of the device's rotation vector.

Game rotation vector : identical to rotation vector except that it doesn't use the geomagnetic field. Therefore the y-axis doesn't point north, but instead to some other reference.

Linear acceleration : measures the acceleration force in m/s^2 that is applied to a device on all three physical axes, excluding the force of gravity.

Speed : measure instantaneous speed of the device.

All motion sensors can be relevant to us, but we prefer accelerometer, gyroscope, and speed because they are the only base sensors in this category. Game rotation vector, gravity, and linear acceleration are all based on their data. Rotation vector instead is based also on magnetometer sensor. Speed, instead, is a GPS-derived information, and unfortunately, using the GPS chip, consumes energy which may negatively impact user experience because of battery drainage.

Other sensors that we collect data from are **ambient sensors** that measure various environmental parameters, such as ambient air temperature and pressure, illumination ...

Pressure : measures the ambient air pressure in hPa or $mbar$.

Sound : measures the level of noise through microphone audio source.

Light : measures the ambient light level (illumination) in lx.

Ambient sensors are all base sensor but their output can be strongly related to the environment where data are collected more than the activity that is performed. For example, the pressure value is dependent on the user position when the experiments are taking, if the user walks near the sea or on top of a mountain the pressure output is completely different in spite of the action is the same. Also, light output value can be influenced by where the user holds the phone during the measurements, in the pocket rather than in hands. The value of the light is greatly influenced by the way the user interacts with the phone during experiments, for example consider difference from collecting data while user is walking with cellphone on the pocket or in hands.

At last, the sound, even when influenced by how the mobile phone is used during a given task, is the most reasonably related with the activity. Just consider the difference of sounds between being on a train and walking.

The last type of sensors we collected are **position sensors**, that measure the physical position of a device. This category includes orientation sensors and magnetometers.

Magnetometer : measures the strength and perhaps the direction of geomagnetic fields for all three physical axes in μT (microtesla).

Magnetometer uncalibrated : similar to magnetometer but the calibration is not considered in the given sensor values.

Proximity : measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.

Orientation : measures degrees of rotation that a device makes around all three physical axes.

Step counter : number of steps taken by the user since the last reboot while the sensor was activated.

The orientation data are the only ones in this category to be useful to our task. Magnetometer output is location dependent and can be interference by the presence of permanent magnets or electronic items, as we already said in 2.2, so it's not a good sensor to include. Proximity is the only base sensor in this category, but its data are related only to user relation with the device during experiments, giving no relevant information to detect activity. Lastly, the step counter returns data related to the last reboot, so the dimension data depend on when last reboot is made.

4.3 Data transformation

According to 4.3, our dataset is based on fifteen sensors. Each sensor returns an array of values, with different lengths and content. Before use them is important to consider if they are influenced by the orientation of the device during experiments.

Some sensors, like ambiental (sound, light and pressure) and proximity, returns a single data value as the result of sense, this can be directly used in dataset. Instead, all the other return more than one values that are related to the coordinate system used, so their values are strongly related to orientation.

For almost all we can use an orientation-independent metric, *magnitude* (defined in Section 3.4). Magnitude is an appropriate transformation for all the remaining sensor except for rotation vector and game rotation vector, that capture the rotation of the device, rotation can be described by an angle θ.

All the transformations made to the raw sensors data are summarized in Table 4.4. As evidence that the use of the magnitude metric preserves patterns in the data in the Figure 4.3 shows magnitude trends for all considered transportation modes.

Sensor	**Sensor output**	**Meaning**	**Change**	**Dataset output**
sound	1 float	level of noise	-	1 float
light	1 float	illuminance	-	1 float
pressure	1 float	pressure	-	1 float
speed	1 float	speed	-	1 float
proximity	1 float	proximity to the screen of the device	-	1 float
accelerometer	3 float	acceleration force along 3 axes	magnitude	1 float
linear acceleration	3 float	acceleration force along 3 axes (excluding gravity)	magnitude	1 float
gravity	3 float	direction and magnitude of gravity.	magnitude	1 float
orientation	3 float	angle around 3 axes	magnitude	1 float
magnetic field	3 float	geomagnetic field along 3 axes	magnitude	1 float
magnetic field uncalibrated	6 float	geomagnetic field uncalibrated along 3 axes, bias along 3 axes	magnitude	1 float
gyroscope	3 float	angular speed around 3 axes	magnitude	1 float
gyroscope uncalibrated	6 float	angular speed (w/o drift compensation) around 3 axes estimated drift around 3axes	magnitude	1 float

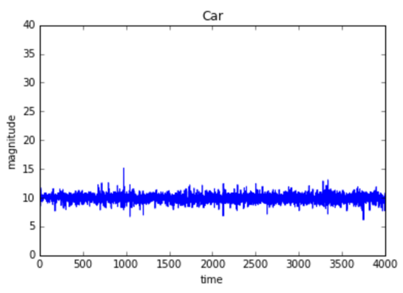
rotation vector	4 float	rotation vector along 3 axes, scalar of the rotation vector $\cos(\frac{\theta}{2})$	compute θ	1 float
game rotation vector	4 float	rotation vector along 3 axes, scalar of the rotation vector $\cos(\frac{\theta}{2})$, (without using the geomagnetic field)	compute θ	1 float

Table 4.4: Summarize all transformation made to the raw sensors data.

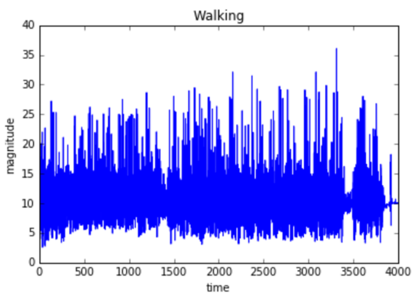
4.4 Features

In order to facilitate the processing of data, the samples need to be cut, namely windowed. The size of the time window depends on the types of actions to be recognized. If the adopted time of the sliding window is too short, the window data may not have covered the information of a complete action, If the width of the sliding window is too long, it will not only make the data sophisticated and increase the amount of calculation. So we first must transform the raw time series data into examples. To accomplish this we divided the data into five-second non-overlapping intervals or windows. Each example, summarize the user activity over an interval. An example is labeled with the activity that occurred while that data was being collected. Each raw of the output dataset D represents the status of all the sensors in the time window, as following:

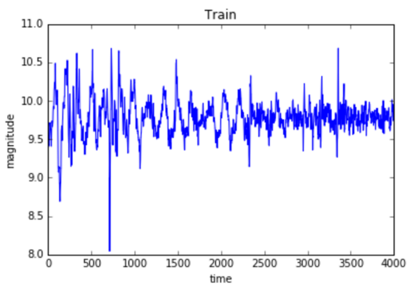
$$< window, sensorOutput_1, ..., sensorOutput_n, activity >$$



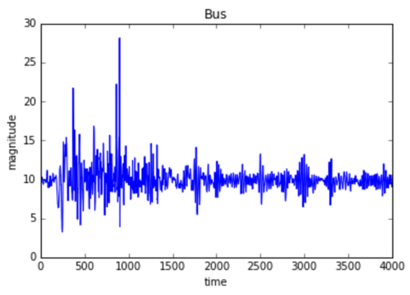
(a) Magnitude (car).



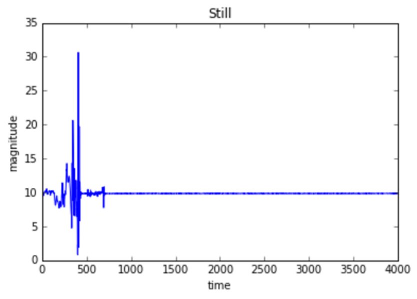
(b) Magnitude (walking).



(c) Magnitude (train).



(d) Magnitude (bus).



(e) Magnitude (still).

Figure 4.3: The the magnitude of all considered transportation modes: being on a car (Figure 4.3a), walking (Figure 4.3b), being on a train (Figure 4.3c), being on a bus (Figure 4.3d), and standing still (Figure 4.3e)

We chose a five-second window because we felt that it provided sufficient time to capture several repetitions of the motions involved in activities. Although we have not performed experiments to determine the optimal example duration value.

Next, we generated statistical features based on the multiple raw sensor readings. We generated a total of sixty summary features, four for each sensor:

- maximum;
- minimum;
- mean;
- standard deviation.

4.5 Handling missing values

Our dataset has many missing values, and they must be imputed before a classification algorithm can be trained (as we already said in Section 2.3.3.

The data is missing because the measurement was not taken often as a consequence of non-availability of the sensor in the device. So there is not information around the reason for the unavailability of the information. The strategy to discard entire rows and/or columns containing missing values, considering the number of missing data, we would lose too much information. We decide to impute the missing values, inferring them from the known part of the data. We decide to replace missing value with mean.

Chapter 5

Classification

As we discussed extensively in the previous chapters, transportation mode recognition plays an important role in context-aware applications. Many studies utilize mobile phone sensors to collect data and then apply machine machine learning techniques to build a model. However, current methods have two key limitations, first on the dataset, on the way it is collected as on the dimension in user's, and second on sensors choice.

We try to overcome the first limitation collecting data from thirteen users without any restriction on the device's position and on the sensor device characteristics as we see in Chapter 2.3.3. In this Chapter instead, we investigate how transportation mode recognition can benefit from the use of sensor data from sensors already available on mobile phones. In literature, as we see in Chapter 3, the use of GSM, GPS, accelerometer and gyroscope data has already been explored reaching good accuracy results. Unfortunately, due to the lack of recognized dataset by the scientific community, compare these results in not possible. To the best of our knowledge, no study has been conducted on the possibility to take advantage from other device's sensors.

Before expanding the base sensors on which the classification model is built, is important to consider the current mobile context. Mobile and handheld devices are generally constrained due to resource limitations primarily caused by limited battery life, limited size of memory or limited power of the processor. Limitations on resource motivated us to carefully consider if, how, and when to use sensors data. We also have to consider that many sensors are always active and already providing services to the system, then collection and use will not result in a greater battery consumption. However, we still have to consider memory and CPU consumption needed to process the data because that can impact other applications and the overall user experience.

In this thesis, we do not explore the possibilities for system architecture, but we look at *mobile cloud computing* [1] as a possible way to overcome the resource limitations in mobile devices context. Appears clear the need of a trade-off between the consumption of resources and the accuracy of the model. Obviously, this equilibrium point changes under different scenarios. Applications have different accuracy requirements or different use cases or even different devices on which they are used. All of these variations can make more or less relevant resource consumption.

For these reasons it seems hard to find one model right for all situations but it is clear the importance to understand which and how much mobile phone sensors data can help in distinguishing between different user activity. In Chapter 2.3.3 we have selected fifteen sensors among those that have been collected by our volunteers, based on activity and user support (4.3). Among these fifteen sensors, there are some whose data can give their contribution to the transportation mode recognition, others which introduce only noise. Based on meaning reasons detailed in Section 4.2 we exclude from data used for training model the following sensors:

- light;
- pression;
- magnetic field;
- magnetic field uncalibrated;

[1]Mobile cloud computing is the combination of cloud computing, mobile computing and wireless networks to bring rich computational resources to mobile users, network operators, as well as cloud computing providers.

- gravity;
- proximity.

From remaining nine possibly relevant sensors we have created three different sensors set, summaries in 5.1. First set contain only three sensors whose are base sensors and have almost complete or complete users support (accelerometer, gyroscope, and sound). The second set contains sensors from first set plus all the other sensor excluded speed, that is a GPS-based sensor, for its high consumption of the battery. Lastly, the third set contains all sensor that we define as relevant for the task.

Sensors of first classification	**Sensors of second classification**	**Sensors of third classification**
Accelerometer	Accelerometer	Accelerometer
Sound	Sound	Sound
	Orientation	Orientation
	Linear acceleration	Linear acceleration
		Speed
Gyroscope	Gyroscope	Gyroscope
	Rotation vector	Rotation vector
	Game rotation vector	Game rotation vector
	Gyroscope uncalibrated	Gyroscope uncalibrated

Table 5.1: Three sets of sensors on the basis of which we train three models to be able to compare results on the same test set.

Our main purpose is to create three different models, based on these three different set and see which of them perform better on the test set and try to understand why.

For each set, we build four model with four different classification algorithms: Decision Trees (DT), Random Forest (RF), Support Vector Machines (SVM), and Neural Network (NN). Choosing every time the model that is the most accurate for transportation mode inference on the test set. NN and SVM, require accurate selection of thresholds and parameters. In section 2.3.2, we already discussed that learning the parameters of a prediction function and testing it on the same data is a methodological mistake that causes overfitting. To avoid overfitting we use k-fold cross-validation. With this approach the training set is split into k groups of samples of equal size, called *folds*. The following procedure is followed for each of the k folds:

- a model is trained using $k - 1$ of the folds as training data;
- the resulting model is validated on the remaining part of the data.

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop.

5.1 Five classes classification

5.1.1 Single sensors dataset

As we saw in Chapter 3 many classification algorithms as been used to address the transportation mode detection task. Random forest is widely used and seems to perform pretty well on different datasets. Therefore, for a first investigation of how the sensors can be discriminating with the defined classes, we choose random forest algorithm.

We first restricted the dataset to the features related to a single sensor, trained the model on this new dataset, and then tested on the test set. This preliminary result shows in Figure 5.1 gives us the idea of how great can be the impact of different sensors.

Accuracy's values with only one sensor are between 0.57 and 0.75, so all considered sensors have the capability to capture some pattern related to the activity the user is performing during the measurements.

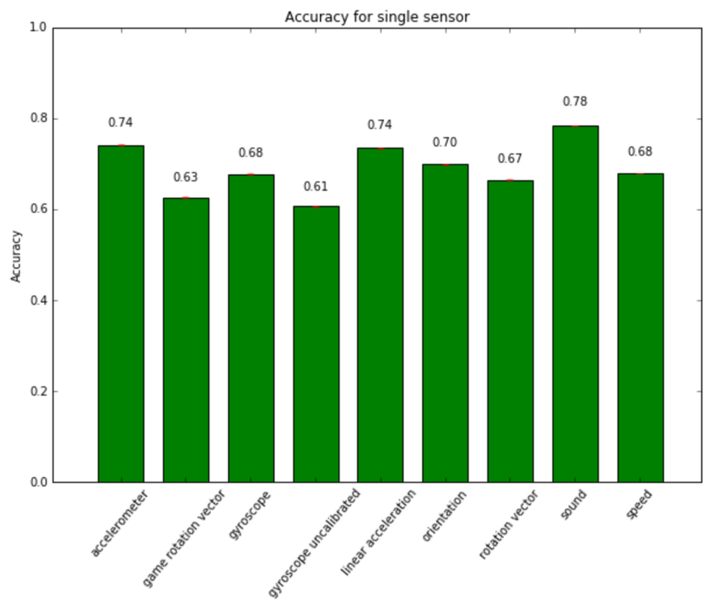


Figure 5.1: The accuracy of the model trained on a dataset composed of features of a single sensor with random forest algorithm.

According to what we could expect, accelerometer and linear acceleration data are two of most discriminating of the considered sensors. Even if no studies have investigated the use of the linear acceleration data, its relevance is the direct consequence of accelerometer, because its data are based only on accelerometer sensors. Gyroscope reaches good accuracy, but this result was to be expected from previous studies. The most interesting result is the high relevance of sound's data. At the best of our knowledge no one has investigated the use of microphone data, although it is an essential sensor for mobile phone use, so is present in all devices.

5.1.2 Three sensors set dataset

For reasons that we already explained previously, we start from dataset formed by a smaller set of sensors. Sensors included in the first set are accelerometer, sound, and gyroscope. These three sensors have the highest values of accuracy taken individually. First dataset D_{first} is formed by twelve features, four for each sensor. We perform classification with the four classification algorithms mentioned before. The overall accuracy for algorithms is between 82% and 88%. Even if random forest produce the highest accuracy values (88%), all algorithms perform substantially well.

By expanding the dataset adding all other relevant sensors except speed, for battery saving purposes, we reach better results in term of accuracy. With second set dataset D_{second}, formed by eight sensor and thirty-two features, accuracy increases up to values between 86% and 93%.

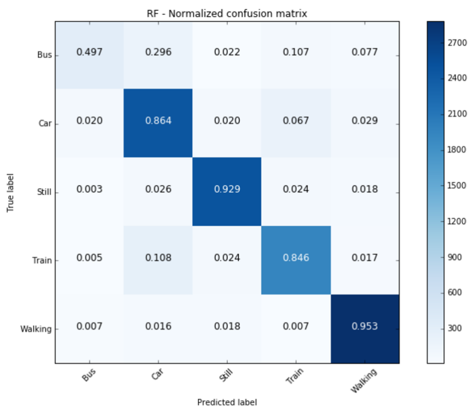
Lastly we train a model on the third dataset D_{third} formed by all nine relevant sensors and thirty-six features, differ from previous D_{second} only for speed derived features. Result show how considering speed, further increasing the ability of the model to infer which transportation mode the user is currently using. In this last case, the accuracy reached a range level between 91% and 96%

Detailed values for overall accuracy in all different dataset and for all algorithm is in Table 5.2

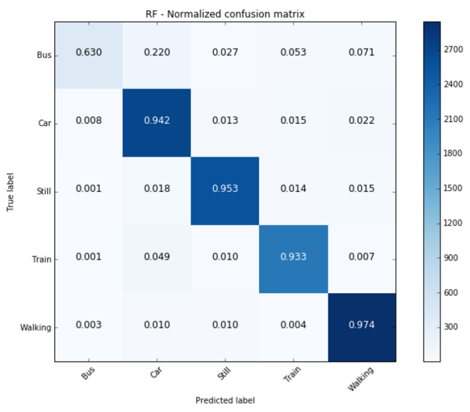
Algorithm	**Accuracy of first dataset**	**Accuracy of second dataset**	**Accuracy of third dataset**
Decision Tree (DT)	82%	86%	91%
Random Forest (RF)	88%	93%	96%
Support Vector Machine (SVM)	85%	93%	95%
Neural Network (NN)	85%	92%	95%

Table 5.2: Overall accuracy with all four classification algorithms on the test set. The three different models are training on different datasets, each dataset varies set of sensors that consider.

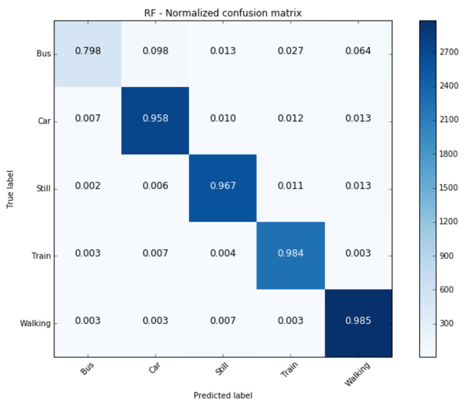
In each model, regardless of the sensor on which dataset's features are based, the accuracy is quite high. However accuracy varies significantly for different transportation modes, since some patterns appear more difficult to identify correctly than others.



(a) First model.



(b) Second model.

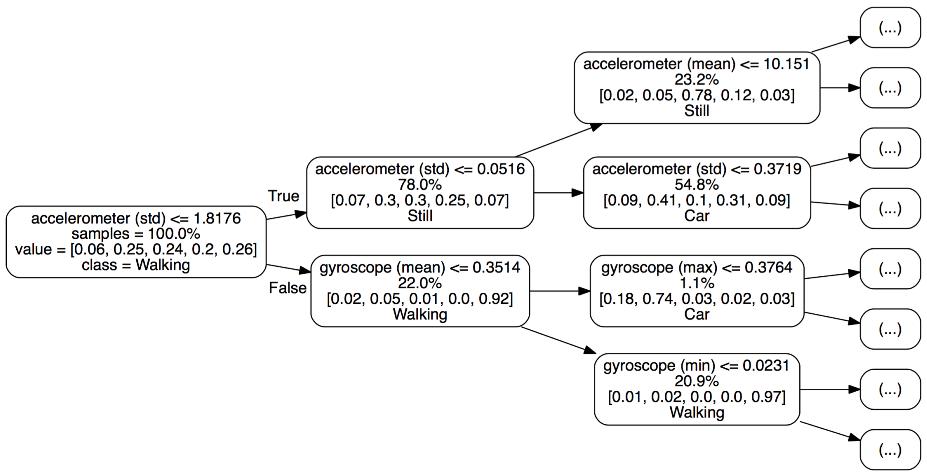


(c) Third model.

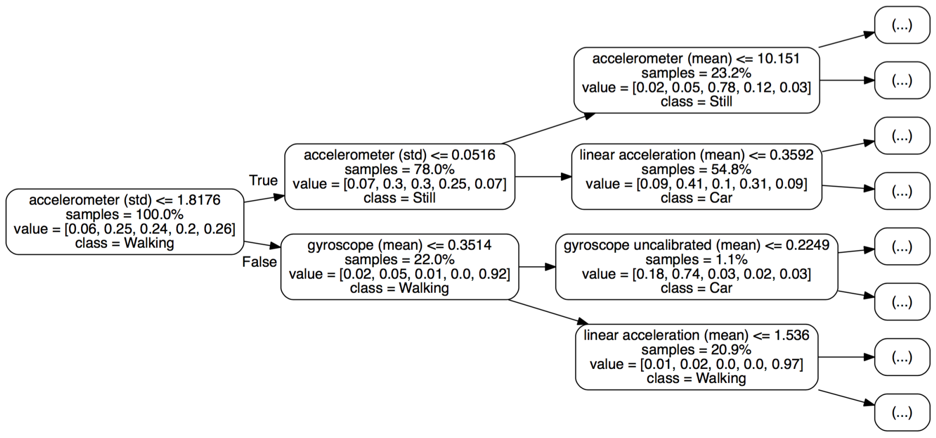
Figure 5.2: Confusion matrices of the three models builds with random forest on three different datasets correspond to three sensors set, with an increased coverage on relevant sensors.

Confusion matrices for random forest models, relative to three models, are shown in Figure 5.2. Even if each of them has different accuracy values, all bring out the same difficulty in distinguishing the same pairs of classes.

The activity that is more difficult to distinguish from the others is being on a bus. We observe significant misclassifications between being on a bus and being on a car, which can be explained by the similarity between these classes. Extending dataset features increases capability of the models to distinguish even between such similar classes. It's clear that the extension of the dataset with the features derived from several sensors, relevant to the task, improves the ability of the model inference.



(a) First model.


accelerometer (std) <= 1.8176
samples = 100.0%
value = [0.06, 0.25, 0.24, 0.2, 0.26]
class = Walking
True
False
accelerometer (std) <= 0.0516
samples = 78.0%
value = [0.07, 0.3, 0.3, 0.25, 0.07]
class = Still
gyroscope (mean) <= 0.3514
samples = 22.0%
value = [0.02, 0.05, 0.01, 0.0, 0.92]
class = Walking
accelerometer (mean) <= 10.151
samples = 23.2%
value = [0.02, 0.05, 0.78, 0.12, 0.03]
class = Still
linear acceleration (mean) <= 0.3592
samples = 54.8%
value = [0.09, 0.41, 0.1, 0.31, 0.09]
class = Car
gyroscope uncalibrated (mean) <= 0.2249
samples = 1.1%
value = [0.18, 0.74, 0.03, 0.02, 0.03]
class = Car
linear acceleration (mean) <= 1.536
samples = 20.9%
value = [0.01, 0.02, 0.0, 0.0, 0.97]
class = Walking
(...)

(b) Second and third model.

Figure 5.3: First two levels of decision trees of the models build on three different datasets correspond to three sensors set, with an increased coverage on relevant sensors.

Another result from confusion matrices is that walking is consistently the most easily identifiable activity, with no strong difference between three models, because rules for walking recognition are mostly based on features relative to accelerometer and gyroscope. Therefore adding other sensors to the dataset improve slightly the accuracy, this consideration can be clearly visualized in Figure 5.3. The Figure 5.3 show the first two level of all three decision tree build on the three different dataset. As we can observe they are exactly the same.

Comparing the three matrices we can see that the use of different datasets, composed by different sensors data, have different effects on the ability to recognize the transportation mode. Tables 5.3 and 5.4 are built to show how changing the dataset has a different effect on the capacity of the model to infer the transportation mode. Although a more detailed analysis is deferred until the next section, it is interesting to see how speed features has no effect on the capacity of the model to distinguish between standing still and being on a bus or walking.

	Bus	Car	Still	Train	Walking
Bus	+ 24,04%	-20,79%	-5,26%	-47,17%	-6,5%
Car	-57,89%	+ 8,8%	-31,58%	-74,63%	-26,67%
Still	-50%	-29,63%	+2,81%	-53,57%	-5,88%
Train	-60%	-59,13%	-54,17%	+10,69%	-42,86%
Walking	-50%	-33,33%	-31,25%	-42,86%	+1,78%

Table 5.3: Differences in confusion matrix from classification with random forest between first dataset, composed by accelerometer, sound and gyroscope features, and the second dataset, composed by features from all sensors excluded speed.

As we said in Chapter 4, we collect more than thirty-one hours of measurements, and we use half for the trainig and half to the test set. We investigate result in term of accuracy on the reduction of the training set size. Starting from the use of 10% of the training dataset every time we add more examples until 100% of training set is used to train model. To understand the distance in terms of dimension and time, consider that 10% of training set is

	Bus	Car	Still	Train	Walking
Bus	+29,48%	-61,25%	-5,55%	-57,14%	-4,17%
Car	-12,5%	+1,70%	-15,38%	-29,41%	-36,36%
Still	0%	-63,16%	+1,68%	-15,38%	-12,5%
Train	-50%	-87,23%	63,63%	+5,47%	-50%
Walking	0%	-60%	-45,45%	-25%	+1,23%

Table 5.4: Differences in confusion matrix from classification with random forest between second dataset, composed by features from all sensors excluded speed, and the second, in which there are the features based on the speed.

just 1145 examples, almost two hours of measurements. In Figure 5.4 there is the result of this experiments for three dataset. As expected, the accuracy growth trend is the same. The greater result of this test is that also if the model is trained on a poor training set can perform pretty good on unseen data. We suppose that this good result is the direct consequence of the user's variability on dataset.

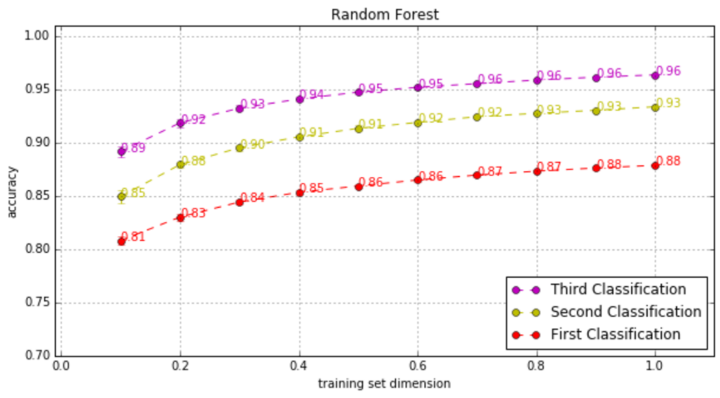


Figure 5.4: The accuracy of three models builds with random forest, trained with a different dimension of the training set, and test on the complete test set.

5.2 Two classes classification

Considering the diversity of transportation modes that we try to recognize (walking, driving a car, being on a train or on a bus or standing still) our classification can be defined as a *fine-grained* classification. As explained previously, distinguishing between activities gets hared if the difference between classes is small. So we also investigate how results change to the restriction of target classes. This kind of results can be relevant, because many applications do not need to recognize one activity between many activities, but they need to identify only activity one or a small set. We start this kind of consideration in previous section looking at Tables 5.3 and 5.4.

A more systematic analysis has been conducted on each pair of transportation mode included in our target classes. Like five class classification, we try to addestrate our models with four classificaion algorithms, finding that also on two class classification random forest perform better than the others classification algorithms.

Last result merits further study, because for the parameterization of the SVM and of the neural network classifier, we use the same choises we made in the five classes classification. Definitely better results would be obtained with a more careful choice of the parameters. However, we do not expect any change to the accuracy growth trend.

Without any exception, reducing the set of target classes increases the accuracy of the model. Table 5.5 reports all results in terms of accuracy for two class classification when the random forest algorithm is used. However when we train model on first dataset, that is based on few key sensors, accuracy increase more in percentage compared to the five classes classification accuracy. Overall accuracy on complete dataset, obtained with the model trained on first datasetk reach 88% accuracy, while reducing the target class to two the accuracy move to the range of 90,24%-97,57% with a growth rate between 2,55% to 10,87%. Instead, when we consider models builds on second or third dataset, the increase of accuracy, when we reduce the target classes from five to two, is lower: with second dataset is between 1,92% to

6,30%, and with third dataset, is between 1,10% to 3,57%.

	Accuracy on first dataset	**Accuracy on second dataset**	**Accuracy on third dataset**
accuracy with five target classes	*88%*	*93%*	*96%*
(Train, Walking)	97,57%	98,85%	99,42%
(Still, Walking)	97,53%	98,60%	98,84%
(Bus, Walking)	97,47%	97,70%	98,19%
(Car, Walking)	96,92%	98,03%	98,76%
(Bus, Still)	96,81%	98,42%	98,63%
(Car, Still)	96,32%	97,55%	98,24%
(Still, Train)	96,06%	97,52%	98,69%
(Bus, Train)	94,43%	97,39%	98,42%
(Bus, Car)	91,70%	94,79%	97,06%
(Car, Train)	90,24%	96,24%	98,91%

Table 5.5: Accuracy of models in distringuish between each combination of our class trasportation mode. Models are build with random forest, and trained on three datasets with an increasing number of features. Couple of transportation mode in Table are ordered based on decreasing accuracy respect on first dataset.

In the first lines of Table 5.5 we find group of two target classes that can ensure greater ability of the model to recognize them with few sensors of the first dataset: accelerometer, gyroscope and sound. However, this order changes a little if we train model the others two dataset. Looking at the table from this point of view, confirms what had already emerged from the confusion matrices of previous section: detecting when a user is walking is the simplest sub-task of our classification task. Instead, distinguishing between motorized transportation mode, like being on a train, on a car, or on a bus is the hardest sub-task. The hard task take more benefits, than the simplest one, by the extension datasets to relevant sensor's features.

In order to go deeper on how the addition of features related to sensors can increase the ability of inference of a model with respect to the classes that you want to classify we analyze the most relevant feature for each group of two target class. Results are in Table 5.6.

	Relevant features in D_{first}	**Relevant features in D_{second}**	**Relevant features in D_{third}**
(Train, Walking)	$gyro_{mean}$ 0.197	$gyro_{mean}$ 0.143	$gyro_{max}$ 0.162
	$gyro_{std}$ 0.186	acc_{std} 0.119	$gyro_{std}$ 0.123
	acc_{std} 0.162	$gyro_{max}$ 0.118	$gyro_{mean}$ 0.01
	$gyro_{max}$ 0.146	acc_{max} 0.093	acc_{std} 0.074
(Still, Walking)	acc_{std} 0.264	acc_{std} 0.154	acc_{std} 0.169
	acc_{max} 0.182	$linearA_{mean}$ 0.133	$linearA_{mean}$ 0.115
	$gyro_{max}$ 0.102	$linearA_{min}$ 0.095	$gyro_{max}$ 0.083
	$gyro_{mean}$ 0.099	acc_{max} 0.087	$linearA_{max}$ 0.074
(Bus, Walking)	$gyro_{mean}$ 0.202	$gyro_{mean}$ 0.134	$gyro_{mean}$ 0.111
	$gyro_{std}$ 0.16	$gyro_{std}$,0.11	$gyro_{max}$ 0.085
	$gyro_{max}$ 0.123	$gyroUn_{mean}$ 0.1	$gyroUn_{mean}$ 0.082
	acc_{std} 0.085	$gyroUn_{max}$ 0.066	$gyro_{std}$ 0.074
(Car, Walking)	acc_{std} 0.239	acc_{std} 0.145	$linearA_{mean}$ 0.131
	$gyro_{mean}$ 0.155	$linearA_{mean}$ 0.128	acc_{std} 0.103
	acc_{max} 0.122	$gyro_{mean}$ 0.102	$gyro_{mean}$ 0.093
	$gyro_{std}$ 0.119	$gyroUn_{mean}$ 0.061	$gyroUn_{mean}$ 0.068
(Bus, Still)	acc_{std} 0.183	$linearA_{max}$ 0.107	$speed_{max}$ 0.115
	acc_{min} 0.134	acc_{std} 0.091	$linearA_{max}$ 0.096
	$sound_{mean}$ 0.119	$sound_{mean}$ 0.084	acc_{std} 0.086944
	$sound_{max}$ 0.104	$sound_{max}$ 0.072	$speed_{min}$ 0.077

	acc_{std} 0.251	$linearA_{mean}$ 0.118	$linearA_{mean}$ 0.136
(Car,	acc_{max} 0.128	$linearA_{min}$ 0.098	$linearA_{max}$ 0.131
Still)	acc_{min} 0.117	acc_{std} 0.083	acc_{std} 0.108
	$sound_{min}$ 0.073	acc_{max} 0.081	acc_{max} 0.062
	acc_{std} 0.244	acc_{std} 0.129	acc_{std} 0.117
(Still,	$sound_{min}$ 0.126	$sound_{min}$ 0.092	$speed_{max}$ 0.083
Train)	acc_{max} 0.107	$sound_{max}$,0.072	$speed_{min}$ 0.079
	$sound_{mean}$ 0.099	$sound_{mean}$ 0.070	$sound_{max}$ 0.062
	$gyro_{mean}$,0.14	$gyro_{mean}$,0.065861	$speed_{mean}$ 0.113
(Bus,	$gyro_{max}$ 0.113	$gyroU_{mean}$ 0.05	$speed_{min}$ 0.109637
Train)	acc_{mean} 0.102	$sound_{std}$ 0.048	$speed_{max}$ 0.106307
	$gyro_{std}$ 0.093	acc_{mean} 0.048	$gyro_{mean}$ 0.057
	$gyro_{max}$ 0.104	$sound_{std}$ 0.065	$speed_{max}$ 0.076
(Bus,	acc_{max} 0.099	$rotationV_{min}$ 0.057	$rotationV_{min}$ 0.066
Car)	$gyro_{mean}$ 0.097	acc_{max} 0.054	$speed_{min}$ 0.064
	$sound_{std}$ 0.096	$rotationV_{max}$ 0.051	$rotationV_{mean}$ 0.064
	acc_{std} 0.185	$linearA_{mean}$ 0.089	$speed_{min}$ 0.137
(Car,	acc_{min} 0.144	$linearA_{min}$ 0.076	$speed_{max}$ 0.124
Train)	$gyro_{mean}$,0.09	acc_{std} 0.076	$speed_{mean}$ 0.117
	acc_{max} 0.086	acc_{min} 0.056	acc_{std} 0.078

Table 5.6: Table show first four features for relevance in classifcation between target classes. Abbeviations used in Table are: *Acc* for accelerometer, *gyro* for gyroscope, *linearA* for linear acceleration, *rotationV* for rotation vector, *gyroU* for gyroscope uncalibrated. Suffixes *mean*, *max*, *min*, and *std* are used to identify features.

As we can see, when one of the target classes is walking, features relative to the accelerometer, gyroscope (uncalibrated and no), and linear acceleration are always the most relevant features. In Figures 5.5, 5.6, 5.7, and 5.8 they are shown the trends of features relative to the accelerometer, linear acceleration, gyroscope and gyroscope uncalibrated.

First of all, we can see that although accelerometer and linear acceleration features have different values, they exhibit the same ability to distinguish the walking activity from other. For example, the mean feature of linear acceleration and standard deviation of accelerometer have, for walking, values higher than the other without any overlapping.

Looking, instead, at gyroscope and gyroscope uncalibrated features, we can note that they have not only similar trend, but also similar values. Like accelerometer and linear acceleration, their values can be discriminant in walking detection. For example, mean feature for both sensors has a good grade of difference from other.

Consequently, a model the aim of which is to figure out if a user is walking can easily be based on these sensors. Moreover, considering that gyroscope is based on gyroscope uncalibrated and linear acceleration on acceleration only two of these for can be used.

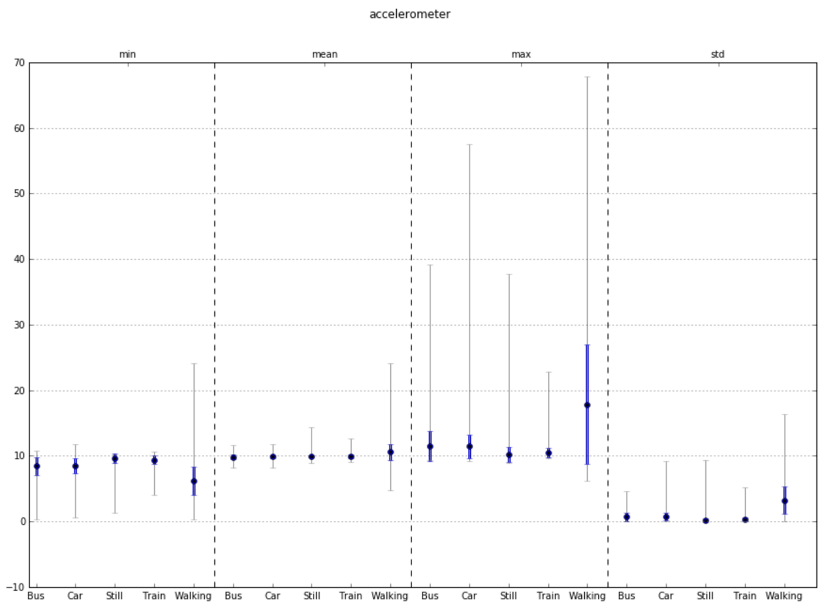


Figure 5.5: Accelerometer features (min, mean, max and standard deviation) respet to the target classes.

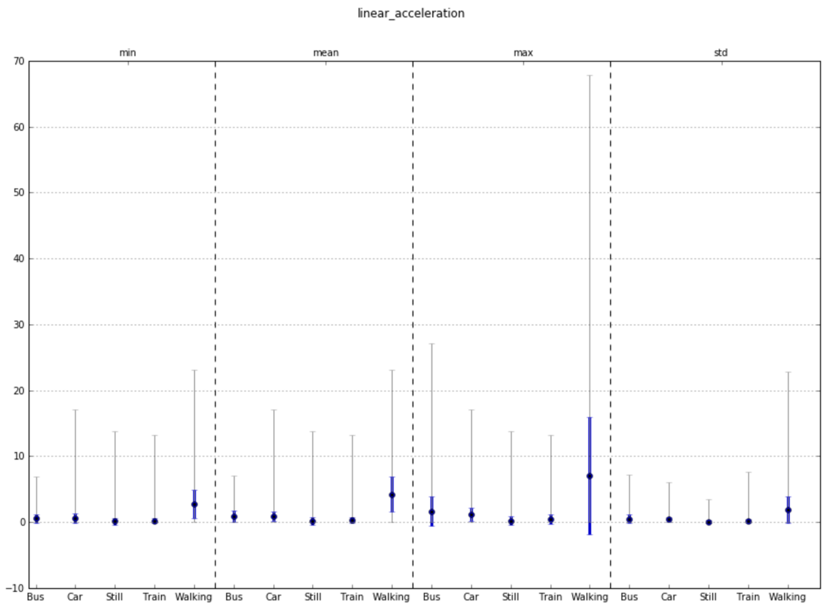


Figure 5.6: Linear acceleration features (min, mean, max and standard deviation) respet to the target classes.

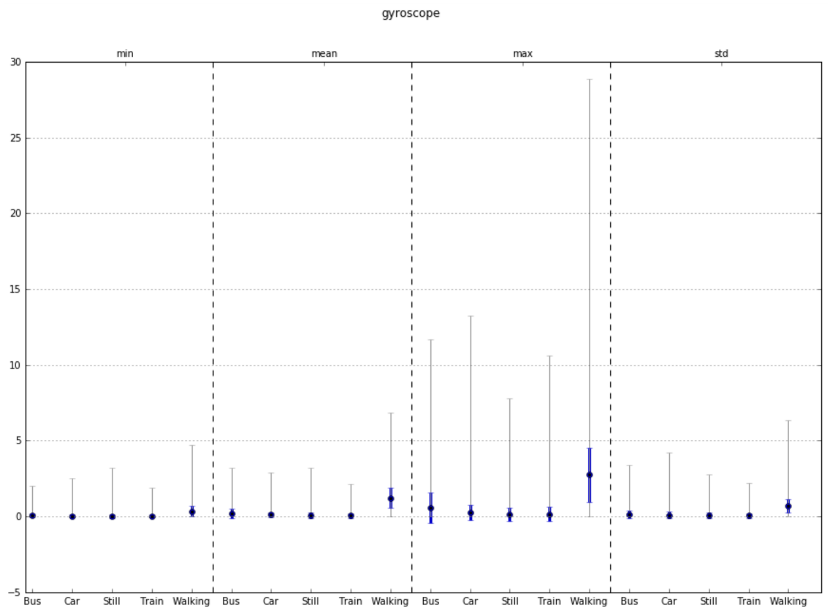


Figure 5.7: Gyroscope features (min, mean, max and standard deviation) respet to the target classes.

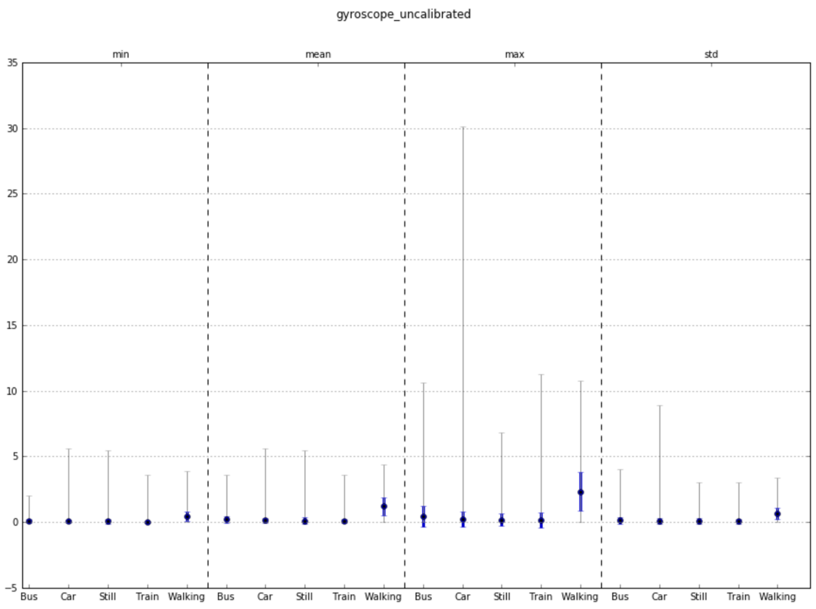


Figure 5.8: Gyroscope uncalibrated features (min, mean, max and standard deviation) respet to the target classes.

Instead, when still is one of the target classes, features relative to the accelerometer and linear acceleration remain relavant (see Figures 5.5, 5.6), but the model considers also sound, and benefit mostly of speed's features. Speed is always helpful to distinguish still from motorized transportation mode. Instead is not useful to distinguish walking from standing still, probably due to the fact that we don't impose rules to users during the experiments, so during walking activity maybe they stop in front of shops or at traffic lights. For detailed values of speed's feature look at Figure 5.10. It is clear from the figure that the standard deviation in the case of the speed is not a relevant feature.

Lastly, even in the case that the model tries to distinguish between motorized transport modes (car, train, and bus), speed's feature are relevant, mostly to detect if the user is on a bus or on a train. This two transportation mode have different speeds, especially when buses moves in urban enviroments. Different considerations have to be made for being on a car.

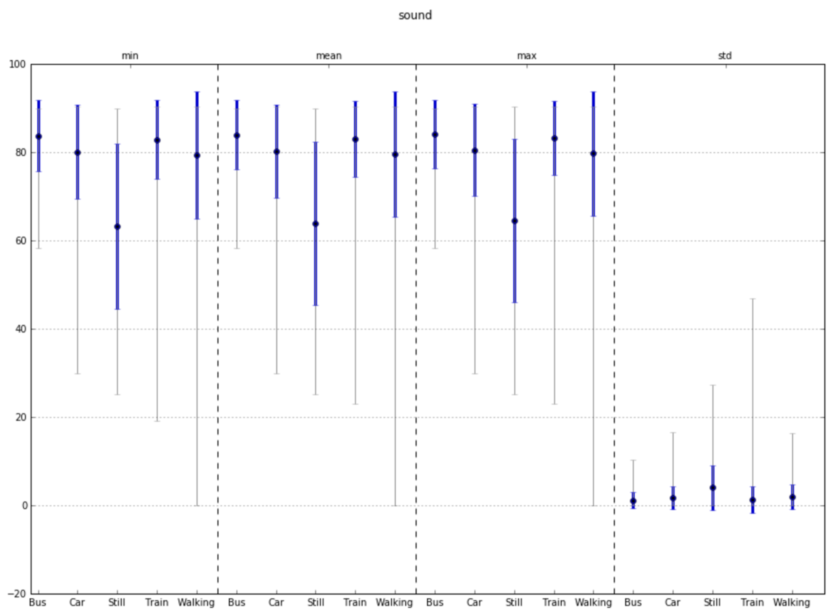


Figure 5.9: Sound features (min, mean, max and standard deviation) respet to the target classes.

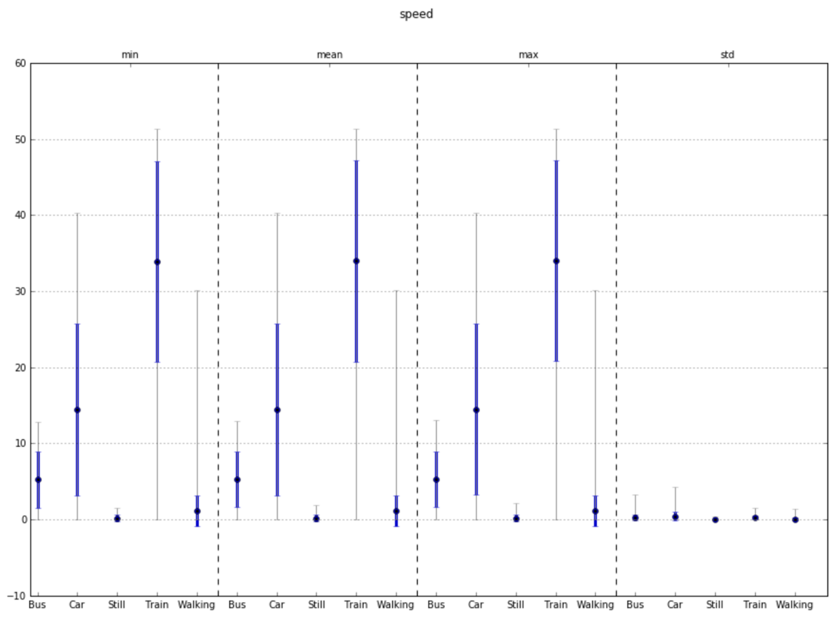


Figure 5.10: Speed features (min, mean, max and standard deviation) respet to the target classes.

Cars have great variability on speed, data were collected both in an urban environments, where speed is reduced from traffic congestion and traffic lights and on high-speed roads. Compared to the previous cases where few features are very relevant and the remaining have a very low relevance, in the case in which the model should distinguish between being on a car, being on a bus or being on a train, there are many features with medium relevance.

5.3 Google activity awareness

Google released their activity awareness API, which allows applications to register for activity recognition updates. Google detects the following transportation modes.

- Vehicle : the device is in a vehicle, such as a car.
- Bicycle : the device is on a bicycle.
- Foot : the device is on a user who is walking or running.
- Running : the device is on a user who is running.
- Still : the device is still (not moving).
- Tilting : the device angle relative to gravity changed significantly.
- Walking : the device is on a user who is walking.

In addition to these classes, google introduces *unknown* target class, mean that is unable to detect the current activity.

Activity detection call of Google activity awareness API returns two values: activity type, the activity that was detected and confidence values, a value from 0 to 100 indicating how likely it is that the user is performing this activity. Larger values indicate that it's likely that the detected activity is correct.

Although we know that we can not compare our result with those obtained from google, we consider interesting show and discuss the results obtained by google.

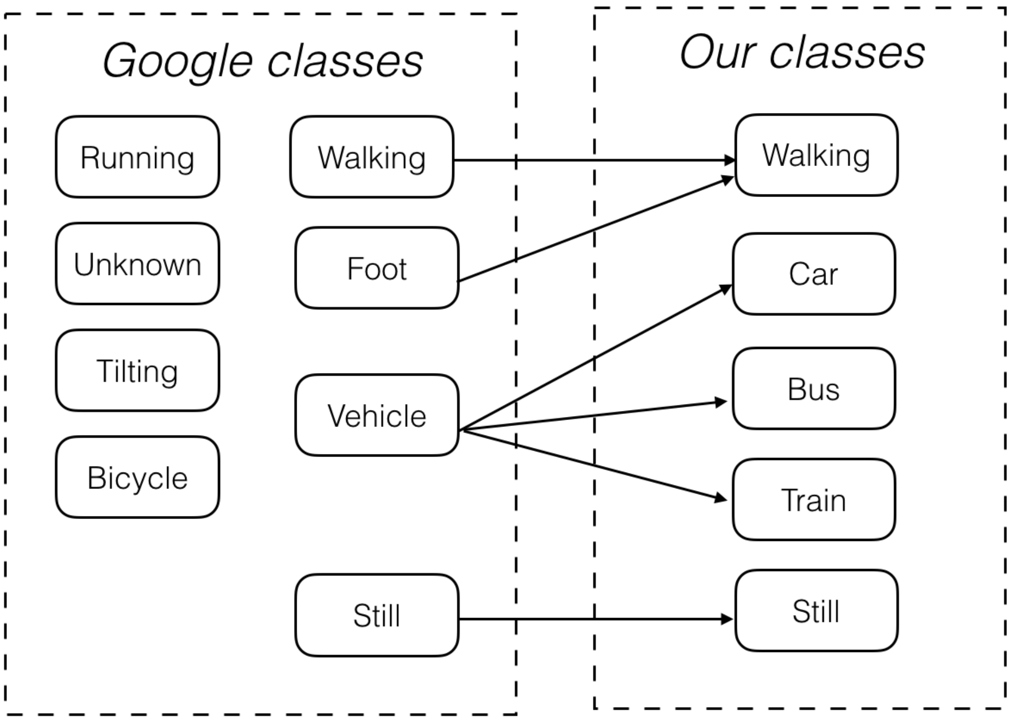


Figure 5.11: Mapping from google activity awerness API to our model classes.

First of all, we have to map google classes into our classes in order to be able to determinate if google classification result can be considered as correct. Target classes mapping is shown in Figure 5.11.

During our data collection, we register to activity awareness service, and we save google results as the other sensors results. When we reduce from raw data into five-second windows we report Google classification only if at least one is present in the interval, if we found more than one we choose that one with the greater confidence value.

Result dataset has only 698 examples over 22.904, about 3%, of windows with google classification. The accuracy, based on classes mapping 5.11, is 85,2%. Although this result is pretty good, the number of classified windows respect the total is very low.

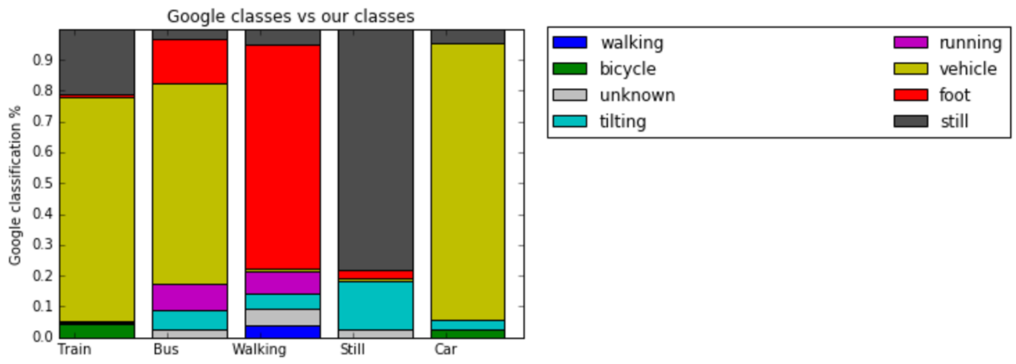


Figure 5.12: Google classification results on our dataset assuming that the classification once gave a result considers these valid until the next is return. Example with no Google classification result are excluded.

We then tried to make assumptions respect to the logic of classification result return. First, we try to assuming that the classification once gave a result considers these valid until the next is return. This hypothesis is reasonable, all the Android sensors work this way. Under this assumption, result dataset has 14.381 examples over 22.904, about 62,79%, with google classification, with an accuracy value equal to 79,25% (detailed in 5.12). Results, in terms of accuracy, seem to confirm our first hypothesis on classification return policy, the accuracy decreases slightly compared to the growth of the classified examples.

With the last assumption, more than 8.000 examples don't have a result of Google classification. These examples correspond to cases where users start a recording and google did not return anything. Our imputation policy, impute the last returned value, but we have not defined any rules for the cases that Google has not yet returned a result. Adding a rule for this last case, that imputes the unknown class we obtain a dataset with all example with google result, but with only 49,76% of accuracy (detailed in 5.13). As we expect, using the same imputing rules only on a test set don't change significantly the accuracy value (49,86%).

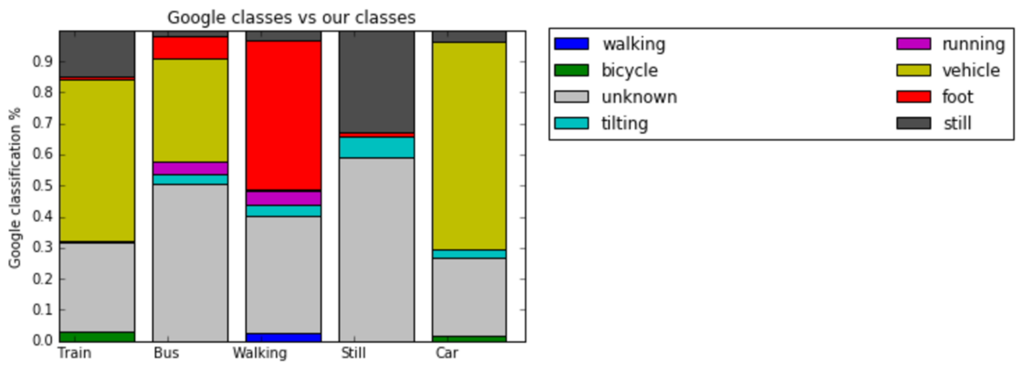


Figure 5.13: Google classification results on our complete dataset assuming that the classification once gave a result considers these valid until the next is return, and impute unknown class to example where we have no information on previous examples.

It should be considered that in the absence of a complete documentation, all these results over activity awareness API performance on our dataset, except the first are based on assumptions.

Chapter 6

Conclusion

In this thesis, we studied the problem of transportation mode recognition using sensors data from cellphone. Identification of transportation mode is extremely important in context-aware applications.

6.1 Contributions

Although many works have been published in the last decade, to the best of our knowledge they is the use *ad hoc* datasets. They collect data only from few sensors, often they have a lack in user base and in many cases the collection processes use non-real conditions. This shortcoming in datasets has two main consequences: *(1)* results of different researchers can't be compared to find the best strategy to solve a specific task, *(2)* we can not assume that the proposed strategies are applicable to real contexts, obtaining the same results.

Each work in the context of the transportation mode detection has built its own dataset and found the best strategy for the recognition based on it. Often these datasets were obtained through users in conditions that are not real. For example, many studies were conducted with specific devices placed at different locations on the human body, while other studies imposed a specific route to the users, others have collected the data under more realistic conditions, but from a very limited set of users.

One of the main contributions of our work is to have built a new dataset from thirteen users, with different gender, age, and occupation. This data collection occurs under real-world conditions; users during the experiments are free to use their devices, carry, and move it as they want. This process was controlled by an Android application running on the user's phone as they performed five different activities such as walking, driving a car, being on a train, on a bus or standing still. At the end, we gathered more than 31 hours of annotated data for five different transportation modes.

Our dataset is not limited to few sensors, we decided to monitor the activity of all phone sensors, in order to permit the use of dataset in future studies. This will permit a comparison with results obtained by us.

Another contribution of this thesis is the use of sensors that have been previously ignored for this specific task. Finding that also other sensors can help to discriminate between different transportation activities, so there is still space for improvement and to overcome some of the outstanding problems associated with GPS use.

We initially limited dataset to features based on three base sensors (D_{first}), with a low consumption of battery and a high presence in mobile phones: accelerometer, gyroscope, and sound. Then, we have extended the set of considered sensors to all monitored sensors apart from the speed, for battery reasons (D_{second}). Lastely we added speed data (D_{third}).

We apply machine learning techniques to these three datasets in order to obtain three different comparable models. We test four classification algorithms on all the datasets: Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Neural Network (NN). Among the four classification algorithms considered, Random Forest is the classification model that outperforms all the others. Results show that model able to better infer transportation modes is the one that considers all sensors from the dataset. With random forest overall accuracy is 88% for D_{first}, 93% for D_{second} and 96% for D_{third}.

Considering that sensors data are not obtained without paying a price in terms of resource consumption, it is clear the need of a trade-off between the consumption of resources and the accuracy of the model. This equilibrium changes under different applications scenarios. An important application's characteristic is what activities we want to detect.

We also realized that accuracy varies significantly for different transportation modes. Some activities are difficult to distinguish, for example being on a bus. In these cases adding more sensors data can be very useful for the detection capacity. On the other hand, some activities are easily identificable, as walking. In these cases there is no big difference between training the model with D_{first}, D_{second} or D_{third}. These kind of results can be relevant because many applications need to identify only a small set of activities.

Therefore, we observed how accuracy changes on a restricted number of target classes. Combining any two activities, over the five considered, we trained new models dased on the three datasets. We observed that some activities are easily identified while others still remain difficult to identify. For example walking and standing still are easily detected even on the first dataset D_{first}, reaching an accuracy between 96,32% and 97,57%. On the other hand, motorized transportation, that exhibits similar patterns in the data, reaches the same data accuracy (between 90.24% and 94,43%. However, this distance is reduced when using the complete dataset D_{third}.

These results suggest a careful choice of sensors with respect to the classes to be recognized.

6.2 Future research

The current study has a lot of space for improvement. For instance, it would be interesting to expand the dataset, incorporating more transportation modes (e.g., metro, bicycle . . .) or collecting data from more users, and from different geographic areas.

Furthermore, future works can include a study of different, potentially more sophisticated strategies to transform raw data in examples. We suppose that choosing different lengths or overlapping windows can have different effects on the final results. It would be also interesting to investigate how classification accuracy changes by adding one sensor at a time or extracting different features. Both these tests can be made for specific set of activities offering a custom solution for specific tasks. Another interesting study concerns the consumption of different solutions resources.

The last extension that we propose is to apply different approaches to examine the data as a time sequence. This is because we used consecutive windows as separate examples, ignoring the temporal relationship that exists between them.

Bibliography

[1] Abowd, Gregory D., et al. "Towards a better understanding of context and context-awareness." *International Symposium on Handheld and Ubiquitous Computing*. Springer Berlin Heidelberg, 1999.

[2] Perera, Charith, et al. "Context aware computing for the internet of things: A survey." *IEEE Communications Surveys & Tutorials* 16.1 (2014): 414-454.

[3] Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." *Computer networks* 54.15 (2010): 2787-2805.

[4] T. M. Mitchell, Machine Learning, McGraw Hill (2005)

[5] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." *Machine learning* 20.3 (1995): 273-297.

[6] Allwein, Erin L., Robert E. Schapire, and Yoram Singer. "Reducing multiclass to binary: A unifying approach for margin classifiers." Journal of machine learning research 1.Dec (2000): 113-141.

[7] Anand, Rangachari, et al. "Efficient classification for multiclass problems using modular neural networks." *IEEE Transactions on Neural Networks 6.1* (1995): 117-124.

[8] Clark, Peter, and Robin Boswell. "Rule induction with CN2: Some recent improvements." *European Working Session on Learning*. Springer Berlin Heidelberg, 1991.

[9] Platt, John C., Nello Cristianini, and John Shawe-Taylor. "Large margin DAGs for multiclass classification." *Proceedings of the 12th International Conference on Neural Information Processing Systems.* MIT press, 1999.

[10] Abe, Shigeo. "Analysis of multiclass support vector machines." *Thyroid 21.3* (2003): 3772.

[11] Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.

[12] Bao, Ling, and Stephen S. Intille. "Activity recognition from user-annotated acceleration data." *International Conference on Pervasive Computing.* Springer Berlin Heidelberg, 2004.

[13] Anderson, Ian, and Henk Muller. "Context awareness via gsm signal strength fluctuation". na, 2006.

[14] Sohn, Timothy, et al. "Mobility detection using everyday gsm traces." *International Conference on Ubiquitous Computing.* Springer Berlin Heidelberg, 2006.

[15] Mun, M., et al. "Parsimonious mobility classification using GSM and WiFi traces." *Proceedings of the Fifth Workshop on Embedded Networked Sensors (HotEmNets).* 2008.

[16] Kwapisz, Jennifer R., Gary M. Weiss, and Samuel A. Moore. "Activity recognition using cell phone accelerometers." *ACM SigKDD Explorations Newsletter* 12.2 (2011): 74-82.

[17] Bedogni, Luca, Marco Di Felice, and Luciano Bononi. "By train or by car? Detecting the user's motion type through smartphone sensors data." *Wireless Days (WD), 2012 IFIP.* IEEE, 2012.

[18] He, Zhenyu, and Lianwen Jin. "Activity recognition from acceleration data based on discrete cosine transform and SVM." Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on. IEEE, 2009.

[19] Mohan, Prashanth, Venkata N. Padmanabhan, and Ramachandran Ramjee. “Nericell: rich monitoring of road and traffic conditions using mobile smartphones.” *Proceedings of the 6th ACM conference on Embedded network sensor systems.* ACM, 2008.

[20] Bhoraskar, Ravi, et al. “Wolverine: Traffic and road condition estimation using smartphone sensors.” *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on.* IEEE, 2012.

[21] Zheng, Yu, et al. “Understanding transportation modes based on GPS data for web applications.” *ACM Transactions on the Web (TWEB)* 4.1 (2010).

[22] Hemminki, Samuli, Petteri Nurmi, and Sasu Tarkoma. “Accelerometer-based transportation mode detection on smartphones.” *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems.* ACM, 2013.

[23] Reddy, Sasank, et al. “Using mobile phones to determine transportation modes.” *ACM Transactions on Sensor Networks (TOSN)*6.2 (2010): 13.

[24] Bedogni, Luca, Marco Di Felice, and Luciano Bononi. “Context-aware Android applications through transportation mode detection techniques.” *Wireless Communications and Mobile Computing* 16.16 (2016): 2523-2541.

[25] Sun, Lin, et al. ”Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations.” *International Conference on Ubiquitous Intelligence and Computing.* Springer Berlin Heidelberg, 2010.

[26] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. ”Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors).” *The annals of statistics* 28.2 (2000): 337-407.