Online travel mode detection on a smartphone

Final Project | Master Applied Mathematics | Discrete Mathematics and Mathematical Programming

H.A. Buist
October 2020

Supervisors
Prof. Dr. M.J. Uetz
University of Twente

Dr. Ir. J.W. Koolwaaij
Mobidot

Graduation Committee
Prof. Dr. M.J. Uetz
University of Twente

Prof. Dr. A.J. Schmidt-Hieber
University of Twente

Dr. Ir. J.W. Koolwaaij
Mobidot

Abstract

A classification model for travel mode detection is developed. The input for the model is collected with a triaxial accelerometer integrated in a smartphone. The classification model is based on logistic regression. Four extensions based on logic reasoning are suggested. The developed model is tested using five test cases. Accelerometer based classification does a good job for non-motorized travel modes, such as walking and cycling. However, motorized travel modes are more difficult to classify using only accelerometer data. Using the proposed algorithmic approach, an overall accuracy of 50.0% could be obtained.

Preface

Years ago, when I started elementary school, it felt like school would never end. Nothing could be further from the truth, because this report is my graduation thesis from my masters in applied mathematics. I performed my final project in collaboration with Mobidot, a company in Enschede. Due to Covid-19, I did the main part of my final project at home, instead of at the office of Mobidot. This made it sometimes hard to concentrate and keep going, however, it makes me even more proud of this thesis and the results I achieved.

The report describes a classification model, created to classify travel modes. The input for the model solely consists of accelerometer data measured with a smartphone. Mobidot requested the classification model, since they felt they lacked a tool to determine transportation modes in a uniform way.

I would like to thank both my supervisors, Johan Koolwaaij from Mobidot and Marc Uetz from the University of Twente for their excellent guidance and answers to all of my questions. They where always available if I had any questions or did not know how to deal with something. Furthermore I want to give special thanks to my husband Bram de Jonge how remained patient with me during my whole final project, even during the times we worked together at our very small dining table.

I hope you enjoy your reading.

Hillianne de Jonge-Buist

Enschede, October 1, 2020

Contents

Chapter 1

Introduction

Since 1978, the *Centraal Bureau voor de Statistiek*, a Dutch governmental organisation that collects data of the Dutch society, researches travel behaviour inside the Netherlands every year. This research is nowadays called 'Onderweg in Nederland' [1]. The research is performed by conducting an online survey among Dutch people. Participants have to answer questions about their travel behaviour, and the results of the survey are used to investigate travel behaviour of all Dutch people. However, the survey results are highly influenced by how they are filled in by the participants. This most likely results in a lower quality of the research results.

It is hence clear that there are organisations that are interested in the travel behaviour of individuals. However, conducting a survey is not the best way to obtain results on travel behaviour. A much better way would be to give each participant a device that measures travel behaviour and automatically concludes which travel mode is used for how long. Designing a device is not necessary, since almost all Dutch people already carry such a device with them all day, namely the smartphone. If it would be possible to measure travel behaviour and conclude which travel modes are being used by installing an app on a smartphone, the whole survey would not be needed anymore. The results for the 'Onderweg in Nederland' research, could simply be drawn from the smartphone data.

However, before this can be the case, apart from all the issues regarding privacy, the app should be realised, and it should include a method to determine travel modes.

The research described in this report is conducted on behalf of Mobidot, another company interested in the use of smartphones when it comes to collecting and analysing data about travel behaviour. They would like to have a model that can use smartphone data to determine in which ways someone has travelled. The above is the context under which this research was conducted.

1.1 The use of smartphones in detecting travel modes

Since the development of the smartphone, and even long before that, researchers have been busy researching the determination of travel modes using certain devices, such as an accelerometer. This is a device that measures acceleration [2]. Nowadays, smartphones almost always include an accelerometer, as well as other sensors that can be used for travel mode detection [3]. Since almost everyone owns a smartphone nowadays, almost all recent studies performed in the field of travel mode detection make use of this. All sorts of sensors included in a smartphone are used to collect data and analyse travel behaviour.

A lot of studies use both accelerometer data and GPS signals to detect transportation modes [4–6]. Some studies only use data collected with an accelerometer [7, 8], whereas others use a variety of sensors, such as the gravity sensor, magnetometer, barometer and the light sensor [9, 10].

In most cases, the GPS signal is used to determine velocity, whereas the accelerometer is used to observe changes in direction and find patterns in movements. The gravity sensor and magnetometer are often used to remove the influence of gravity from the accelerometer data, which unfortunately is needed since an accelerometer not only measures motion, but is also influenced by gravitational forces [11, 12]. The magnetometer, barometer and light sensor can amongst other things be used to sense whether the smartphone is in an indoor or outdoor environment [12]. For instance, the light sensor determines the amount of ambient light. When walking or cycling, this amount can be higher than when travelling with a motorised vehicle.

Online and offline classification
There are two approaches possible for performing classification. It can be done *offline*, after all the data is collected, or *online*, while the data is collected. Hence, unlike offline classification, online classification is done in real time. The big advantage of offline classification is that more data is available. However, online classification can be used for updating the user on the travel mode, while the trip has not been finished yet. Also, when performing classification on a smartphone, not all data has to be kept until the end of data collection but can be removed when classification is done.

Machine learning and heuristics
A lot of studies are done on detecting travel modes, and a lot of different approaches have been tried. Almost all studies use some machine learning technique for classification. Often used machine learning algorithms are Support Vector Machines [11], Random Forests [5, 8, 10], Bayesian Networks [13, 14], decision trees and Hidden Markov Models [6], Naive Bayes [15], and Neural Networks [16].

Some studies add a heuristic in addition to the machine learning algorithm or developed their own classification algorithm. An interesting heuristic is used by [17]. First walking is filtered out, based on the average acceleration. Then, it is decided for the intermediate parts what kind of travel mode is used. This approach is based on the assumption that every activity starts and ends with a bit of walking. Also, it is assumed that changing between two travel modes always contains a little bit of walking. Another interesting approach is suggested by [9]. In that paper, a model is described that takes two steps in distinguishing between different travel modes. The first step is to make a selection between wheeler and non-wheeler travel modes. Second, a differentiation between wheeler modes must be made. In the study performed by [18], a algorithm is developed which is called MCODE and is used for classification. No machine learning technique is used.

Features
Most travel mode classification algorithms are based on features derived from a trip. Those features are calculated based on a short period of collected data. Most features are time- or frequency-based [19]. Examples of time-based features are: *mean*, *minimum*, *maximum* and *variance*, whereas frequency-based features are for example: *energy*, *entropy* and *time between peaks*. Frequency-based features are often computed using for example the Fast Fourier Transform [20, 21].

1.2 Research question and scope

Currently, Mobidot does not have a travel mode classification algorithm of their own. Instead, the travel mode detection integrated in smartphones is used. This causes the detection to be dependent on the smartphone brand and type. To overcome this issue, Mobidot would like to have a classification algorithm of their own. The question answered in this report is therefore:

Is it possible to use accelerometer data for travel mode classification?

The collection of data for this research is done with an app provided by Mobidot. Therefore, this report will not focus on how data collection is done. Preprocessing the data will be addressed, as well as calculating features and building a classification model. The classification model must be able to distinguish between travel modes *still, walking, cycling, bus, train* and *car*. Hence, other travel modes are beyond the scope of this research.

The classification model is developed on a laptop, hence running and testing the model on a smartphone are not part of this report.

1.3 Outline

The remainder of this report is structured as follows. In Chapter 2 first some terminology is discussed. Furthermore, the details of the problem addressed by this report are described and the relevance of the research is described. Chapters 3, 4 and 5 contain the description of the developed classification model. In Chapter 3, some necessary preprocessing is described. This preprocessing is needed to make the input data suited to work with in a classification model. Using the preprocessed data, features can be extracted. This process is described in Chapter 4. Multiple classification algorithms, developed for classifying travel modes, are described in Chapter 5. Chapter 6 and 7 respectively describe the experiment setup and the results obtained with the developed classification models. Chapter 7 also establishes values for parameters used in the classification models. Finally, conclusions and recommendations for further research and model development are given in Chapter 8.

Chapter 2

Problem description

2.1 Glossary

Modality	A way of traveling, like walking, cycling, public transport and so on.
Travel mode	Similar to *modality*.
Trip	A journey that lasts from the moment of departure until arrival elsewhere

2.2 The problem explained

Founded in 2013, Mobidot is a company that analyses travel behaviour on behalf of clients, such as governments, service providers, and employers [22]. The goal of analysing this data, is to create insight in travel behaviour and eventually being able to influence travel behaviour in a personal way.

Part of the work of Mobidot is analysing trips taken by users that have installed a specific app on their smartphone. Through this app, data such as GPS locations, is collected in order to estimate the participants position and travel path. If also travel modes would be known, those can for instance be used to improve the position and travel path estimates.

Most modern smartphones already contain a chip that translates the 3D movement from the integrated accelerometer into activities, like still, walking, cycling and in-vehicle. The Mobidot sensing library already uses these deduced activities, for example as priors for the trip modality recognition. But there are some major drawbacks to this, like the heterogeneity between different phone manufacturers, limited set of recognised activities, cycling detection is usually of lower quality, and Mobidot does not have a handle to alter or improve the algorithms used.

Therefore, Mobidot would like to develop a new activity detection algorithm that is able to classify travel modes, and can be executed on every smartphone. Using the same algorithm on every smartphone causes the travel mode prediction to be independent of the smartphone type used, and is therefore more useful for Mobidot. Also, using their own activity detection algorithm allows them to alter or improve the activity detection.

The classification algorithm, as the activity detection algorithm will be called in the remainder of this report, must meet the following requirementes, as determined by Mobidot.

- **Accelerometer based:** The input data for the classification algorithm should *only* consist of accelerometer data, collected with a smartphone. Since all smartphone accelerometers are triaxial, this implies input data of the form (t, a_x, a_y, a_z), with t the time of measurement, and a_x, a_y and a_z the measured acceleration with respect to the x-, y- and z-axis of the smartphone, respectively. The data will be sampled at 5Hz, hence 5 measurements per second.

- **Battery-efficient:** The classification algorithm is executed on a smartphone. In order not to drain the battery, the algorithm must run in a battery-efficient way. This implies developing a classification algorithm that does not require much computing power.
- **Expandable:** In future, the classification algorithm must be expandable, in the sense that new activities can be easily added by Mobidot.
- **Classify during trips only:** Only accelerometer data collected during a trip should be classified. Hence, activities such as *'watching a movie while sitting on the couch'*, which are typically done while not travelling, are not included in the set of activities that have to be recognised.
- **Online classification:** The classification must be done online, which means that it must be done while data is collected. This also means that classification has to be done on the smartphone itself.

Furthermore, the set of travel modes that must be recognised with the classification algorithm consists of *still, walking, cycling, bus, train* and *car*.

The long term goal of the classification algorithm, together with other algorithms for trip analysis by Mobidot, will be that that it leads to an activity story line per day, with a focus on personal trips. As an example: "Today I left the office at 17:44, took the elevator to street level, and walked to the bus stop. I had to wait for 3 minutes until the bus arrived, and I had to stand in the bus until stop X before a seat became available. At station S, I walked to subway platform P, where train T just arrived to bring me home. In the subway I tried to sleep a bit." With this activity story line in mind, the subsequent trip analysis algorithms can be much more targeted and precise, and maybe also be more efficient with better performance. Creating this story line is beyond the scope of this report, which will focus on developing the classification algorithm itself. In that sense, the work done in the report can be seen as starting point for a later tool for generating such an activity story line.

2.3 Relevance of our research

As a conclusion of the previous chapter, we can state that it is highly relevant to Mobidot to have its own travel mode detection algorithm based on accelerometer data. Taking their wishes and requirements into account, no suited model could be found in literature. Hence, a new model had to be developed. Below I describe in more detail why a suited model did not exist.

As described in Chapter 1, there is a difference between online and offline classification. Many studies conducted on travel mode detection focus on offline classification, and therefore have no absolute need to be lean and battery-efficient. However, since one of the requirements for the developed classification model is the possibility to perform online classification, we do need a lean and battery-efficient way of classifying. Using *logistic regression* turned out to be suited for our purposes. As far as we know, no study has used this technique before in a travel mode classification model.

Most studies known from the literature use multiple sensors, including the accelerometer. For our model, however, the input may only consist of accelerometer data. This, in combination with what is mentioned above, makes that our research is unique and not performed before.

Chapter 3

Preprocessing

This chapter will discuss in detail how data is collected and preprocessed to be used as input for the classification algorithm described later in Chapter 5.

3.1 Collecting data

The accelerometer included in a smartphone is a so-called triaxial accelerometer, which means that acceleration is measured in three directions. In Figure 3.1 those directions are visualised. For every phone, the three directions are equal, however the names of the axes may differ per smartphone type and brand. However, it always holds that accelerometer data is of the form (t, a_x, a_y, a_z), where t is the time at which acceleration is measured, and a_x, a_y and a_z are the measured accelerations in the direction of the x-, y- and z-axis, respectively.

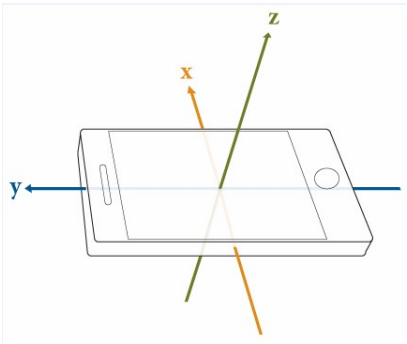


Figure 3.1: Schematic representation of the direction in which acceleration is measured with respect to a smartphone, adopted from [23].

At least every 0.2 seconds, which is 5Hz, the smartphone measures the acceleration in all three directions. However, it can be the case that the phone is busy with other things, and therefore skips a measurement once in a while. If not much is asked from the phone, for instance during nighttime, it may also be that many more measurements are taken, up to 200 Hz, which means 200 measurements per second.

The classification is done online, which means that data collection and classification are done at the same time. To manage this, collected data is stored into windows. Every time s seconds have passed, a new window will be created and filled with new accelerometer measurements. A possibility is to use overlapping windows.

Hence, every s seconds, a new window w is filled with collected data. This data is then stored in matrix A_w, which looks like

$$A_w = \begin{pmatrix} t_1 & a_{x_1} & a_{y_1} & a_{z_1} \\ t_2 & a_{x_2} & a_{y_2} & a_{z_2} \\ \vdots & \vdots & \vdots & \vdots \\ t_{n_w} & a_{x_{n_w}} & a_{y_{n_w}} & a_{z_{n_w}} \end{pmatrix},$$

where n_w is the number of measurements in window w.

When choosing a value for s, two things must be taken into account. First, if s is chosen too small, there will be too few data points in the window to get an accurate classification. Second, if s is chosen to large, activities that last only a short time will be missed or distorted when classifying.

While collecting data, one of the following things can happen:

- It can occur that for two time indicators $t_i, t_j \in A_w, i \neq j$ it holds that $t_i = t_j$, while $(a_{x_i}, a_{y_i}, a_{z_i}) \neq (a_{x_j}, a_{y_j}, a_{z_j})$. Hence, there are two different measurements with the same time point.
- Despite the fact that the sampling rate is set to 5Hz, it can occur that a window A_w is empty or has only a few data points in it.

The reason why those things happen is not clear. In both cases, window A_w can not be used for classification and the window must be classified the same as the previous window.

3.2 Changing the direction of the data with known gravitational acceleration

A described in the previous section, acceleration is collected in three directions. However, due to the fact that the smartphone is not held into the same position all day, the directions in which acceleration is collected, are not consistent. By this, we mean that the forward/backward acceleration, the sideways acceleration and the upwards/downwards acceleration with respect to the earth are not always equal to the acceleration measured in the x-, y- and z-direction with respect to the smartphone, respectively. Consider for example the fact that the smartphone is very likely to be in a different position while cycling than when it is used to navigate while driving.

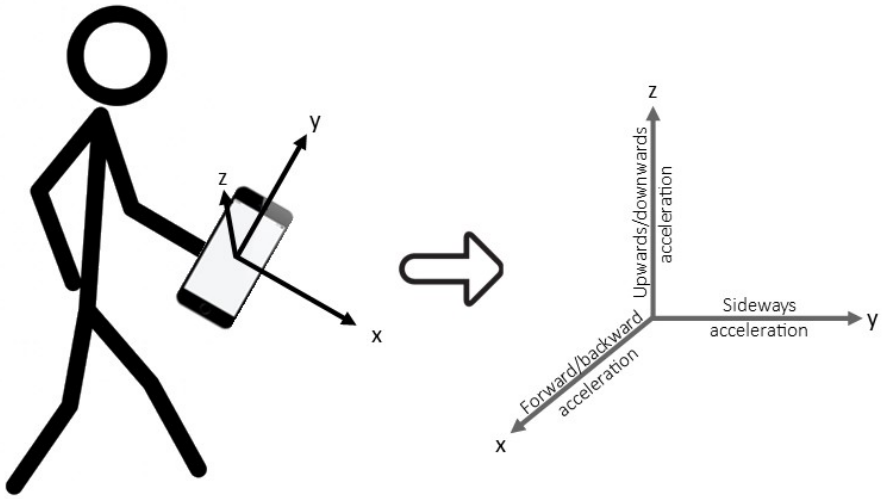


Figure 3.2: Real and desired direction of accelerometer measurements.

To make the data consistent, all measured data should be mapped to the coordinate system relative to the surface to the earth as shown on the right in Figure 3.2. Hence, the forward/backward acceleration must be mapped to the x-axis, the sideways acceleration to the y-axis and the upwards/downwards acceleration to the z-axis. We will refer to those desired axes as the axes of the earth, since we are interested in the orientation of the accelerometer measurement with respect to the earth instead of the smartphone.

If the data would not be mapped onto those axes, it would not be possible to use the separate acceleration measurements a_x, a_y and a_z for classification, because the measured acceleration cannot be

related to a specific direction. We then would only be able to use the magnitude of the acceleration, which does not tell anything about the direction of the movement that caused the acceleration.

To preprocess the data, we use the fact that an accelerometer is affected by both motion and gravity. This causes that the data collected with the accelerometer contains two types of acceleration, which have been merged into one output. The first type of acceleration is gravitational acceleration, caused by gravity. The direction of the gravitational acceleration is always orthogonal and points away from the surface of the earth [24]. The second type of acceleration is called linear acceleration and is caused by movement of the accelerometer [25].

Figure 3.3 shows the process of mapping accelerometer data onto the axes of the earth. For the sake of simplicity, the z-direction has been omitted in the pictures. In order to project the data onto the axes of the earth, the direction of the gravitational acceleration must be known.

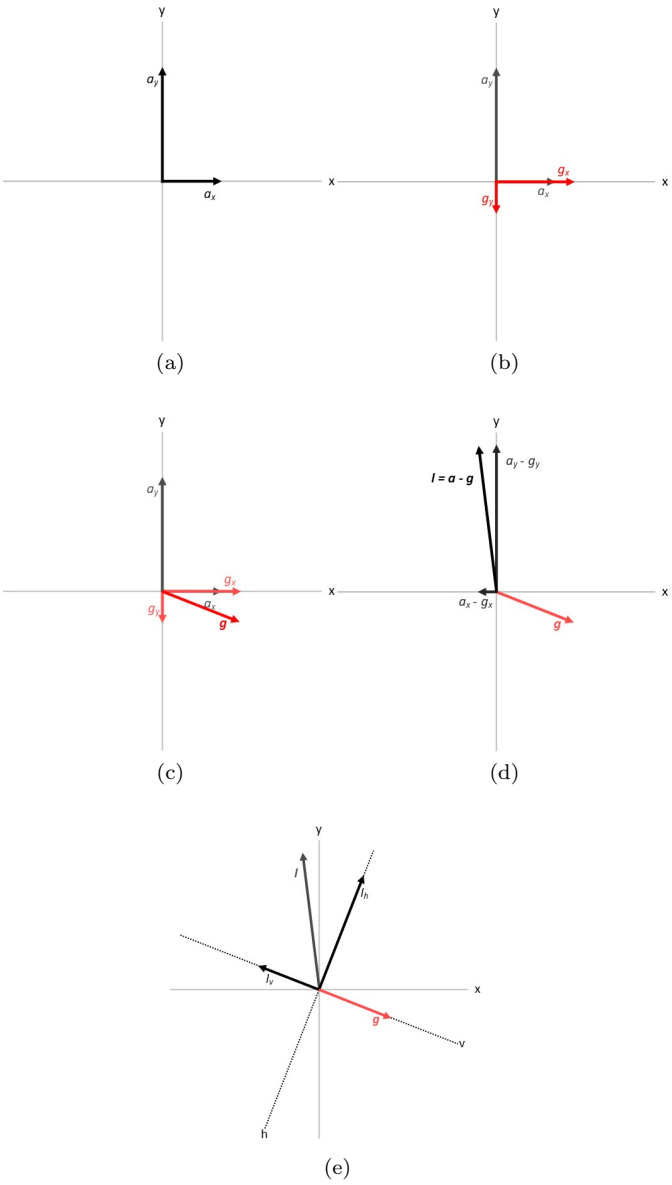


Figure 3.3: Projecting accelerometer data onto the axes of the earth, while removing gravitational acceleration

In the following, let us assume for the time being that the gravitational acceleration $\boldsymbol{g} = (g_x, g_y, g_z)$ relative to measured acceleration $\boldsymbol{a} = (a_x, a_y, a_z)$ would be known. Then $\boldsymbol{a}$ and $\boldsymbol{g}$ can be depicted as in Figures 3.3a and 3.3b. Both the acceleration $\boldsymbol{a}$ and the gravitational acceleration $\boldsymbol{g}$ are measured in three directions, x, y and z. Those directions are not necessarily equal to the axes of the earth. The first step in changing the axes of the measured acceleration to the axes of the earth, is finding the direction of the gravitational acceleration $\boldsymbol{g}$, with respect to the axes of the acceleration measurement, as shown in Figure 3.3c. Also, the gravitational acceleration $\boldsymbol{g}$ must be subtracted from the total acceleration $\boldsymbol{a}$, to find the linear acceleration $\boldsymbol{l} = (l_x, l_y, l_z) = \boldsymbol{a} - \boldsymbol{g} = (a_x - g_x, a_y - g_y, a_z - g_z)$, depicted in Figure 3.3d.

The direction of the gravitational acceleration $\boldsymbol{g}$ is always perpendicular to and away from the surface of the earth. Hence, the direction of the gravitational acceleration corresponds to the axis to which the vertical acceleration should be mapped. Therefore, we know that the vector projection of linear acceleration $\boldsymbol{l}$ to the gravitational acceleration represents the part of the linear acceleration that is perpendicular to the surface of the earth, which will be called $\boldsymbol{l_v}$.

However, since we are interested in the length of vertical linear acceleration $\boldsymbol{l_v}$, it suffices to take

the dot product of the linear acceleration $\boldsymbol{l}$ and the gravitational acceleration $\boldsymbol{g}$ instead of the vector projection, see Equation (3.1). However, the vector projection of $\boldsymbol{l}$ onto $\boldsymbol{g}$ is needed to find the part of the linear acceleration that is parallel to the surface of the earth, which will be called horizontal linear acceleration. When subtracting this vector projection from the total linear acceleration $\boldsymbol{l}$, the remainder is equal to the horizontal linear acceleration, $\boldsymbol{l_h}$.

Gravity could be used to find the direction of the upwards acceleration, l_v. However, no such thing exists that can be used to find the direction of the sideways or forwards acceleration. Therefore, the length of the horizontal linear acceleration is used. The final horizontal linear acceleration l_h is determined with the aid of Equation (3.2).

$$l_v = \boldsymbol{l} \cdot \boldsymbol{g} \tag{3.1}$$

$$l_h = \|\boldsymbol{l} - \frac{\boldsymbol{l_v}}{\boldsymbol{g} \cdot \boldsymbol{g}} \cdot \boldsymbol{g}\|_2 \tag{3.2}$$

After transforming the measured acceleration $\boldsymbol{a}$ to l_v and l_h, the mapping to the axes of the earth is completed, see Figure 3.3e. The data can now be handled with respect to the axes of the earth and no longer depends on the direction of the smartphone.

It is important to note that some phones are able to remove gravity from the signal themselves. However, the way this is done is not always trustworthy and accurate. Also, older phones are not able to remove gravity from the accelerometer measurement. Therefore, we chose to use the raw accelerometer data for all smartphones, and remove gravity for all smartphones on the same way. This makes our results more unambiguously.

3.3 Estimating gravitational acceleration

The result of the procedure described above heavily depends on the assumption that the gravitational acceleration $\boldsymbol{g}$ relative to $\boldsymbol{a}$ is known. This is not the case, and therefore the estimation of the gravitational acceleration should be done as accurately as possible.

Since only accelerometer data is used, the gravitational acceleration must be estimated with the aid of the data that we have. The study of [7] proposes an algorithm for estimating the gravitational acceleration, based on a more basic approach developed in an earlier study done by Mizell [25]. Both studies are based on the fact that the magnitude of the gravitational acceleration is always equal, namely approximately 9.81 m/s^2. Mizell suggests to obtain an estimate of the gravitational acceleration with respect to each axis by taking the average of the accelerometer measurements in the window on that axis. For instance, the gravitational acceleration in the direction of the x-axis is calculated as $g_x = \left(a_{x_1} + \dots a_{x_{n_w}}\right)/n_w$, with $a_{x_1}, \dots a_{x_{n_w}} \in A_w$ and n_w the number of rows in A_w.

The study done by Heminki [7] notes that this approach contains two major drawbacks. The first problem occurs if a window contains sustained acceleration, which is a constant acceleration caused by motion that lasts several seconds. When using the method of Mizell, the sustained acceleration will influence the gravity estimate, making it deviate from the real gravitational acceleration. Therefore, both the gravity estimate and the linear acceleration are not correctly computed in case of sustained acceleration. Second, sudden changes in orientation of the device are not taken into account and will result into a wrong gravitational acceleration estimate, in case such a change takes place.

To avoid the above disadvantages, we use a slightly modified version of the algorithm developed by Heminki [7], summarised in Alg. 1. Short data windows are considered, and gravity is estimated in windows that have sufficiently small variance. During periods with small variance, the sensor is approximately stationary, which means that the main acceleration measured is caused by gravity. Therefore, taking the window mean as gravitational acceleration is appropriate in windows with

small variance. In many situations, however, variation is to high to properly estimate gravity. High variance is caused by activities such as walking, cycling or travelling in a vehicle along an uneven road. To estimate gravity during such activities, the variance threshold is dynamically adjusted according to the current movement pattern. The variance threshold may increase until a hard upper threshold of 1.5 is reached. In case that happens, the gravity estimate becomes overly inaccurate, and it is more appropriate to use Mizell's technique to estimate the gravitational acceleration.
To make sure orientation changes of the sensor do not influence the gravity estimate, the estimate is reset in case a large shift in orientation is observed. Shifts in orientation occur when the phone is used, or when for example the wearer of the device stands up or sits down. Shifts in orientation are detected by comparing the current gravity estimate against the mean of the current window. Whenever these differ by more than 2 $\mathrm{m/s^2}$ along any of the axes, the gravity estimate is re-initialised to the mean of current accelerometer window.

Algorithm 1: Algorithm used to estimate the gravitational acceleration

Input : The current gravity estimate $\boldsymbol{g}$, and the new window of measurements A_w
Output: The new gravity estimate $\boldsymbol{g}$

$W_{\text{mean}} = \text{mean}(A_w)$ *% take the mean of A_w with respect to the x-, y- and z-axis*
$W_{\text{var}} = \text{var}(A_w)$ *% take the variance of A_w with respect to the x-, y- and z-axis*
if $|W_{mean} - \boldsymbol{g} \geq 2m/s^2$ **then**
    $\boldsymbol{g} = W_{\text{mean}}$
    $TH_{var} = [0.01\ 0.01\ 0.01]$
else if $W_{var} < 1.5$ **then**
    if $W_{var} < TH_{var}$ **then**
        $\boldsymbol{g} = W_{\text{mean}}$
        $TH_{var} = (W_{\text{var}} + TH_{var})/2$
        $VarInc = 0.1 \cdot TH_{var}$
    else
        $TH_{var} = TH_{var} + VarInc$
    end
else
    $\boldsymbol{g} = W_{\text{mean}}$
end

Heminiki uses windows of 1.2 seconds to estimate gravity, combined with a sample rate of at least 60 Hz. This results in at least 72 measurements per window. We collect data at a sample rate of 5Hz, which gives only 6 measurements in 1.2 seconds. Forced by this, a choice must be made between two non-optimal situations:

1. Taking short windows to be able to observe (almost) all changes in gravitational acceleration.
2. Taking longer windows such that enough data points are included.

Both situations have their drawbacks. Choosing method 1 results in more estimates, that are most likely less accurate due to lack of enough data points. However, choosing method 2 results in fewer estimates that apply to relatively large windows. This too is most likely inaccurate. In Chapter 7, Alg. 1 is tested with multiple window lengths and results are shown.

Due to the fact that only accelerometer data is used, a less accurate estimate of the gravitational acceleration will be obtained compared to the case when multiple sensors are used. However, recall that by scope of the project, other sensors are not allowed.

3.4 Computation of magnitude, angle and angle speed

Next to the vertical and horizontal linear acceleration, we are also interested in the magnitude of linear acceleration $\boldsymbol{l}$. The linear acceleration magnitude, which will be called c, is calculated by

$$c = \sqrt{l_x^2 + l_y^2 + l_z^2}. \tag{3.3}$$

The acceleration magnitude c is calculated for all measurements in A_w.

Two other relevant characteristics of the accelerometer measurements are the angle between the linear acceleration and the gravitational acceleration, and the related angular velocity. To calculate the angle, both the gravitational acceleration $\boldsymbol{g}$ and the linear acceleration $\boldsymbol{l}$ must be known. The angle is then calculated using Equation (3.4),

$$\alpha = \arccos\left(\frac{\boldsymbol{g} \cdot \boldsymbol{l}}{\|\boldsymbol{g}\| \cdot \|\boldsymbol{l}\|}\right), \tag{3.4}$$

where α is the angle between $\boldsymbol{g}$ and $\boldsymbol{l}$. The angle is calculated for each acceleration measurement in A_w.

Subsequently, the angular velocity is calculated. Angular velocity refers to how fast the orientation of an object changes over time, and is therefore calculated by taking the difference in two consecutive angles, and dividing this difference by the difference in time. Suppose that α_1 and α_2 are angles related to two consecutive measurements in A_w, with corresponding time points t_1 and t_2, then the angular velocity v is calculated as

$$v = \frac{\mathrm{d}\alpha}{\mathrm{d}t} = \frac{\alpha_2 - \alpha_1}{t_2 - t_1}. \tag{3.5}$$

3.5 Non-constant sample frequency

As said before, the rate at which samples are taken, differs from 5Hz to 200Hz. This is due to how the accelerometer works, and cannot be influenced. Because of this, it is possible that the different data points in A_w are not taken with equal time distances from each other. Therefore, it happens that some consecutive data points are very close together, while others are more apart. This is illustrated in Figure 3.4. All blue dots are data points collected at approximately 5Hz, the red dots are extra data points added by the phone, for no clear reason.

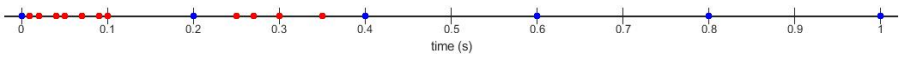


Figure 3.4: Data points representing timestamps of data collected with the accelerometer.

It is not desirable to keep these additional data points for several reasons:

- For classification purposes, several properties of the data have to be determined, such as the mean and the variation. However, if the mean is taken over a window with data points that are unevenly distributed over time, this will influence the quality of the mean. Clearly, the mean is more representative if data points are evenly distributed over time.
- Some features, such as the fast Fourier transform, require data to be evenly distributed over time, with a constant sample frequency. Hence, the fast Fourier transform cannot be used if all data points are kept.
- More data means more computations to be done, resulting in high battery usage.

For the above reasons, the next step in the preprocessing is interpolation. Interpolation is done with a sample frequency of 5Hz, the minimum rate at which data is collected.

To interpolate the data, first a grid of 'new' timestamps is created. The first grid point x_1 is the timestamp of the first data point present in window A_w, which is t_1. Then, the grid is constructed as follows: $\{x_1 = t_1, x_2 = t_1 + 0.2, x_3 = t_1 + 0.4, \ldots, x_m\}$, where x_m is as close to t_{n_w} as possible, the time point of the last measurement in A_w.

Now, let the grid points be $\{x_1, x_2, \ldots, x_m\}$, with m the total number of grid points. Using linear interpolation, the data is interpolated on the grid.

Linear interpolation works as follows: For every point $x_i,\ i = 1, \ldots, m$ in the grid, two consecutive values t_1 and t_2 must be found for which it holds that $x_i \in [t_1, t_2)$. The function values corresponding to t_1 and t_2 are called $f(t_1)$ and $f(t_2)$. A straight line is drawn between the coordinates $(t_1, f(t_1))$ and $(t_2, f(t_2))$, see Figure 3.5. The goal is to find the value of $f(x_i)$. By linear interpolation, see Figure 3.5, we obtain Equation (3.6) for the value of $f(x_i)$. In this equation, t_1 and t_2 are two consecutive time points in the window.

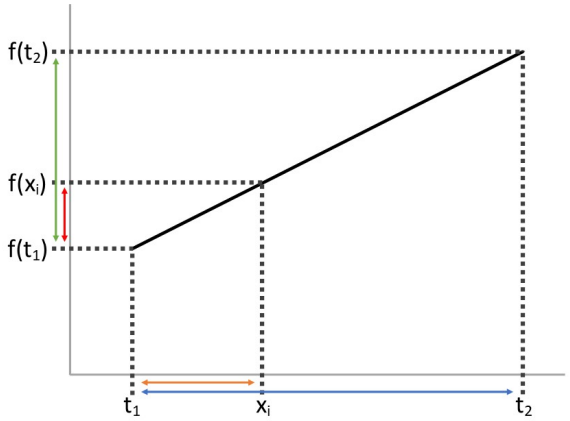


Figure 3.5: Schematic representation of linear interpolation.

$$f(x_i) = [f(t_2) - f(t_1)] \cdot \frac{x - t_1}{t_2 - t_1} + f(t_1), \qquad x_i \in [t_1, t_2),\ i = 1, \ldots, m, \tag{3.6}$$

For this thesis, we choose to use linear interpolation since it does not require a lot of computational effort [26]. Therefore, it can be used on a smartphone without draining the battery. Another possible option would be using spherical interpolation.

After interpolating the data, the preprocessing is finished.

3.6 Preprocessing summarised

The flow chart in Figure 3.6 provides an overview of all the steps used to preprocess the collected accelerometer data.

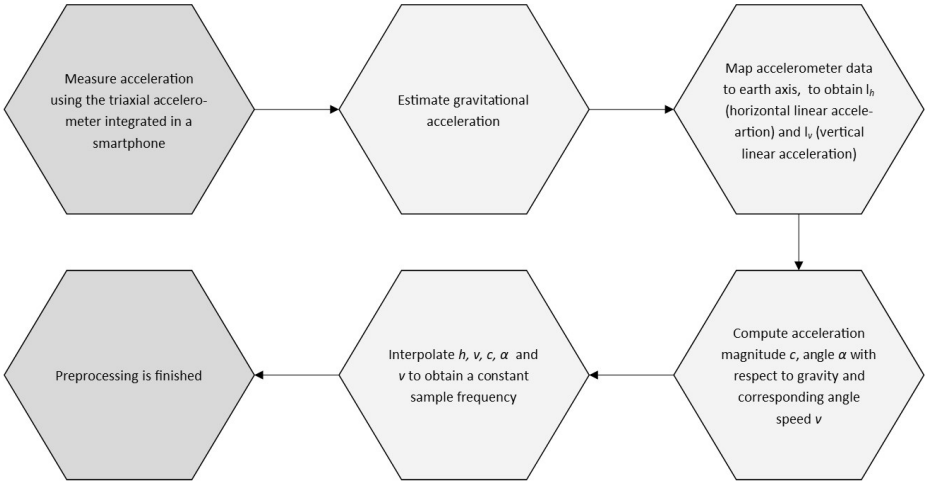


Figure 3.6: Flow chart summarising the preprocessing process.

Chapter 4

Feature selection using L1 regularised logistic regression

Most machine learning algorithms have an integrated feature selection. However, using a machine learning algorithm on a smartphone would most likely drain the battery way faster than desired. Therefore, we had to come up with our own way of feature selection. This section describes the process of calculating features and making a selection of useful features, using L1 regularised logistic regression.

4.1 Computation of features

In the previous chapter, preprocessing accelerometer data is discussed. The following list shows all relevant computed data that is contained in a window, after preprocessing.

- **Linear acceleration magnitude:** $\boldsymbol{c} = \{c_1, c_2, \ldots, c_{n_w}\}$, magnitude of the linear acceleration.
- **Vertical linear acceleration:** $\boldsymbol{l_v} = \{l_{v_1}, l_{v_2}, \ldots, l_{v_{n_w}}\}$, linear acceleration pointing away from the surface of the earth.
- **Horizontal linear acceleration:** $\boldsymbol{l_h} = \{l_{h_1}, l_{h_2}, \ldots, l_{h_{n_w}}\}$, linear acceleration parallel to the surface of the earth.
- **Angle:** $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_{n_w}\}$, angle between the gravitational acceleration and the linear acceleration.
- **Angle speed:** $\boldsymbol{v} = \{v_1, v_2, \ldots, v_{n_w-1}\}$, angle speed related to the angle between the gravitational acceleration and the linear acceleration.

For all five vectors, kind of *randomly* and based on common sense, a list of potentially relevant features for the classification model is proposed. A side constraint for the proposed features is that they can be computed fast. The idea then is that L1 regularised logistic regression will identify which of the features will help with the classification of travel modes. The process of making a selection of features is described in Section 4.3.

  - **Mean:** The mean of the vector components.
  - **Variance:** The variance of the vector components.
  - **Maximum:** It can happen that some measurements deviate from reality. For instance, a very high measurement that is not in line with the rest of the measurements in the window can occur. Therefore, instead of taking the maximum, the 95th percentile is taken. This means that the 5 percent highest numbers are skipped and the next highest number is taken as the

maximum. We expect this to be enough to avoid the event that a very high measurement, which is most likely noise, is chosen as the maximum.

- **Minimum:** For the reason mentioned above at *Maximum*, the 5th percentile is chosen as the minimum value.
- **Skewness:** The skewness of the window is a measure for how the data is distributed around its mean.
- **Kurtosis:** This parameter is a measure for the probability of outliers. The higher the kurtosis, the less chance there is for having an outlier.
- **Median:** The middle number in the vector.
- **Most dominant frequency:** Both walking and cycling consist of repeated movements that can be detected by a smartphone. When driving a motorised vehicle, less repeated movements happen that can be detected by the smartphone. This makes sense, since one sits still while driving a car, but has to move in order to walk or cycle. Therefore, every type of movement has its own frequencies with which movements are repeated. Most likely, walking and cycling have a higher frequency with which movements are repeated than driving a motorised vehicle, if it is already the case that there is some repeated movement in the latter case.
  Therefore, one of the features is equal to the frequency that is most dominant in the signal. For example, if one takes 2 steps per second while walking, the most dominant frequency will be 2 Hz. To obtain this frequency, the Fourier Transform is computed, with the aid of the fft (fast fourier transform). The frequency with the hightest energy level is then selected as the frequency that is most dominant. For more information about the fft and how to calculate the most dominant frequency, see [27].
- **Root mean square:** To calculate the root mean square, the following formula is used:

$$\mathrm{RMS}_x = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}$$

- **Zero-crossings:** The number of zero-crossings is equal to the number of times that two consecutive measurement in a vector do not have the same sign. For instance, $\boldsymbol{x} = [1, -2, 4, 5, -7]$ contains 3 zero-crossings, since there are three cases for which consecutive numbers do not have the same sign.
- **Energy:** To calculate the energy of the signal, the sum of the squares of all elements in the vector is taken.

A final feature that is calculated is based on linear acceleration $\boldsymbol{l}$. For each measurement i in window w, l is a vector containing linear acceleration in directions x, y and z. The calculated feature is called the **Signal Magnitude Area**. The signal magnitude area is calculated by taking the sum of the absolute values of the linear acceleration in three directions,

$$\mathrm{SMA} = \sum_{i=1}^{n_w} \left(|l_x^{(i)}| + |l_y^{(i)}| + |l_z^{(i)}| \right),$$

where n_w is the total number of measurements inside window w [28].

Computing the above features results into a vector of feature values of length 56. The algorithm should consume as few battery capacity as possible. Therefore, it is desirable to use as few features as possible in the classification algorithm. To make a selection of features to be used by the model, L1 regularised logistic regression is used. The remainder of this chapter describes L1 regularised logistic regression and how it can be used to make a selection of features to be used by the model.

4.2 L1 regularised logistic regression

4.2.1 Basic principle of L1 regularised logistic regression

The idea behind logistic regression is that, given a few data points, a curve is fit through those data points. This is done in order to estimate the probability that the data points belong to a certain class. In Figure 4.1, an example of a curve fitted by logistic regression is given, for the case that two classes and one feature are involved. The value of this feature is given on the x-axis, and the probability p to belong to class 1 is given by the value of the curve that corresponds to the values on the x-axis. Transforming the curve using the logit-function, see Equation (4.1), results into a straight line, see Figure 4.2.

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) \tag{4.1}$$

Hence, in case one feature is used, this line is represented by

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x,$$

for some coefficients β_0 and β_1. In case multiple features are involved, the line is represented by

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n, \tag{4.2}$$

for some coefficients $\beta_0, \beta_1, \ldots, \beta_n$ and feature values $x_1, \ldots, x_n$ [29].

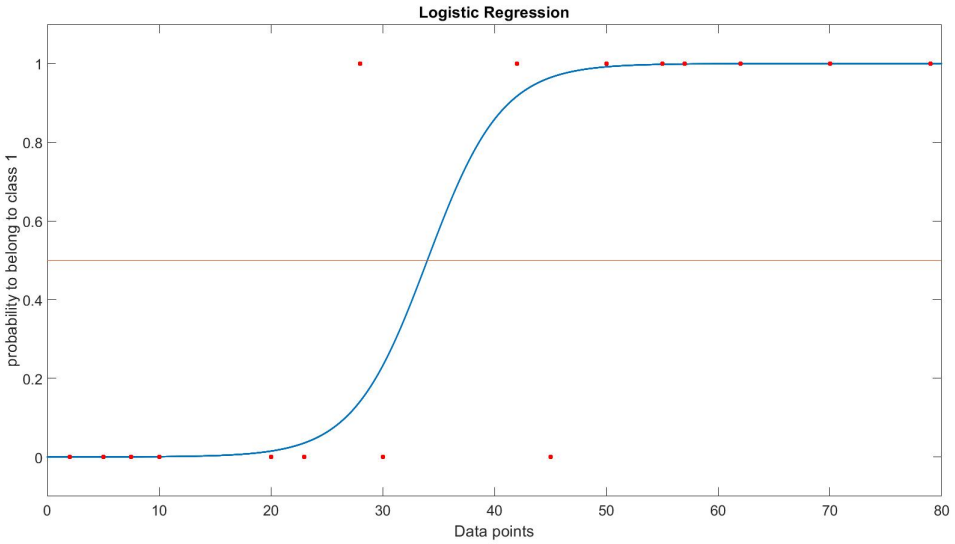


Figure 4.1: Example of a curve fitted by logistic regression.

The curve is thus represented by an equation that depends on multiple coefficients $\beta_i, i \in \{0, \ldots, n\}$. Each of the coefficients relates to one of the feature values. Section 4.2.2 goes into more detail about finding optimal coefficients β_i in order to get the best fitted curve to the data.

By adding L1 regularisation to logistic regression, two things are ensured. The first and main reason for performing regularisation, is to makes sure that the curve will not fit the training data perfectly, to avoid so-called *overfitting*. We say that the curve is overfit, if the model is well fitted to training data, but very ill-fitted to test data. The second thing, and thereby the big advantage of L1

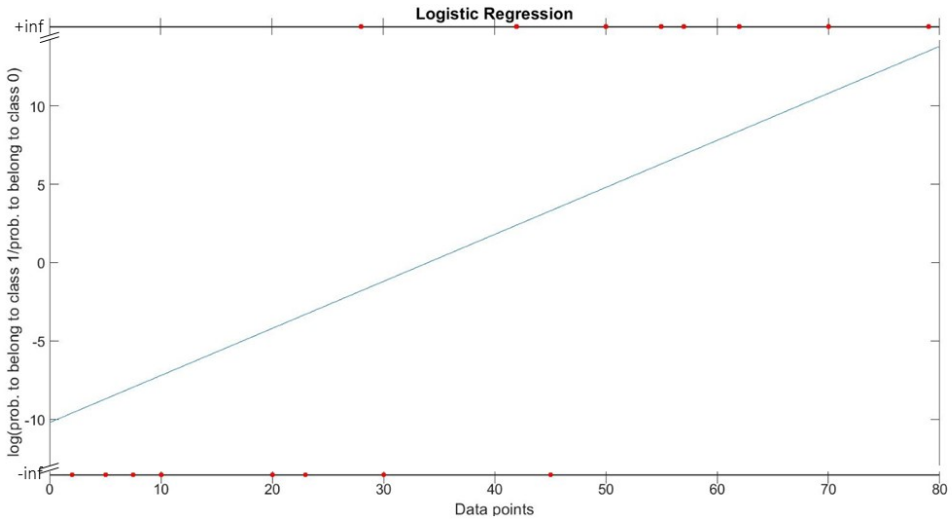


Figure 4.2: Example of a curve fitted by logistic regression, transformed using the logit function.

regularisation for our model, is that some of the coefficients used to form the curve, are forced to be zero. This means that the features related to those specific coefficients are not relevant in classifying the data points. Therefore, those features can be left out and do not have to be calculated. The selection of features is further explained in Section 4.3.

The fact that feature selection is possible with L1 regularised logistic regression, is the main reason to use it in the model. To be allowed to use logistic regression, however, the data has to fulfil some requirements, which are listed below [30].

- The classes must be non-ordinal, so no clear ordering may exist for the classes. This is clearly the case for the classes *still, walking, cycling, bus, train* and *car*.
- All observations must be independent from each other. Hence, data may not be used more than once to obtain multiple observations. Since our observations are obtained using non-overlapping windows, this requirement is fulfilled.
- Preferably, the used features may not be too highly correlated to each other. Using to highly correlated features makes the computation of logistic regression unstable and increases computation times. In our case, one could argue that *mean* and *median* are highly correlated. L1 regularised logistic regression was performed with and without including *median* in the list of features. Both times, the same coefficients were found. Hence, we decided to keep both *mean* and *median* in the list of suggested features.
- A large sample size is needed. Since we have at least 500 windows of measurements per class, this requirement is easily obtained.

Based on the above, we conclude that using L1 regularised logistic regression is possible for the data we are dealing with.

4.2.2 Binary logistic regression

Let us first consider binary logistic regression. In this case we are given a set $S = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ of m training samples. Here, $\boldsymbol{x}^{(i)}$ represents the i-th training sample consisting of n feature values calculated from the input data. Hence, $\boldsymbol{x}^{(i)}$ is of the form

$$\boldsymbol{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}\right].$$

Furthermore, $y^{(i)} \in \{0, 1\}$ is the class label for the i-th training sample. Since we are dealing with a binary case, there are only two classes the data can belong to. The construction of such a set S is discussed in Section 4.4.

In case of binary logistic regression, the curve is fitted using coefficients $\beta_i, i = 0, \ldots, n$ and probabilities are calculated with the aid of Equation (4.3). Here, $p\left(y^{(i)} = 1\right)$ is the probability that $y^{(i)} = 1$.

$$\ln\left(\frac{p\left(y^{(i)}=1\right)}{1-p\left(y^{(i)}=1\right)}\right) = \beta_0 + \beta_1 x_1^{(i)} + \cdots + \beta_n x_n^{(i)} \tag{4.3}$$

To simplify notation, we will add a 1 to the beginning of each $\boldsymbol{x}^{(i)}$, such that β_0 does not have to be denoted separately. The righthandside of Equation (4.3) therefore becomes $\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}$.

Rewriting Equation (4.3) gives

$$\begin{aligned}
\frac{p\left(y^{(i)}=1\right)}{1-p\left(y^{(i)}=1\right)} &= \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right), \\
p\left(y^{(i)}=1\right) &= \left(1-p\left(y^{(i)}=1\right)\right) \cdot \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right), \\
p\left(y^{(i)}=1\right) \cdot \left(1+\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)\right) &= \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right), \\
p\left(y^{(i)}=1\right) &= \frac{\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)}{1+\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)}, \\
&= \frac{1}{1+\exp\left(-\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)}.
\end{aligned} \tag{4.4}$$

And hence,

$$\begin{aligned}
p\left(y^{(i)}=0\right) &= 1 - p\left(y^{(i)}=1\right) \\
&= \frac{1}{1+\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)}.
\end{aligned} \tag{4.5}$$

The coefficients are fit by maximum likelihood, using the conditional likelihood of $y^{(i)}$ given $\boldsymbol{x}^{(i)}$ and $\boldsymbol{\beta}$. The log-likelihood for m observations is

$$l(\beta) = \sum_{i=1}^{m} \log p\left(y^{(i)} = 1 | \boldsymbol{x}^{(i)}, \boldsymbol{\beta}\right).$$

Rewriting the log-likelihood yields

$$\begin{aligned}
l(\beta) &= \sum_{i=1}^{m} \left(y^{(i)} \log p\left(y^{(i)}=1|\boldsymbol{x}^{(i)},\boldsymbol{\beta}\right) + (1-y^{(i)}) \log\left(1-p\left(y^{(i)}=1|\boldsymbol{x}^{(i)},\boldsymbol{\beta}\right)\right)\right), \\
&= \sum_{i=1}^{m} \left(y^{(i)} \left[\log p\left(y^{(i)}=1|\boldsymbol{x}^{(i)},\boldsymbol{\beta}\right) - \log\left(1-p\left(y^{(i)}=1|\boldsymbol{x}^{(i)},\boldsymbol{\beta}\right)\right)\right] + \log\left(1-p\left(y^{(i)}=1|\boldsymbol{x}^{(i)},\boldsymbol{\beta}\right)\right)\right), \\
&= \sum_{i=1}^{m} \left(y^{(i)} \log\left(\frac{p\left(y^{(i)}=1|\boldsymbol{x}^{(i)},\boldsymbol{\beta}\right)}{\left(1-p\left(y^{(i)}=1|\boldsymbol{x}^{(i)},\boldsymbol{\beta}\right)\right)}\right) + \log\left(1-\frac{\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)}{1+\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)}\right)\right), \\
&= \sum_{i=1}^{m} \left(y^{(i)} \boldsymbol{\beta}^T \boldsymbol{x}^{(i)} + \log 1 - \log\left(1+\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)\right)\right).
\end{aligned}$$

To maximize the log-likelihood, the derivative is taken and set to zero.

$$\begin{aligned}\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{m} \left(y^{(i)} \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)} \frac{\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)}{1 + \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)} \right), \\ &= \sum_{i=1}^{m} \left(\boldsymbol{x}^{(i)} \left(y^{(i)} - \frac{\exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)}{1 + \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)} \right) \right) = 0. \end{aligned} \tag{4.6}$$

Solving Equation (4.6) yields coefficients $\boldsymbol{\beta}$ for which the curve is optimally fit to the training data. For a detailed description of how the equation is solved, see [29]. In conclusion, the model that is solved to find coefficients $\boldsymbol{\beta}$ uses maximum likelihood and looks as follows [31]:

$$\max_{\boldsymbol{\beta}} \sum_{i=1}^{m} \log p\left(y^{(i)} = 1 | \boldsymbol{x}^{(i)}, \boldsymbol{\beta} \right). \tag{4.7}$$

The model is only solved for the case that $y^{(i)}$ equals 1, since the probability of belonging to class 0 is equal to 1 minus the probability of belonging to class 1.

4.2.3 Multiclass logistic regression

In multiclass logistic regression, the restriction to have only two classes no longer exists. Instead, it is possible to have multiple (i.e. more than two) class labels. To be able to handle those multiple class labels, some things have to be adjusted to the way the logistic regression is performed. In the following two subsections, two ways of multiclass logistic regression are described, namely multinomial logistic regression, and one-vs-rest logistic regression, which will be referred to as ovr logistic regression in the remainder of this report.

For aid of simplicity, we will assume that there are now K classes to consider.

In both ovr logistic regression and multinomial logistic regression, computing the coefficients is done in almost the same way as it is done for binary logistic regression, using the model described with Equation (4.7). The only difference is that $y^{(i)}$ can be equal to 1 up to K instead of only equal to 1.

One-vs-rest logistic regression

The basic idea behind ovr logistic regression, is to take the data from one class, and compare it to all other classes, as if all those other classes were one class instead of multiple classes. Hence, the problem is transformed from comparing K classes at once, to performing binary logistic regression K times. Thereby, K sets of coefficients are obtained, one for each time logistic regression is performed. For all those binary cases, we compute the probability of belonging to the 'lonely' class as described in section 4.2.2, namely

$$p\left(y^{(i)} = k\right) = \frac{1}{1 + \exp\left(-\boldsymbol{\beta}_k^T \boldsymbol{x}^{(i)}\right)}, \quad \forall k = 1, \ldots, K.$$

Computing the above probabilities for all the classes, will result in probabilities that do not necessarily add up to one, if we consider all classes. For instance, if we consider a case where $K = 3$, the probabilities can become

$$\begin{aligned} p(y = 1) &= 0.25, \\ p(y = 2) &= 0.30, \\ p(y = 3) &= 0.28. \end{aligned}$$

Clearly, those numbers do not add up to one. The probabilities are therefore scaled such that they add up to one. The formula for computing the probabilities thus becomes

$$p\left(y^{(i)} = k\right) = \frac{1}{\sum_{k=1}^{K} \frac{1}{1+\exp\left(-\boldsymbol{\beta}_k^T \boldsymbol{x}^{(i)}\right)}} \cdot \frac{1}{1 + \exp\left(-\boldsymbol{\beta}_k^T \boldsymbol{x}^{(i)}\right)}, \quad \forall k = 1, \ldots, K. \tag{4.8}$$

Multinomial logistic regression

Instead of handling the multiclass logistic regression with multiple binary cases, multinomial logistic regression handles all the classes at once. To calculate probabilities, one reference class has to be taken out of all classes in the classification problem. The choice of the reference class is arbitrary, since it does not influence the final result.

For this report, we chose to set the last class, class K, as our reference class, Then, given the definition of logistic regression, the probabilities for class 1 to $K-1$ can be calculated by using the formulas below,

$$\begin{aligned}
\ln \frac{p\left(y^{(i)}=1\right)}{p\left(y^{(i)}=K\right)} &= \boldsymbol{\beta}_1^T \boldsymbol{x}^{(i)}, \\
\ln \frac{p\left(y^{(i)}=2\right)}{p\left(y^{(i)}=K\right)} &= \boldsymbol{\beta}_2^T \boldsymbol{x}^{(i)}, \\
&\vdots \\
\ln \frac{p\left(y^{(i)}=K-1\right)}{p\left(y^{(i)}=K\right)} &= \boldsymbol{\beta}_{K-1}^T \boldsymbol{x}^{(i)}.
\end{aligned}$$

Here, $\boldsymbol{\beta}_1^{(i)}, \ldots, \boldsymbol{\beta}_K^{(i)}$ are all vectors of coefficients of length $n+1$, where n is the number of features computed. Taking the exponential on both sides and multiplying with $p\left(y^{(i)}=K\right)$ yields

$$\begin{aligned}
p\left(y^{(i)}=1\right) &= p\left(y^{(i)}=K\right) \cdot \exp\left(\boldsymbol{\beta}_1^T \boldsymbol{x}^{(i)}\right), \\
p\left(y^{(i)}=2\right) &= p\left(y^{(i)}=K\right) \cdot \exp\left(\boldsymbol{\beta}_2^T \boldsymbol{x}^{(i)}\right), \\
&\vdots \\
p\left(y^{(i)}=K-1\right) &= p\left(y^{(i)}=K\right) \cdot \exp\left(\boldsymbol{\beta}_{K-1}^T \boldsymbol{x}^{(i)}\right).
\end{aligned}$$

The probability that $y^{(i)}$ equals K can now be determined, by the fact that probabilities add up to one,

$$p\left(y^{(i)}=K\right) = 1 - \sum_{k=1}^{K-1} p\left(y^{(i)}=k\right).$$

The fact that

$$\sum_{k=1}^{K-1} p\left(y^{(i)}=k\right) = p\left(y^{(i)}=K\right) \cdot \sum_{k=1}^{K-1} \exp\left(\boldsymbol{\beta}_k^T \boldsymbol{x}^{(i)}\right),$$

yields

$$\begin{aligned}
p\left(y^{(i)}=K\right) &= 1 - p\left(y^{(i)}=K\right) \cdot \sum_{k=1}^{K-1} \exp\left(\boldsymbol{\beta}_k^T \boldsymbol{x}^{(i)}\right), \\
&= \frac{1}{1+\sum_{k=1}^{K-1} \exp\left(\boldsymbol{\beta}_k^T \boldsymbol{x}^{(i)}\right)},
\end{aligned} \tag{4.9}$$

and

$$p\left(y^{(i)}=k\right) = \frac{\exp\left(\boldsymbol{\beta}_k^T \boldsymbol{x}^{(i)}\right)}{1+\sum_{j=1}^{K-1} \exp\left(\boldsymbol{\beta}_j^T \boldsymbol{x}^{(i)}\right)}, \quad \forall k = 1, \ldots, K-1. \tag{4.10}$$

4.2.4 L1 regularisation

The basic idea of L1 regularisation is adding a penalty term to the model given in Equation (4.7). As said before, the purpose of this is to make the model less fitted to the training data to prevent overfitting.

The penalty term added by L1 regularisation is the sum of the absolute value of the coefficients, multiplied by some regularisation parameter $\lambda_k > 0$. Normally, the intercept term, β_0, is left out of the penalty. For simplicity, we will ignore this. We then get the model as shown in Equation (4.11).

$$\max_{\boldsymbol{\beta}_k} \sum_{i=1}^{m} \log p\left(y^{(i)} = k \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\beta}_k\right) - \lambda_k \sum_{i=1}^{K} |\boldsymbol{\beta}_k|. \tag{4.11}$$

The regularisation parameter λ_k has to be determined based on the training data, and each class can get a different λ_k.

To find values for all λ_k that are best to use, we split the training data into two parts, of which one part is now used for training the model, and the other part is used for testing the model. Now, train the model for a lot of different values for λ_k, ranging from 0.00001 to 10000. Then, with the aid of the test set, validate for each λ_k how many samples are classified correctly. Then, choose as final λ_k's the values that performed best in correctly classifying test samples. [31].

4.3 Feature selection

By using L1 regularised logistic regression, the chosen value for regularisation parameter λ will make some coefficients equal to 0. In cases where this happens, we conclude that we do not need the corresponding feature for classification purposes.

The larger the value chosen to use for λ, the less features are used in the feature selection. However, this does not automatically mean a worse result. It may even be the case that using less features results into a better classification. Figure 4.3 shows the values of 5 coefficients related to 5 feature values, computed for different values of λ. Indeed, higher values for λ result into more coefficients equal to 0.

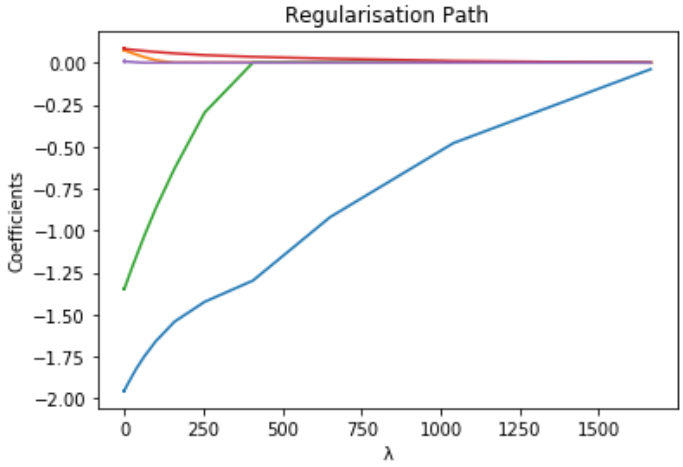


Figure 4.3: Coefficients for logistic regression computed using L1 regularisation with several values of λ

In Chapter 7, the selection of features, based on L1 regularised logistic regression will be discussed.

4.4 Training data

In order to find the coefficients $\boldsymbol{\beta}_k$ that are used in Equations (4.8), (4.9) and (4.10), it is required to have some training data, see Section 4.2.2.

Since the classification model must be able to classify movements in the classes *still, walking, cycling, bus, train* and *car*, training data needs to be collected for those six classes. Using a smartphone and an app provided by Mobidot, we were able to create our own training data. Over 30 hours of data was collected and used to train coefficients used for performing L1 regularised logistic regression.

The collected training data is divided into windows of length s, and features are calculated for each window, see Section 4.1. This results into m training samples of the form $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$, where $\boldsymbol{x}^{(i)}$ is a vector containing the computed features for the i-th window of training data, and $y^{(i)} \in \{1, \ldots, K\}$ is the class label belonging to the data in the window. The m training samples form the training set $S = \{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\}_{i=1}^{m}$. Based on this set, coefficients $\boldsymbol{\beta}_k$ are determined for all classes $k \in K$, where K is the set of classes used in the classification model.

Chapter 5

Classification algorithm

After preprocessing the data and calculating a set of derived features, the feature values are used to determine the used travel modes. In order to do so, two basic methods are developed, again based on logistic regression. Furthermore, four extensions to these models are developed, which can all be switched on and off as desired. In all cases, one of the basic methods must be selected to be used for classification, and extensions can be added. In Chapter 7, the performances of the basic models and the extensions are evaluated using five test cases, which are described in Chapter 6.

5.1 Basic methods

The two basic methods are both based on logistic regression. The choice for using logistic regression is based on the following.

- The coefficients used for logistic regression can be computed based on a training set. Hence, coefficients are computed once, and can from then on be used on every test set. Therefore, no additional computation has to be done regarding computation of coefficients, when it comes to online classification. Taking into account the requirement for battery-efficiency, using logistic regression for classification is suited.
- In fact, computing the feature selection also yields the coefficients used for logistic regression. Thereby, logistic regression can be used for both feature selection and classification, and no additional classification technique will be needed.

The first basic method is based on the 'one-vs-rest'-principle, see Section 4.2.3. For each window, probabilities to belong to each of the six classes are calculated using Equation (4.8). The class with the maximum probability is then chosen as the current travel mode. In the remainder of this report, referring to this basic method is done with OVR.

The second basic method works the same, only the probabilities are calculated based on multinomial logistic regression. This is done using Equations (4.9) and (4.10). In the remainder of this report, referring to this basic method is done with MULT.

5.2 Extension 1: Dealing with close probabilities

The two basic methods both select travel modes based on maximum probability. However, it can be the case that probabilities are as shown in Table 5.1. Both basic methods would now classify the window with class 4. However, since the probabilities associated with classes 3 and 4 are only 10% apart, it is not at all unlikely that class 3 is the correct travel mode to chose. However, if probabilities would be as shown in Table 5.2, it is very likely that class 4 is the correct travel mode. As can be seen

Class	1	2	3	4	5	6
Probability	0	0	0.45	0.55	0	0

Table 5.1: Possible outcome of probabilities calculated for 5 classes.

Class	1	2	3	4	5	6
Probability	0	0	0.1	0.9	0	0

Table 5.2: Possible outcome of probabilities calculated for 5 classes.

from the above example, picking the class with the highest probability will not always result into a correct classification, especially when probabilities are close to each other. Extension 1 provides a way to deal with situations in which the two highest probabilities are relatively close. The flow chart shown in Figure 5.1, shows how the extension works. It starts with creating a set of all classes in the classification problem. Then, using one of both basic methods, the probabilities of belonging to all of these classes are calculated. If it holds that $p_{\max} - p_{2\text{nd max}} \geq th_1$ for some threshold value $th_1 \in [0, 1]$, the class corresponding to the maximum probability is chosen. Otherwise, classes with a corresponding probability lower than $th_2 \in [0, 1]$, are removed from the set, since it would be improbable that one of those classes is the correct travel mode. After removing those classes, probabilities are again calculated using the formulas corresponding to the chosen basic method. This process of removing classes and recalculating and comparing probabilities is repeated, until one of three things happens:

1. $p_{\max} - p_{2\text{nd max}} \geq th_1$.
2. There is only one class left in the set.
3. After calculating probabilities, it does not hold that $p_{\max} - p_{2\text{nd max}} \geq th_1$ and there are no classes with probabilities lower than th_2. Hence, the selection of classes will be equal to the entire set.

If case 1 holds, the case corresponding to the maximum probability is set as the current travel mode. If case 2 holds, the leftover class is set as the current travel mode. Lastly, if case 3 holds, the window is classified the same as the previous window.

After a window is classified, the same process is repeated for the next window.

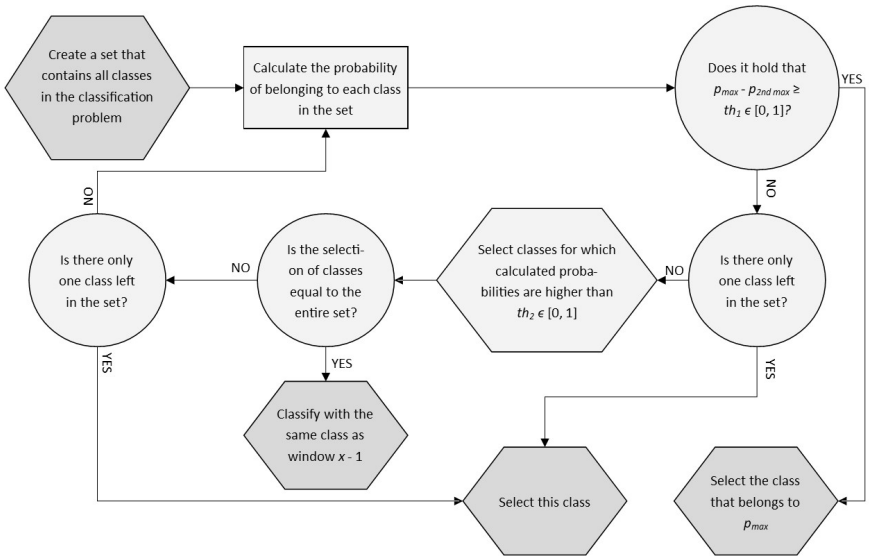


Figure 5.1: Flow chart for Extension 1

5.3 Extension 2: Walk-still logic

A possible sequence of consecutive classified travel modes could be *car - bike - train - walk*. However, for example, it is (almost) impossible to get on a bike directly out of the car without walking a short distance. Similarly, there are other combinations that are not possible in real life. In Extension 2, this sort of logical reasoning is implemented to remove the possibility of obtaining those combinations.

The extension is based on four rules. Notation-wise, $k-1$, k and $k+1$ represent the last, current en next classified activities, respectively, and K represents the entire set of activities. For the sake of completeness, the last two rules have been added.

1. if $k-1 \neq walk$ and $k = still$, then $k+1 \in \{still, walk, k-1\}$,
2. if $k \neq walk, k \neq still$, then $k+1 \in \{walk, still, k\}$.
3. If $k = walk$, then $k+1 \in K$.
4. if $k-1 = walk$ and $k = still$, then $k+1 \in K$,

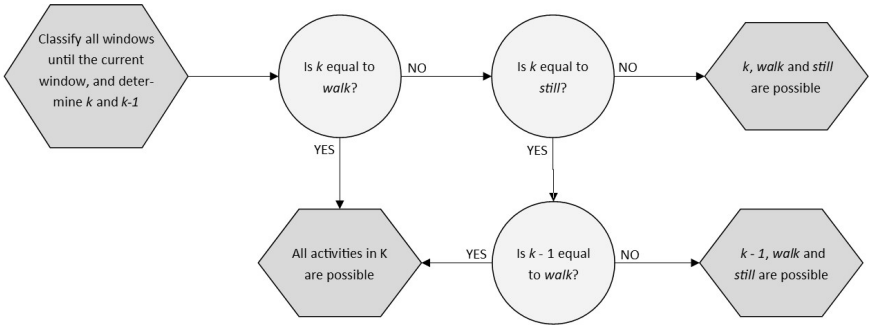


Figure 5.2: Flow chart for Extension 2

Using extension 2 all the time, has one major drawback. Consider the case where the current classified activity is *train*. Then Extension 2 would allow the next window to be classified as *train, walk* or *still*. This will give a problem if the current activity is not equal to *train*, which can be the case since classification is not flawless. Therefore, Extension 2 can only be used if it is certain that classification has been done correctly.

Therefore, we introduce a certainty measure, which is described in Section 5.6. Only if certainty is *high*, Extension 2 is used to make a selection of classes to chose from. Otherwise, the algorithm will skip this step.

5.4 Extension 3: Shortcut on computation times

It almost never happens that an activity is performed for the length of one window. Most activities take much longer. Extension 3 takes advantage of that.

In case the last window is classified as activity k with *high* certainty, the probability is calculated that the activity belonging to the current window is also k. This is done by making use of the equations used in one-vs-rest logistic regression, however, only the equation for class k is used, instead of calculating probabilities for all K classes. Hence, the probability of belonging to class k is calculated using Equation (5.1). This equation calculates the probability that a set of features belongs to class k. Obviously, 1 - $p\left(y^{(i)} = k\right)$ is the probability that the set of features does not belong to class k.

$$p\left(y^{(i)} = k\right) = \frac{1}{1 + \exp\left(-\beta_k^T x^{(i)}\right)}, \tag{5.1}$$

If now $p\left(y^{(i)} = k\right) \geq th_3$ for a threshold $th_3 \in [0, 1]$, class k is chosen as the activity belonging to the current window. If $p\left(y^{(i)} = k\right) < th_3$, the classification algorithm continues as usual.

5.5 Extension 4: Exponential smoothing of feature values

Suppose one is taking a ride on a bicycle and suddenly encounters a red traffic light. After five seconds, however, the traffic light turns green again and one can continue cycling. This will heavily influence the measurements done by the accelerometer. Suppose the five seconds of standing still exactly fall inside one window, then the feature values of this window will differ from the feature values of surrounding windows. Also, the feature values of this window are not equal to typical 'still-features' or typical 'cycling-features'. Hence, most likely, this window will not be classified correctly. Even worse is the case where the 5 seconds are divided over two windows, in which case both windows most likely will not be classified correctly.

To solve this problem, Extension 4 can be used. This extension is based on exponential smoothing of the feature values. The exponential smoothing equation used is of the form

$$\hat{\boldsymbol{x}}^{(i)} = \gamma \boldsymbol{x}^{(i)} + (1 - \gamma)\hat{\boldsymbol{x}}^{(i-1)} \tag{5.2}$$

where $\boldsymbol{x}^{(i)}$ is the feature vector for window i, $\hat{\boldsymbol{x}}^{(i)}$ the smoothed value for window i, and $\hat{\boldsymbol{x}}^{(i-1)}$ is the smoothed value for window $i - 1$. Parameter γ is called the smoothing constant [32]. The smaller γ, the more $\hat{f}_i$ depends on feature values of previous windows. This method requires little computation, as well as little extra storage space.

Exponential smoothing works best if the data pattern is *approximately horizontal* [32]. This means that the data fluctuates around a constant mean. During continuous movement in the same way, this requirement is met. Only if the way of movement is changed, the mean of the data changes. Basically, changes in travel mode cause a jump in the mean. Experiments should show whether the benefits of exponential smoothing outweigh the behaviour of exponential smoothing related to changes in travel mode, see Chapter 7.

5.6 Certainty measure

To be able to assess the quality of the classification, a certainty measure is suggested. This measure uses the computed probability p_w related to the class that has been selected for the current window by the classification algorithm, and the certainty calculated for the previous window, C_{w-1}. That said, certainty C_w is computed as

$$C_w = \frac{C_{w-1} + p_w}{2}. \tag{5.3}$$

We say that:

- certainty is *low* for window w, if $C_w \in [0, th_c)$,
- certainty is *high* for window w, if $C_w \in [th_c, 1]$,

for some threshold $th_c \in [0, 1]$. A specific value for threshold th_c is determined in Chapter 7.

It for instance can happen that a few consecutive windows are classified as walking with *high* certainty. Suppose the model would then make a mistake and classify the next window as cycling with $p_w > th_c$, where it had to be walking. Then, using the suggested measure, this classification will be assessed with *high* certainty, whereas the classification is not correct. To prevent this from happening, certainty C_w is reset to 0 in case a change in classified travel modes occurs. Hence, certainty is set to *low* in this case.

We chose to use this measure instead of just using p_w as a certainty measure, because this measure makes sure that p_w has to be quite high for a few consecutive windows, before certainty becomes *high*. Thereby, the chance of having a false *high* certainty is reduced.

Chapter 6

Experiment setup

To evaluate the model, some test cases are needed. Using the test cases, the behaviour of the model regarding some 'special' situations can be tested. Examples of such special situations are the following.

- **Transitions between two travel modes** It is important to know how the model behaves on transitions between two travel modes. Especially when Extension 4 is activated, we want to be able to see if there occurs a shift transitions.
- **Test data collected by a different person than the one that collected the training data** The model is trained on data that is collected by one person. To know if this suffices, it is needed to test the model with test data collected by a different person. In case it is not sufficient, training data should namely be collected by multiple persons.
- **A travel mode that is not contained in the six classes considered by the classification model** A lot of travel mode are outside the six classes contained currently contained in the classification model. It is interesting to see how the model behaves on a travel mode that is not one of the six classes. Perhaps some movemements can be contained in one class. For example, cyling on an electric bike perhaps behaves the same as cycling on a regular bike. This should be tested by including a travel mode that is not part of the six classes.

Based on the above list, the following requirements are chosen for the test cases.

- Each test situation consists of several travel modes.
- The travel modes *still, walking, cycling, bus, train* and *car* must be contained in the test situations.
- One of the test files has to contain a 'travel mode' that is not included in the classes considered by the classification model, to see what the model does in this case.
- All training data is collected by one person. Therefore, someone else has to create data for one test situation, to test the performance of the model in that case.

The data for the test cases is collected using an app provided by Mobidot. Below follows a list of test cases designed to fulfil the requirements. Each schematic representation shows the activities that are performed, together with the exact duration of each activity.

Case 1:
Contained classes: *still, walking* and *cycling.*
Peculiarities: None.

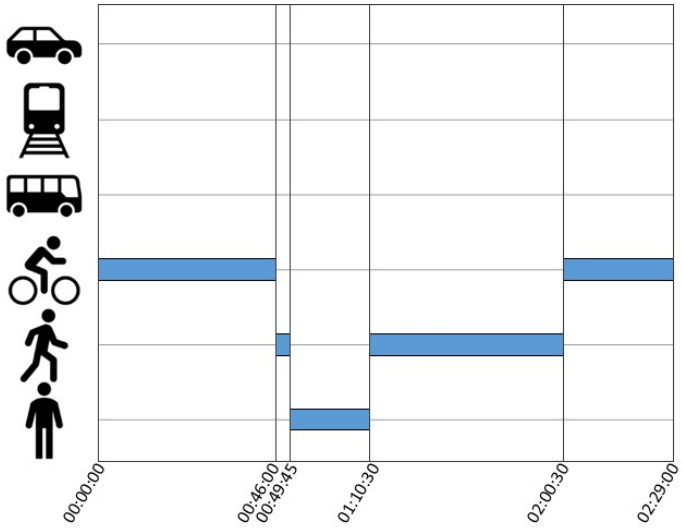


Figure 6.1: Schematic representation of Case 1

Case 2:
Contained classes: *still, walking, bus, train* and *car.*
Peculiarities: None.

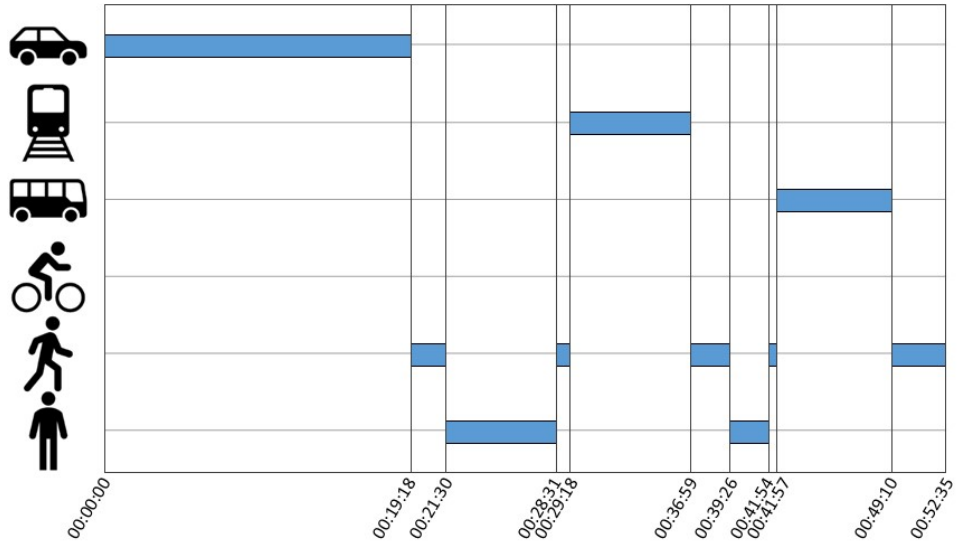


Figure 6.2: Schematic representation of Case 2

Case 3:
Contained classes: *walking* and *car*.
Peculiarities: The car ride includes driving in a traffic jam.

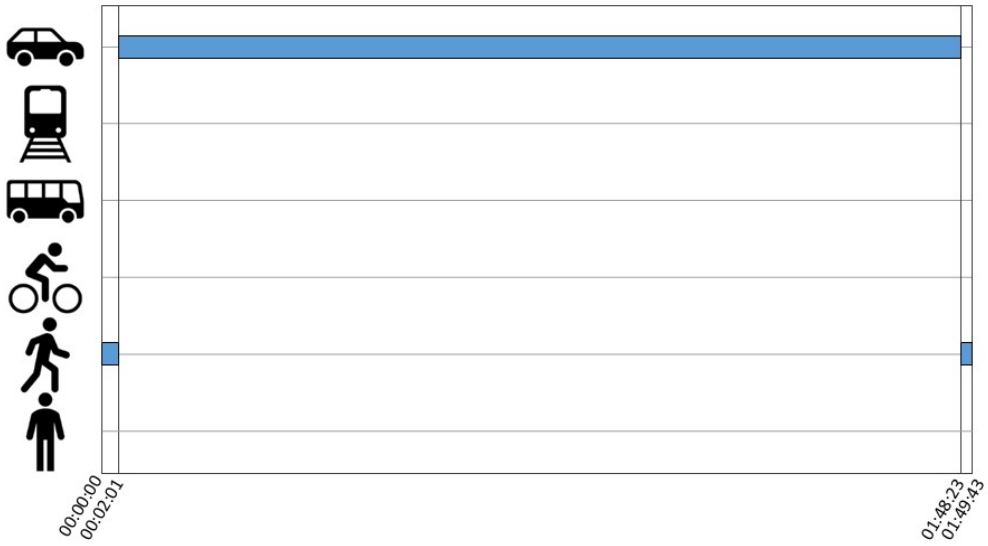


Figure 6.3: Schematic representation of Case 3

Case 4:
Contained classes: *walking*, *cycling* and *shopping*.
Peculiarities: This case includes *shopping*, which is not one of the classes considered by the classification model.

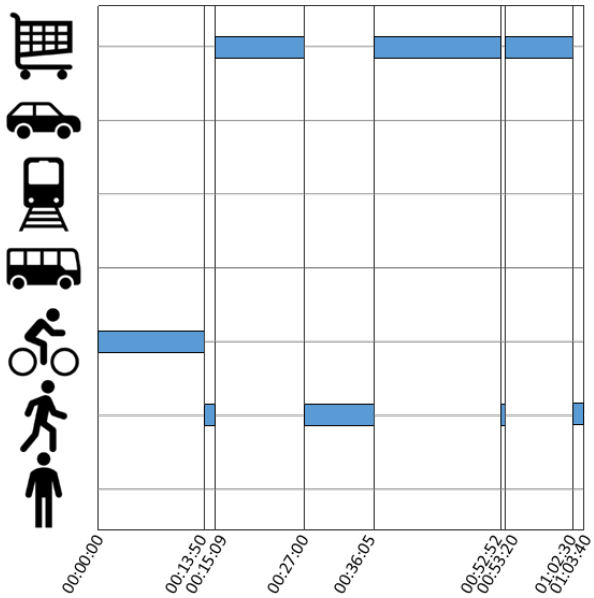


Figure 6.4: Schematic representation of Case 4

Case 5:
Contained classes: *still, walking* and *cycling.*
Peculiarities: The data for this case is collected by another person than the one that collected the training data for the classification model.

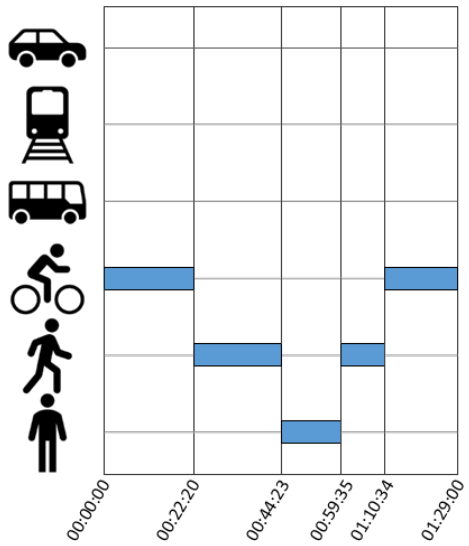


Figure 6.5: Schematic representation of Case 5

Chapter 7

Computational results

To be able to compute results with our model and test cases, first some parameter values have to be calibrated. Cases 1 and 2 are used to tune the parameter values, since together they contain movements of all 6 classes. After the parameter values have been determined, the model is tested using Cases 1 to 5. The determination of the parameter values and the obtained results are discussed in this chapter.

7.1 Gravitational acceleration estimate

In Section 3.3, the estimation of the gravitational acceleration is explained. As said, there are two options when choosing a window length for estimating gravitational acceleration.

1. Taking short windows to be able to observe (almost) all changes in gravitational acceleration.
2. Taking longer windows such that enough data points are included.

Therefore, gravitational acceleration is estimated using windows of 2.5, 5 and 10 seconds. Since the results for the x-, y- and z-direction of the accelerometer measurements are comparable, here only the results for the x-direction are given.

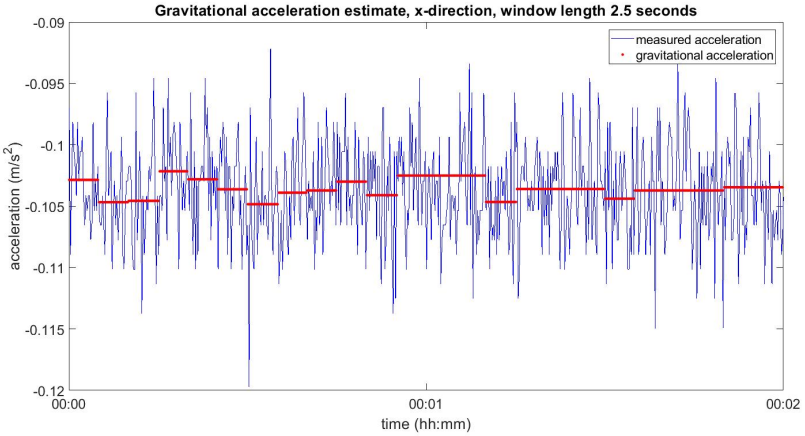


Figure 7.1: Gravitational acceleration estimate for movement *still*, using a window length of 2.5 seconds.

A disadvantage of using a window length of 2.5 seconds for estimating the gravitational acceleration is that the estimation is based on only 12 or 13 data points. This makes the estimate sensitive to noise

present in the measurements. This becomes clear when looking at Figure 7.1, where the estimates of the gravitational acceleration for movements belonging to the class *still* are depicted, using a window length of 2.5 seconds. During movements belonging to class *still*, we expect gravitational acceleration to be constant. However, the lower the window length used for estimation, the larger the variance, as can be seen from Table 7.1, where the variance, maximum and minimum of the gravitational acceleration in the x-direction are shown for window lengths 2.5 and 5 seconds. The corresponding estimates of the gravitational acceleration for movements belonging to the class *still* using a window length of 5 seconds, are shown in Figure 7.2.

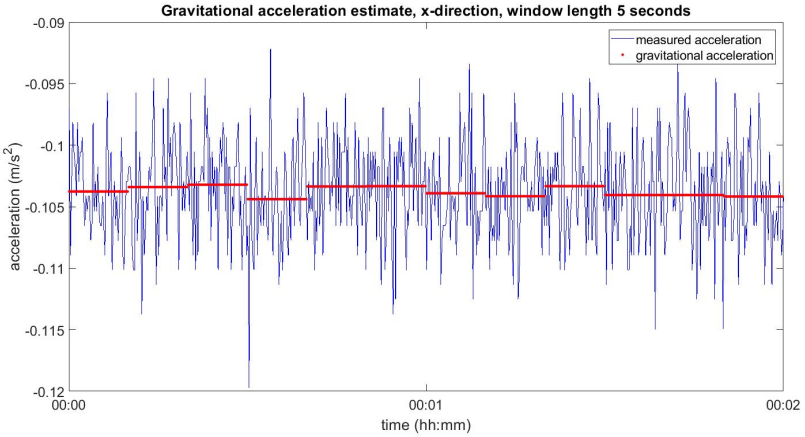


Figure 7.2: Gravitational acceleration estimate for movement *still*, using a window length of 5 seconds.

Window length	2.5 sec		5 sec
Minimum	-0.1048	$<$	-0.1044
Maximum	-0.122	$>$	-0.1032
Variance	0.054e-05	$>$	0.0155e-05

Table 7.1: Minimum, maximum and variance for the gravitational acceleration estimates, using window length 2.5 and 5 seconds.

Using a window length of 10 seconds also has disadvantages. When estimating gravitational acceleration for the class *walking*, see Figure 7.3, the estimate becomes almost constant. However, when walking, the phone will change position all the time, and therefore, the gravitational acceleration is not constant at all. When using smaller windows, this is less of a problem, as can be seen from Figure 7.4 and Table 7.2.

Window length	5 sec		10 sec
Minimum	-0.4576	$<$	-0.3948
Maximum	0.4877	$>$	0.4061
Variance	0.0558	$>$	0.0510

Table 7.2: Minimum, maximum and variance for the gravitational acceleration estimates, using window length 5 and 10 seconds.

Observation 1: *Both of the above mentioned disadvantages are less of a problem when using a window length of 5 seconds for estimating gravitational acceleration, see Figures 7.2 and 7.4.*

Therefore, henceforth the window length is set to 5 seconds for estimating gravitational acceleration.

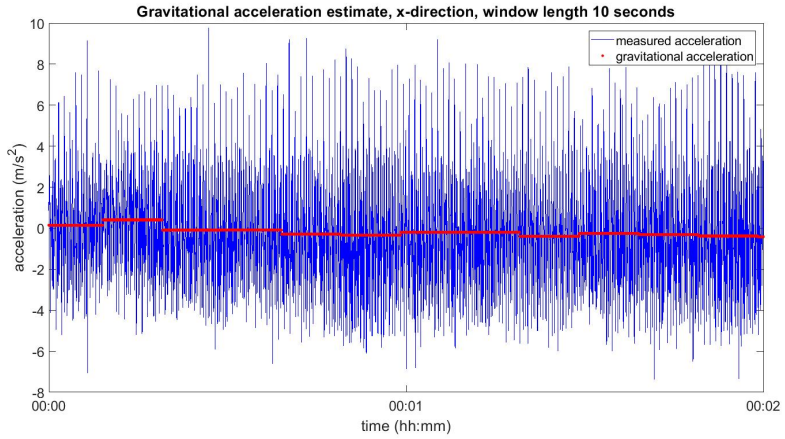


Figure 7.3: Gravitational acceleration estimate for movement *walking*, using a window length of 10 seconds.

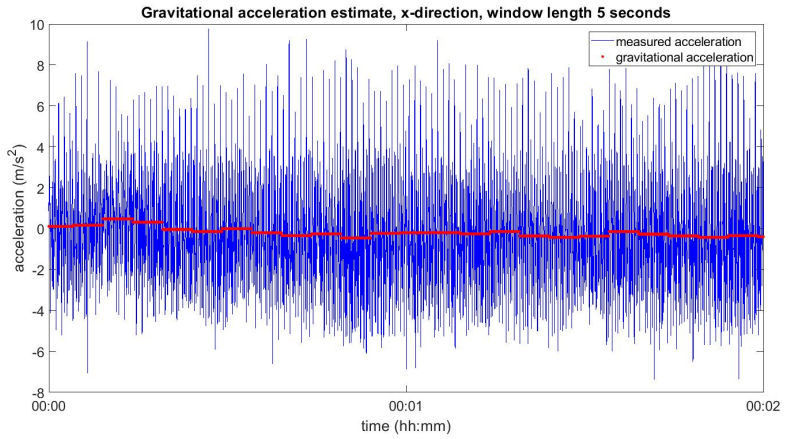


Figure 7.4: Gravitational acceleration estimate for movement *walking*, using a window length of 5 seconds.

Observation 2: *Both of the above mentioned problems are not solved by taking a window length of 5 seconds. To fully solve both issues, the sample frequency should be increased to be able to pick smaller window lengths.*

7.2 Window length

When choosing a length for the windows used for classification, it must be taken into account that gravitational acceleration is estimated using windows with a length of 5 seconds. Using windows for classification with a length equal to a multiple of 5 seconds, is computationally easiest. Therefore, window lengths of 5, 10, 15 and 20 seconds are tested, using both basic methods and Cases 1 and 2.

To determine which window length gives the best results, two measures are used.

1. **Accuracy:** The fraction of windows that is correctly classified,

$$\text{ACC} = \frac{\#\text{ correctly classified windows}}{\text{total }\#\text{ windows}} \cdot 100\%. \tag{7.1}$$

2. **Relative accuracy:** The mean fraction of windows that is correctly classified per class,

$$\mathrm{ACC}_{\mathrm{REL}} = \frac{1}{K} \cdot \sum_{k=1}^{K} \frac{\#\text{ correctly classified windows belonging to class } k}{\text{total } \#\text{ windows belonging to class } k} \cdot 100\%. \tag{7.2}$$

The results of using both measures to asses the quality of the classification are given in Table 7.3, where the maximum values are marked.

To finally asses the quality of the classification and to decide which window length is best to use, a score measure is suggested that takes all accuracy measures into account. To score each window, the mean of the differences between the achieved (relative) accuracy and the maximum (relative) accuracy is taken. For instance, the score for window length 5 seconds and basic method OVR is calculated as:

$$\frac{|72.04 - 64.60| + |71.85 - 69.61| + |34.86 - 34.86| + |42.91 - 42.91|}{4}$$

With this definition, the window length with the lowest score is assumed to be the better window length.

Basic method	Window length (sec)	Case 1 ACC (%)	Case 1 $\mathrm{ACC}_{\mathrm{REL}}$ (%)	Case 2 ACC (%)	Case 2 $\mathrm{ACC}_{\mathrm{REL}}$ (%)	score
OVR	5	64.60	69.61	34.86	42.91	2.42
	10	66.89	71.85	31.65	39.05	3.055
	15	69.30	55.53	23.70	31.05	10.52
	20	72.04	57.25	24.05	30.76	9.39
MULT	5	72.65	74.74	31.22	37.90	1.91
	10	75.95	77.68	31.65	38.87	0
	15	71.48	56.92	25.12	31.62	9.75
	20	72.48	57.48	25.95	31.93	9.0775

Table 7.3: *Accuracy* and *Relative accuracy* when classifying Cases 1 and 2, using the two basic methods and several window lengths.

Observation 3: *As can be seen in Table 7.3, it is clear that for basic method* MULT *it is best to use a window length of 10 seconds. For both Case 1 and Case 2, ACC and* ACC_{REL} *are maximum for a window length of 10 seconds, and therefore, the score is minimal for this window length.*

Observation 4: *For basic method* OVR*, using a window length of 5 seconds results in the lowest score.*

However, for Case 1, the maximum values for ACC and $\mathrm{ACC}_{\mathrm{REL}}$ are not obtained when using this window length. Also, the score for window length 5 seconds is only a bit lower than the score for window length 10 seconds, whereas computation wise taking a window length of 5 seconds will require more computation time overall. Taking into account that battery efficiency is one of the requirements for the classification model, we decided to use window lengths of 10 seconds for both basic methods.

7.3 Overlap

Since the gravitational acceleration is estimated in windows of 5 seconds, and classification is done in windows of 10 seconds, it is possible to use a five second overlap between consecutive windows used for classification. This will not violate the assumption made in Section 4.2.1 regarding independence of observations. This assumption only has to hold for training data. Running Cases 1 and 2 with 0 or 5 seconds overlap for both basic methods, gives the results as shown in Table 7.4.

Basic method	overlap (%)	Case 1		Case 2		score
		ACC (%)	ACC_{REL} (%)	ACC (%)	ACC_{REL} (%)	
OVR	0	66.89	71.85	31.65	39.05	0.99
	0.5	67.17	71.98	33.28	41.17	0
MULT	0	75.95	77.68	31.65	38.87	0.4325
	0.5	76.34	78.02	32.17	39.35	0

Table 7.4: *Accuracy* and *Relative accuracy* when classifying Cases 1 and 2, using the two basic methods with and without overlap

Observation 5: *The performance of the classification model is only a little bit better when using the overlap, however will result in almost twice the running time and computational effort, due to a double amount of windows that need classification.*

Taking in mind the requirement for battery-efficiency, we choose to use no overlap between consecutive windows.

7.4 Certainty threshold th_c

Cases 1 and 2 are used to establish threshold th_c for the certainty measure. Running both basic methods using a window length of 10 seconds, results into a list of classified windows with corresponding certainties. In Table 7.5, the 99th percentile of certainties that belong to a misclassified window is given for all combinations of basic methods and Cases. The 99 percentile is used instead of the maximum, to reduce the influence of noise.

Basic Method	Case 1	Case 2
OVR	0.62	0.29
MULT	0.59	0.47

Table 7.5: 99th percentile of *certainties* related to misclassified windows.

Observation 6: *The 99th percentile for certainties corresponding to a misclassified window equals 0.62 for Case 1, and 0.47 for Case 2.*

To have a little margin, we choose to set $th_c = 0.70$.

The above reasoning does not assure that all certainty measures belonging to misclassified windows are below 0.70. More testing on different cases should be done.

7.5 Parameter values for the extensions

There are several parameters present in the extensions, namely thresholds th_1 and th_2 in Extension 1, threshold th_3 in Extension 3 and smoothing constant γ in Extension 4. This section describes the determination of those parameter values.

Thresholds th_1 and th_2
Extension 1 contains two threshold values that have to be determined, namely thresholds th_1 and th_2. As described in Section 5.2, threshold th_1 is used to set a threshold on the minimal distance between the two highest computed probabilities. Threshold th_2 is used to eliminate classes with a computed probability less than th_2. Both thresholds must get a value between 0 and 1.

To find optimal values for thresholds th_1 and th_2, basic methods OVR and MULT are run with several values for th_1 and th_2. Scores are calculated using the method explained in Section 7.2. The fifteen results with the lowest scores for basic methods OVR and MULT are shown in Table 7.6 and Table 7.7, respectively.

			Case 1		Case 2		
Method	th_1	th_2	ACC (%)	ACC_{REL} (%)	ACC (%)	ACC_{REL} (%)	Score
OVR	0.1	0.1	85.01	88.97	24.37	27.98	1.85
	0.2	0.2	72.37	75.97	30.06	37.03	3.59
	0.3	0.2	72.15	75.8	29.43	36.27	3.81
	0.5	0.2	72.15	75.76	29.43	36.27	3.81
	0.6	0.2	72.15	75.76	29.43	36.27	3.81
	0.7	0.2	72.15	75.76	29.43	36.27	3.81
	0.8	0.2	72.15	75.76	29.43	36.27	3.81
	0.9	0.2	72.15	75.76	29.43	36.27	3.81
	1	0.2	72.15	75.76	29.43	36.27	3.81
	0.4	0.2	72.04	75.69	29.43	36.27	3.84
	0.1	0.2	71.81	75.64	29.43	36.1	3.89
	0	0	66.89	71.85	31.65	39.05	4.57
	0	0.1	66.89	71.85	31.65	39.05	4.57
	0	0.2	66.89	71.85	31.65	39.05	4.57
	0	0.3	66.89	71.85	31.65	39.05	4.57

Table 7.6: *Accuracy* and *Relative accuracy* when classifying Cases 1 and 2, using basic method OVR with several values for th_1 and th_2.

			Case 1		Case 2		
Method	th_1	th_2	ACC (%)	ACC_{REL} (%)	ACC (%)	ACC_{REL} (%)	Score
MULT	0.1	0.3	83.33	81.4	32.28	39.91	2.35
	0.2	0.3	83.33	81.4	32.28	39.91	2.35
	0.3	0.3	83.33	81.4	32.28	39.91	2.35
	0.4	0.3	83.33	81.4	32.28	39.91	2.35
	0.5	0.3	83.33	81.4	32.28	39.91	2.35
	0.6	0.3	83.33	81.4	32.28	39.91	2.35
	0.7	0.3	83.33	81.4	32.28	39.91	2.35
	0.8	0.3	83.33	81.4	32.28	39.91	2.35
	0.9	0.3	83.33	81.4	32.28	39.91	2.35
	1	0.3	83.33	81.4	32.28	39.91	2.35
	0.1	0	92.62	94.3	22.47	25.72	2.49
	0.1	0.6	92.62	94.3	22.47	25.72	2.49
	0.1	0.7	92.62	94.3	22.47	25.72	2.49
	0.1	0.8	92.62	94.3	22.47	25.72	2.49
	0.1	0.9	92.62	94.3	22.47	25.72	2.49

Table 7.7: *Accuracy* and *Relative accuracy* when classifying Cases 1 and 2, using basic method MULT with several values for th_1 and th_2.

Observation 7: *For basic method* OVR*, clearly the lowest score is obtained using* $th_1 = 0.1$ *and* $th_2 = 0.1$*. For basic method* MULT*, the lowest scores are obtained when* $th_1 \in \{0.1, \ldots, 1\}$ *and* $th_2 = 0.3$*.*

Therefore, we choose to set $th_1 = 0.1$, $th_2 = 0.1$ for basic method OVR, and $th_1 = 0.1$, $th_2 = 0.3$ for basic method MULT.

Threshold th_3

Threshold th_3 is used in Extension 3, where the probability is calculated that the activity performed in the current window is equal to the activity performed in the previous window. If this probability is higher than th_3, the current window is classified with the same activity as the previous window. Extension 3 is only executed if the certainty measure is *high*. The certainty measure will for sure stay *high* if th_3 is set to th_c or higher. Therefore, we choose to set th_3 equal to th_c, hence $th_3 = 0.70$.

Smoothing constant γ

To find a proper value to be used for smoothing constant γ, both basic methods are run for Cases 1 and 2, using several values for the smoothing constant.

The results in terms of ACC and ACC_{REL} are given in Tables 7.8 and 7.9, for basic methods OVR and MULT, respectively. To score the results, the scoring method described in Section 7.2 is used.

		Case 1		Case 2		
Method	γ	ACC (%)	ACC_{REL} (%)	ACC (%)	ACC_{REL} (%)	Score
Ovr	0	66.89	71.85	31.65	39.05	5.3925
	0.1	67.34	72.14	32.59	40.33	4.6525
	0.2	68.23	72.76	33.23	41.44	3.9975
	0.3	69.35	73.92	32.91	41.01	3.455
	0.4	70.81	74.91	33.23	41.39	2.8275
	0.5	71.92	76.03	32.59	40.70	2.4425
	0.6	73.15	76.68	31.96	39.83	2.3475
	0.7	74.83	77.75	31.96	39.71	1.69
	0.8	76.96	79.48	30.06	37.10	1.8525
	0.9	77.18	78.39	21.84	27.03	6.6425
	1	50.67	33.33	13.61	20.00	28.35

Table 7.8: *Accuracy* and *Relative accuracy* when classifying Cases 1 and 2, using the Basic Method Ovr and several values for smoothing constant γ.

		Case 1		Case 2		
Method	γ	ACC (%)	ACC_{REL} (%)	ACC (%)	ACC_{REL} (%)	Score
Mult	0	75.95	77.68	31.65	38.87	6.505
	0.1	77.40	78.70	32.60	40.44	5.2575
	0.2	79.87	80.58	32.59	40.56	4.1425
	0.3	81.10	81.78	31.33	38.91	4.2625
	0.4	82.55	82.96	30.70	38.23	3.9325
	0.5	83.33	83.51	30.38	37.76	3.7975
	0.6	84.68	84.55	30.06	37.38	3.375
	0.7	86.80	86.54	28.48	35.35	3.25
	0.8	87.92	86.85	26.58	33.00	3.955
	0.9	90.16	86.63	13.61	16.49	10.82
	1	50.67	33.33	0	0	41.5425

Table 7.9: *Accuracy* and *Relative accuracy* when classifying Cases 1 and 2, using the Basic Method Mult and several values for smoothing constant γ.

Observation 8: *For both* OVR *and* MULT*, the lowest score is obtained when the smoothing constant is equal to 0.7.*

Therefore, smoothing constant γ is set to 0.7.

7.6 Parameter overview

In the previous sections, the following list of parameters is determined, to be used in combination with the two basic methods:

- Window length used for gravitational acceleration estimation: 5 seconds.
- Window length used for classification: 10 seconds.
- Overlap between windows used for classification: none.
- Certainty threshold th_c: 0.70.
- Thresholds th_1 and th_2 (Extension 1):
  - For basic method OVR: $th_1 = 0.1$, $th_2 = 0.1$.
  - For basic method MULT: $th_1 = 0.1$, $th_2 = 0.3$.
- Threshold th_3 (Extension 3): 0.7.
- Smoothing constant γ (Extension 4): 0.7.

7.7 Feature selection

As explained in Chapter 4, a selection of features is made to be used in the classification model by making use of L1 regularised logistic regression. Using a window length of 10 seconds, coefficients are trained by making use of training data, see Section 4.4. The final selection of features is the following:

- linear acceleration magnitude c: energy.
- horizontal linear acceleration l_v: variance, maximum, minimum, energy.
- vertical linear acceleration l_h: energy.
- angle α: mean, variance, minimum, root mean square, energy.
- angle speed v: variance, maximum, minimum, median, root mean square, energy.
- linear acceleration l: Signal Magnitude Area.

Hence, in total there are 18 features used in the classification model.

Observation 9: *All 5 vectors computed in the preprocessing step are used in the final model. However, the majority of features is related to the angle and the angle speed. Also, the energy of all 5 vectors for which it is calculated is included in the feature selection.*

7.8 Results obtained with the classification model

The classification model starts with measuring accelerometer data, preprocessing and calculating features. This process is described in Chapters 3 and 4.

In Figure 7.5, the measured acceleration in three directions and the variance of the magnitude of the linear acceleration for a window of 10 seconds, which is one of the calculated features, are given for Case 1. The figures regarding Case 2 - 5 are placed in Appendix B, Figures B.1 to B.4.

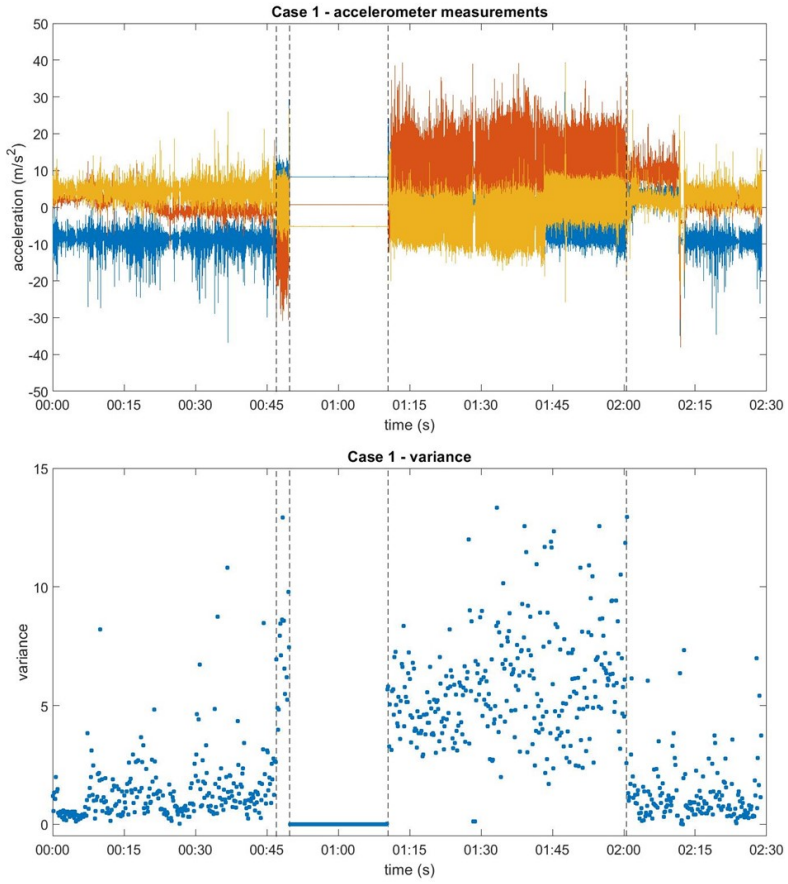


Figure 7.5: *Acceleration* in three directions as measured by the accelerometer and *variance* of the magnitude of the linear acceleration for a window of 10 seconds for Case 1.

Observation 10: *A clear difference is visible between movements belonging to different classes. This is also the case for the other Cases, however, the difference is less visible for motorised vehicles. Case 5 contains the same movements as Case 1, However, as said, the data is collected by two different persons. This is also very clearly visible in the measured accelerometer data and the corresponding variances.*

By simply looking at the variances for the 5 cases, it is at least imaginable that a classification model should be able to make (some) distinction between the six classes.

After calculating features values, classification is done as described in Chapter 5. Since there are two basic methods and four extensions available to form the classification model, a total of 32 different combinations can be made. To asses the quality of the classifications done with all combinations, three measurements are used.

- ACC, see Equation (7.1),
- ACC_{REL}, see Equation (7.2),
- TIME_{high}, the fraction of windows that is correctly classified with *high* certainty,

$$\text{TIME}_{high} = \frac{\#\text{ windows correctly classified with } high \text{ certainty}}{\text{total } \#\text{ windows}} \cdot 100\%.$$

The results of running Cases 1 to 5 with all possible combinations of basic methods and extensions,

using above three measures to asses the quality of the classification, are placed in Appendix C, Table C.1 to Table C.5.

Observation 11: *The results for Case 5 do not match the results for Case 1, despite the fact that the same movements are included. This is due to the fact that the data for Case 5 is collected by a different person than the data for training the model. Therefore, Case 5 will not be considered in the following analysis of the results.*

Looking at the results for Cases 1 to 4, the following things stand out.

- Basic method OVR:
  - Extension 1 can be used to classify *still*, *walk* and *bicycle*. However, using it to classify *car*, *bus* and *train* reduces the quality of the classification.
  - Extension 2 and 3 do not improve or deteriorate the quality of the results.
  - Extension 4 should only be used if Extension 1 is not used.
- Basic method MULT:
  - Extension 1 should be used to classify *still*, *walk*, *bicycle* and *car*. However, using it for classifying *bus* and *train* should only be done if Extension 4 is not used.
  - Extension 2 sometimes improves the quality of the classification, it never makes it worse.
  - Extension 3 should be used to classify *bus* and *train*. When classifying *car*, Extension 3 should only be used in combination with Extension 4. Extension 3 should only be used to classify *still*, *walk* and *bicycle* in combination with Extension 1 or Extensions 1 and 4.
  - Extension 4 only improves the quality of the classification for *still*, *walking* and *cycling*.

Based on the results and the above remarks, we expect the best combinations of basic methods and extensions to be as shown in Table 7.10. It should be noted that different classes can have a different optimal combination of basic method and extensions.

	Still, Walk, Bicycle	Bus, Train	Car
OVR	1	4	4
MULT	1,2,3,4	1,2,3	1,2,3,4

Table 7.10: Expected best combinations of basic methods and extensions when classifying movements belonging to one of the six mentioned classes.

In Table 7.11, the combinations of basic methods and extensions that obtained the overall best results for the five cases are shown.

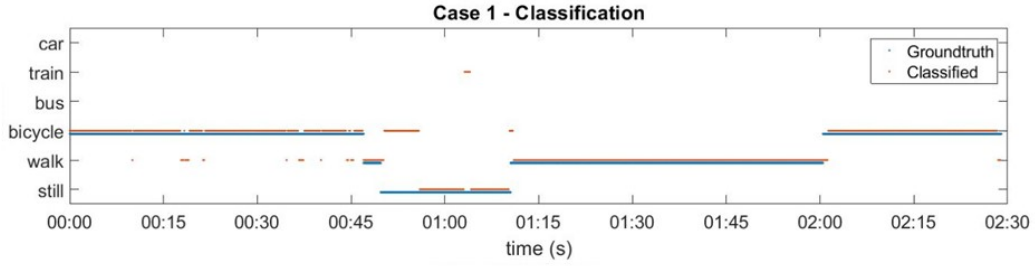
Observation 12: *When comparing the underlying movements present in the cases with the expectations listed in Table 7.10, it must be noted that the conclusions agree with each other.*

	OVR	MULT	Ext 1	Ext 2	Ext 3	Ext 4
Case 1		•	•	•	•	•
Case 2		•	•	•	•	
Case 3		•	•	•	•	•
Case 4	•		•			
Case 5		•	•			

Table 7.11: The combinations of basic methods and extensions that result into the best classification, per Case.

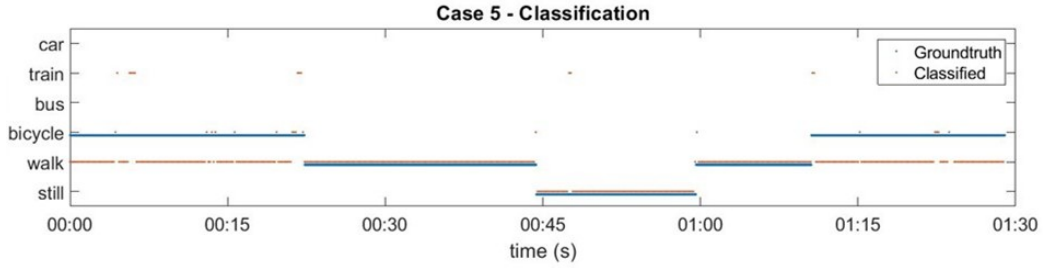
In Figures 7.6 to 7.10, the results are visualised for running Cases 1 to 5 with the combination of basic methods and extensions as given in Table 7.11. The blue lines are the ground truths. For each window of 10 seconds, a red dot is placed by the class with which the window is classified. Together with the plots, *confusion matrices* are given for each case. In a confusion matrix, on the horizontal dimension, the ground truth is given, while on the vertical dimension the prediction made by the classification model is given. Shortly said, a confusion matrix shows the ways in which the classification model is confused when it makes predictions.

Cases 1 and 5 are based on the same movements, however, the data is collected by two different persons carrying the same smartphone.



		Predicted				
		still	walking	cycling	train	
Actual	still	80	3	35	6	65%
	walking	0	314	3	0	99%
	cycling	0	30	432	0	94%

Figure 7.6: Classification and confusion matrix for Case 1, using basic method MULT and all four extensions.



		Predicted				
		still	walking	cycling	train	
Actual	still	88	1	0	2	97%
	walking	0	197	2	0	99%
	cycling	0	217	15	12	6%

Figure 7.7: Classification and confusion matrix for Case 5, using basic method MULT and extension 4.

Observation 13: *For Case 1, most windows are correctly classified. The only part where the classification model goes wrong for a long time, is the first part of* still.

This is most likely caused by one of the extensions, since this behaviour is not included in all model outputs.

Observation 14: *Looking at the output belonging to Case 5, it stands out that almost all windows belonging to* bicycle *are classified as walk.*

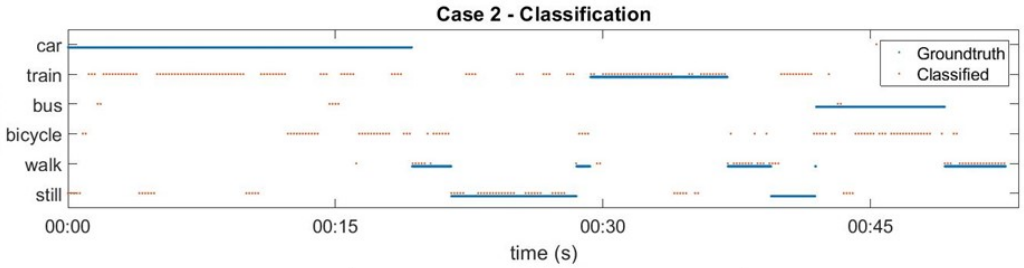
This is most likely caused by the fact that the data is collected by another person. This person probably moves his legs faster when cycling, and therefore causes the model to think that the correct movement is *walking*.

Observation 15: *The output of Cases 2 and 3 shows that the model is not doing well at classifying movements belonging to the class* car. *For Case 2, those movements are mostly classified as* still, train *or* bicycle, *and for Case 3, a lot of windows with ground truth* car *are classified as* still.

The later is probably partly caused by the traffic jam, which took place from approximately 00:45 to 01:00. Also, steady driving with little breaking and turning can cause the model to think that the movement belongs to *still* instead of *car*.

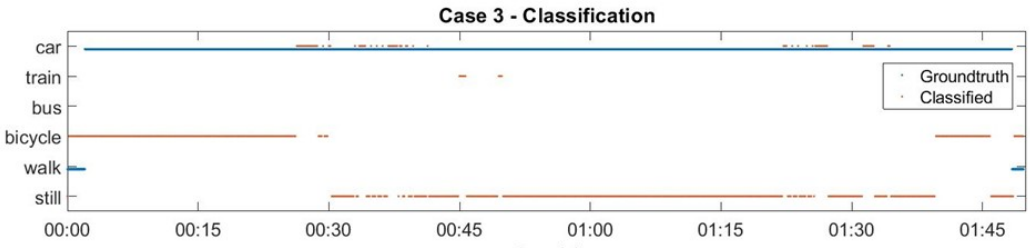
Observation 16: *Looking at the output of Case 2, it stands out that* train *can be classified using the classification model. Movements belonging to the class* bus, *however, are very badly classified.*

Perhaps this is the case since *bus* is a lot more alike with *car* than with *train*.



		Predicted						
		still	walking	cycling	bus	train	car	
Actual	still	29	4	1	0	24	0	50%
	walking	0	37	16	0	0	0	70%
	bus	4	0	31	2	1	5	72%
	train	7	2	0	0	37	0	80%
	car	16	1	27	6	66	0	0%

Figure 7.8: Classification and confusion matrix for Case 2, using basic method MULT and Extensions 1, 2 and 3.



		Predicted					
		still	walking	cycling	bus	train	
Actual	walking	2	0	19	0	0	0%
	car	375	0	189	8	66	10%

Figure 7.9: Classification and confusion matrix for Case 3, using basic method MULT and all four extensions.

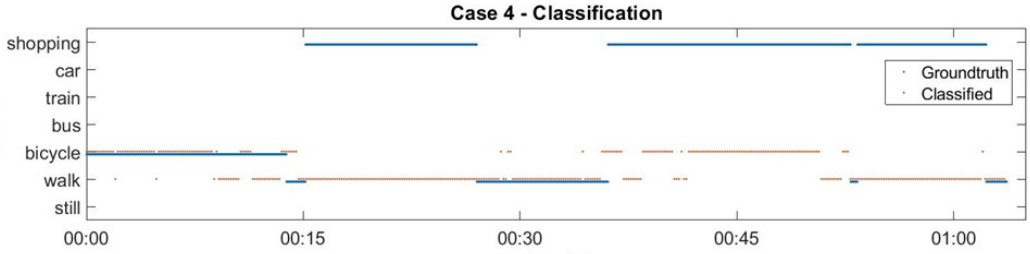
Case 4 is the case that includes a movement that is not part of the six classes used in the classification model, namely *shopping*.

Observation 17: *All movements belonging to shopping are classified as* walking *and* cycling.

A reason for this is that slowly walking, which is common to do in some shops, probably is easily seen as cycling.

Observation 18: *The model does not identify* shopping *as* bus, train *or* car.

Hence, shopping is most likely more alike with non-motorised movements such as *walking* and *cycling* than with motorised movements such as *bus, train* and *car*.



		Predicted			
		walking	cycling	shopping	
Actual	walking	61	12	0	84%
	cycling	24	59	0	71%
	shopping	147	79	0	0%

Figure 7.10: Classification and confusion matrix for Case 4, using basic method OVR and Extension 4.

The final model used for classification, should exist out of one basic method combined with a selection of extensions. Using basic method MULT and all four extensions yields an average accuracy of 50.0%, which is on average the best result obtained.

7.9 Computation times

One requirement for the model was that online classification must be possible, hence, data collection and classification must be done real time. Real time classification is only possible if classifying a window costs less time than filling the next window for classification, which in our case takes 10 seconds. If this would not be the case, windows ready to be classified will pile up, resulting in a non-real time classification that can only be finished when the collecting of data stops. Therefore, it must be known how long it takes on average to classify one window of data.

Table 7.12 shows the results for running the 5 cases with their optimal classification model settings, which can be found in Table 7.11.

Observation 19: *As becomes clear, the average time it takes to classify a window is 0.0101 seconds, which is far below the window length of 10 seconds. Also, the mean variance is 0.000097, which implies that with high probability it will never happen that it takes longer than 10 seconds to classify a window.*

Therefore, we believe that it is safe to say that online classification is possible with the suggested classification model.

The above results are obtained by running the classification model on a laptop with 8 GB RAM and a Intel(R) Core(TM) i5-5200U CPU 2.20GHz. It seems a bit odd to discuss computation times for running the model on a laptop, since a requirement for the model is that it must be able to run on a smartphone. However, since nowadays smartphones are (almost) as fast as laptops in terms of

	Mean duration (sec)	Variance
Case 1	0.0086	0.000061
Case 2	0.0108	0.000175
Case 3	0.0107	0.000067
Case 4	0.0121	0.000121
Case 5	0.00853	0.000063
Mean	0.0101	0.000097

Table 7.12: Computation times for classifying one window with a length of 10 seconds.

computing, we think it is valid to belief that the computational times shown in Table 7.12 can also (or at least almost) be obtained when running the model on a smartphone.

It is (almost) not possible to compute the amount of memory used by the algorithm. This is due to the fact that the model is currently programmed in MATLAB, and will eventually be programmed on a smartphone, where MATLAB is not available. Therefore, most likely, the programming code will change, resulting in a different use of memory. However, since no complex algorithms are used, we expect the model not to blow up and therefore always use a reasonable amount of memory, which can be easily handled by a smartphone.

Chapter 8

Conclusions and recommendations

This thesis presents a first attempt to classify travel modes using accelerometer data collected with a smartphone. As can be concluded from this report, classifying travel modes with accelerometer data is generally possible, however, more development is needed in order to get accurate classification results.

Requirements for the classification model
The classification model is constructed according to the requirements set by Mobidot, as stated in the problem description. The requirements were the following.

1. Accelerometer based.
2. Battery-efficient.
3. Expandable.
4. Classify during trips only.
5. Online classification.

(1) Clearly, the first requirement is met by definition.

(2) It cannot be said with certainty that the battery-efficiency has been obtained. However, during the development of the model and while determining parameter values, it has been taken into account that choices must be made in such a way that battery capacity is not used unnecessary. For instance, a selection of feature values is made instead of using all features. Also, window lengths are not set to 5 seconds, due to battery saving considerations. To be entirely sure about the battery-efficiency, however, it is necessary to run the model on a smartphone and perform some additional tests.

(3) By the way the model is set up, it is made sure that it is not hard to add more classes to the list. When adding an extra class, however, new data has to be collected and coefficients have to be trained again by using L1 regularised logistic regression. This can also cause that the selection of features changes and more or less features are used in the final classification model. Thereby, the running time and computational complexity of the model will most certainly change.
It is also possible to add features to the feature selection. Coefficients for logistic regression should be determined again if this is done.

(4) When running the model on the cases, the fourth requirement is also met by definition.

(5) It is not entirely certain whether the model will work on a smartphone and if online classification is indeed possible. However, it is expected to be the case. Since no machine learning model has been used for preprocessing, feature calculation, and classifying with the basic methods and the extensions, it most certainly is possible to program the model on a smartphone. Also, as shown in Section 7.9, the model takes slightly over 0.01 seconds to classify a window of 10 seconds of

accelerometer data. It must be noted that this result is achieved using a laptop, however, nowadays, most smartphones are nearly as fast as a laptop or even faster. Therefore, we expect the model to be able to run on a smartphone and classify windows in a reasonable time. It remains uncertain how much memory and RAM will be needed to run the model on a smartphone.
The model is created in such a way that online classification is possible. However, when running the model on a smartphone while continually collecting data, we need to know when to start classifying and when to stop. Therefore, a boolean has to be included to the model, which is *true* if the phone is changing position with respect to GPS-coordinates and *false* if this is not the case. Such a feature is already part of the app constructed by Mobidot used to gather data, hence it will not be a big deal to include it into the model. The boolean could also be based on detection of walking by the classification model. However, therefore, continuous classification is needed. To know for sure if online classification is possible, the model must be implemented on a smartphone and tested.

Conclusion 1: *The proposed classification algorithm appears to be a feasible approach for online classifying of travel modes on a smartphone.*

Estimation of the gravitational acceleration
We believe that the current used method for estimating gravitational acceleration is suitable and the highest achievable when using accelerometer data collected at 5Hz. A better result can be obtained with decreasing the length of windows used for estimating gravitational acceleration in combination with increasing the sample rate. This allows the estimate to better follow the data, while the estimate is still based on enough data points, making sure the estimate is not influenced by noise.

It would be even better to include measurements of the gyroscope into the model, to obtain more accurate gravitational acceleration estimates. With the aid of a gyroscope, the position of the smartphone relative to the earth can be determined, and therefore, it can be used to find the exact direction of the gravitational acceleration.

We believe it is needed to improve the estimation of gravitational acceleration, as it might help with obtaining a better classification. Looking at the obtained results, it strikes that motorised travel modes are often confused with movements belonging to the class *still*. This can be explained by the fact that differences in acceleration are often very subtle for those movements. For example, when driving on the highway with a constant speed, the acceleration will not change drastically, but stay nearly zero. Improving the gravitational acceleration estimate will prevent that to much of the accelerometer signal is removed when changing the directions of the measurements. It must be investigated whether this ensures a more clear difference between motorised vehicles and movements belonging to the class *still*.
Again, of course, increasing the sampling frequency can also help in separating *still* from motorised movement.

Conclusion 2: *We believe the current used method for estimating gravitational acceleration is suitable. However, the sample rate for collecting data should be increased to obtain more accurate estimations, resulting in a better classification of motorised vehicles.*

Logistic Regression: feature selection
The feature selection done in this thesis is based on L1 regularised Logistic Regression. A selection of 18 out of 56 features is made and used in the classification model. Since the feature selection is done offline, obtaining the lowest computation times is not necessary. However, to improve running time, it is better to avoid having highly correlated features, such as *mean* and *median*, in the original list of features. Removing one of those two features did not improve the classification result, however, the running time decreased and the numerical stability increased.

Conclusion 3: *We believe that L1 regularised Logistic Regression is a suitable method for selecting features. Using highly correlated features should be avoided, to decrease computation times and increase numerical stability.*

Logistic Regression: travel mode classification
When developing the classification model, we chose to use Logistic Regression for classification purposes. We found that an overall accuracy of 50.0% can be obtained using basic method MULT and all four extensions. However, this combination of basic method and extensions is not optimal for all used test cases. Therefore, we conclude that it can be necessary to chose a non-optimal combination of basic method and extensions, to get on average the best classification.

We believe classification can be improved by increasing the sample rate and by training the model with data collected by more than one person. Also using data from other smartphone sensors can help in obtaining a higher accuracy.

Generally, the model separates movements belonging to classes *walking* and *cycling* very well. However, it is striking that whenever *walking* and *cycling* are misclassified, they are often classified as each other. We believe this is caused by the fact that slowly walking might look like walking, and fast cycling might look like walking. Increasing the sample rate might help to improve on this separation, since then the whole movement is easier to follow.

Increasing the sample rate might also help in distinguishing between motorised movement. Due to the fact that no repeated movement is included in motorised movement, it is important that even smaller changes in acceleration are noticed. As can be seen in the results, the model is able to correctly classify movements in the class *train*, however, classifying *car* and *bus* often goes wrong. For the model to be able to separate better between *car* and *bus*, we believe it may help to see all small changes in acceleration, and hence it might help to increase the sampling frequency.
Another thing that might help with better distinguishing between classes such as *car*, *train* and *bus*, is making use of other available smartphone sensors to collect additional data as input for the classification model. For instance, a Bluetooth-sensor can be used to find if many other Bluetooth-devices are nearby for a longer time. If this is the case, most likely the smartphone is in public transport. Other sensors, such as the gyroscope, can also help with estimating the gravitational acceleration, as said.

A thing to take in mind when using multiple sensors, is the fact that the model must be suited to run on all smartphones. Some older smartphones might not include all additional sensors, or access to the sensor might be denied by the operating system. For instance, it is way harder to get access to all kind of sensors when using iOS than when using Android. When using data from additional sensors for the classification model, this should be taken into account.

As can be seen from the model, Logistic Regression works best on classifying test data collected by the same person as the one that collected the training data. This applies mainly to the classes *walking* and *cycling*. However, it is impossible to have training data for every person who's data needs to be classified. We believe the quality of the classification will improve if the model is trained on data collected by more than one person, making the training data more representative to all kind of users who's data needs to be classified.

Conclusion 4: *We believe that Logistic Regression is a suitable method for both feature selection and classification, however, further development is needed to obtain a higher accuracy. Increasing the sample rate for data collection might help to improve the results for classification, as well as training the model with data collected by more than one person, and using data collected with multiple sensors.*

Data collection on a smartphone
The fact that the data is collected with a smartphone, may cause some trouble. During non-motorised travel modes such as cycling, the position of the phone influences the collection of data. For instance, it makes a huge difference whether the phone is in a pocket or in a bag. Cycling with a smartphone in the pocket of your pants or cycling with the smartphone in a bag on the handlebars makes a huge difference for the obtained signal. However, both ways of cycling were included in the training and testing data for our model, and both are classified as cycling as well. Hence, we believe that considering all possible positions of the smartphone when collecting training data, will solve the above issue.

Also, influences on the measurements from holding the smartphone while in motorised vehicles, are almost impossible to deal with, and influence the measurements in such a way that recognising the motorised movement is not possible anymore. Perhaps making a class *user interaction with smartphone* will solve this issue.

Lastly, the unit of acceleration measured by smartphones can differ per smartphone brand. Some accelerometers measure with unit m/s^2, others measure with unit g, where 1 g = 9.80665 m/s^2. Hence, in some cases an extra preprocessing step is necessary, where the unit of the accelerometer measurements is changed to m/s^2. Instead of preprocessing, it is also possible to make the model self learning, where parameter values are learned from the data.

Conclusion 5: *It is possible to collect accelerometer data with a smartphone for classification purposes. A way should be found to deal with external influences on the signal, made by the wearer of the smartphone.*

Adding more travel modes

Currently, six classes are included in the classification model. However, way more travel modes are possible. For instance, using a step or a tram or a metro could be possible other ways of travelling. The use of overarching classes, where for instance all vehicles driving on a track are included in one class, should be further investigated. If this proves to work, more travel modes can be included into the model, without extending the classification model to much, making it to complex to run on a smartphone.

Conclusion 6: *We believe further research is needed on extending the classes taken into account by the classification model, in order to be able to classify more travel modes.*

Bibliography

[1] CBS, "Onderweg in nederland 2018," Centraal Bureau voor de Statistiek, Tech. Rep., 2019.

[2] Y. Hanai, J. Nishimura, and T. Kuroda, "Haar-like filtering for human activity recognition using 3d accelerometer," in *2009 IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop*. IEEE, 2009, pp. 675–678.

[3] J. Kivit, "Smartphone sensoren: hét complete overzicht van alle mogelijkheden," 2017, accessed: 2020-07-13. [Online]. Available: https://www.shareforce.nl/nl/blog/smartphone-sensoren-h%C3%A9t-complete-overzicht-van-alle-mogelijkheden

[4] M. Li, J. Dai, S. Sahu, and M. Naphade, "Trip analyzer through smartphone apps," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2011, pp. 537–540.

[5] X. Zhou, W. Yu, and W. C. Sullivan, "Making pervasive sensing possible: Effective travel mode sensing based on smartphones," *Computers, Environment and Urban Systems*, vol. 58, pp. 52–59, 2016.

[6] P. Widhalm, P. Nitsche, and N. Brändie, "Transport mode detection with realistic smartphone sensor data," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 573–576.

[7] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proceedings of the 11th ACM conference on embedded networked sensor systems*, 2013, pp. 1–14.

[8] J. Urbancic, V. Pejovic, and D. Mladenic, "Transportation mode detection using random forest," *Information Society, Data Mining and Data Warehouses SiKDD, Ljubljana, Slovenia*, 2018.

[9] H. Su, H. Caceres, H. Tong, and Q. He, "Travel mode identification with smartphones," *Sensors*, vol. 15, no. 16, pp. 4871–4889, 2015.

[10] A. Efthymiou, E. N. Barmpounakis, D. Efthymiou, and E. I. Vlahogianni, "Transportation mode detection from low-power smartphone sensors using tree-based ensembles," *Journal of Big Data Analytics in Transportation*, vol. 1, no. 1, pp. 57–69, 2019.

[11] X. Su, H. Caceres, H. Tong, and Q. He, "Online travel mode identification using smartphones with battery saving considerations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, pp. 2921–2934, 2016.

[12] X. Su, "Travel mode identification with smartphone sensors," Ph.D. dissertation, The City University of New York, 2017.

[13] T. Feng and H. J. Timmermans, "Transportation mode recognition using gps and accelerometer data," *Transportation Research Part C: Emerging Technologies*, vol. 37, pp. 118–130, 2013.

[14] Y. S. Lee and S. B. Cho, "Human activity inference using hierarchical bayesian network in mobile contexts," in *International Conference on Neural Information Processing*. Springer, 2011, pp. 38–45.

[15] M. Kose, O. D. Incel, and C. Ersoy, "Online human activity recognition on smart phones," in *Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*, vol. 16, 2012, pp. 11–15.

[16] Y. Maruno, K. Cho, Y. Okamoto, H. Setoguchi, and K. Ikeda, "An online human activity recognizer for mobile phones with accelerometer," in *International Conference on Neural Information Processing*. Springer, 2011, pp. 358–365.

[17] D. Shin, D. Aliaga, B. Tunçer, S. M. Arisona, S. Kim, D. Zünd, and G. Schmitt, "Urban sensing: Using smartphones for transportation mode classification," *Computers, Environment and Urban Systems*, vol. 53, pp. 76–86, 2015.

[18] Y. Lu, Y. Wei, L. Liu, J. Zhong, L. Sun, and Y. Liu, "Towards unsupervised physical activity recognition using smartphone accelerometers," *Multimedia Tools and Applications*, vol. 76, no. 8, pp. 10 701–10 719, 2017.

[19] N. L. Dang, "Mobile online activity recognition system-based on smartphone sensors," *Journal of Intelligent Computing Volume*, vol. 8, no. 1, p. 17, 2017.

[20] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors*, vol. 14, no. 6, pp. 10 146–10 176, 2014.

[21] X. Su, H. Tong, and P. Ji, "Activity recognition with smartphone sensors," *Tsinghua science and technology*, vol. 19, no. 3, pp. 235–249, 2014.

[22] Mobidot, "Over mobidot," 2019, accessed: 2020-07-28. [Online]. Available: mobidot.nl/over_mobidot.php

[23] mathworks, "Sensor data collection with matlab mobile," 2020, accessed: 2020-07-15. [Online]. Available: https://nl.mathworks.com/help/matlabmobile/ug/sensor-data-collection-with-matlab-mobile.html

[24] K. R. Christiansen and A. Shalamov, "Motion sensors explainer," 2017, accessed: 2020-07-24. [Online]. Available: https://www.w3.org/TR/2017/NOTE-motion-sensors-20170830/

[25] D. Mizell, "Using gravity to estimate accelerometer orientation," in *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.* Citeseer, 2003, pp. 252–253.

[26] A. Pownuk and V. Kreinovich, "Why linear interpolation?" *Mathematical Structures and Modeling*, 2017.

[27] M. Cerna and A. F. Harvey, "The fundamentals of fft-based signal analysis and measurement," Application Note 041, National Instruments, Tech. Rep., 2000.

[28] A. M. Khan, Y. K. Lee, S. Y. Lee, and T. S. Kim, "A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer," *IEEE transactions on information technology in biomedicine*, vol. 14, no. 5, pp. 1166–1172, 2010.

[29] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[30] Statistic solutions, "Assumptions of logistic regression," 2020, accessed: 2020-07-17. [Online]. Available: https://www.statisticssolutions.com/assumptions-of-logistic-regression/

[31] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 78.

[32] E. Ostertagová and O. Ostertag, "The simple exponential smoothing model," in *The 4th International Conference on Modelling of Mechanical and Mechatronic Systems, Technical University of Košice, Slovak Republic, Proceedings of conference*, 2011, pp. 380–384.

Appendix A

List of symbols

Symbol	Unit	Description
α	radians	angle between $\boldsymbol{l}$ and $\boldsymbol{g}$
A_w	-	matrix with accelerometer measurements, used for preprocessing data
$\boldsymbol{a} = (a_x, a_y, a_z)$	m/s^2	accelerometer measurements measured by a triaxial accelerometer, in directions x, y and z. For specific directions, see Figure 3.1
$\beta_0, \beta_1, \boldsymbol{\beta}$	-	coefficients used in logistic regression
$\gamma \in [0, 1]$	-	smoothing constant
c	m/s^2	magnitude of linear acceleration
$C_w \in [0, 1]$	-	certainty measure for window w
$\boldsymbol{g} = (g_x, g_y, g_z)$	m/s^2	gravitational acceleration estimate
K	-	set of activities used in the classification model
λ	-	regularisation parameter
$\boldsymbol{l} = (l_x, l_y, l_z)$	m/s^2	linear acceleration
l_h	-	horizontal linear acceleration
l_v	-	vertical linear acceleration
$l(\boldsymbol{\beta})$	-	log-likelihood
n	-	total number of features calculated
m	-	total number of grid points for interpolations
n_w	-	number of measurements in window w
p	-	probability to belong to a certain class
$p_w \in [0, 1]$	-	probability related to the class with which window w is classified
s	seconds	length of windows used for classification
S	-	set containing training data
t	seconds	time stamp of accelerometer measurements
$th_1, th_2 \in [0, 1]$	-	thresholds used in Extension 1
$th_3 \in [0, 1]$	-	threshold used in Extension 3
$th_c \in [0, 1]$	-	threshold used to determine when certainty is *high*
v	m/s	angle speed
w	-	window of length s with accelerometer measurements
$\boldsymbol{x} = \{x_1, \ldots, x_n\}$	-	feature values
$y^{(i)}$	-	class label of i-th training sample
ACC, ACC$_{REL}$, TIME$_{high}$	-	measurements to assess the quality of a classification

Appendix B

Accelerometer data and variance

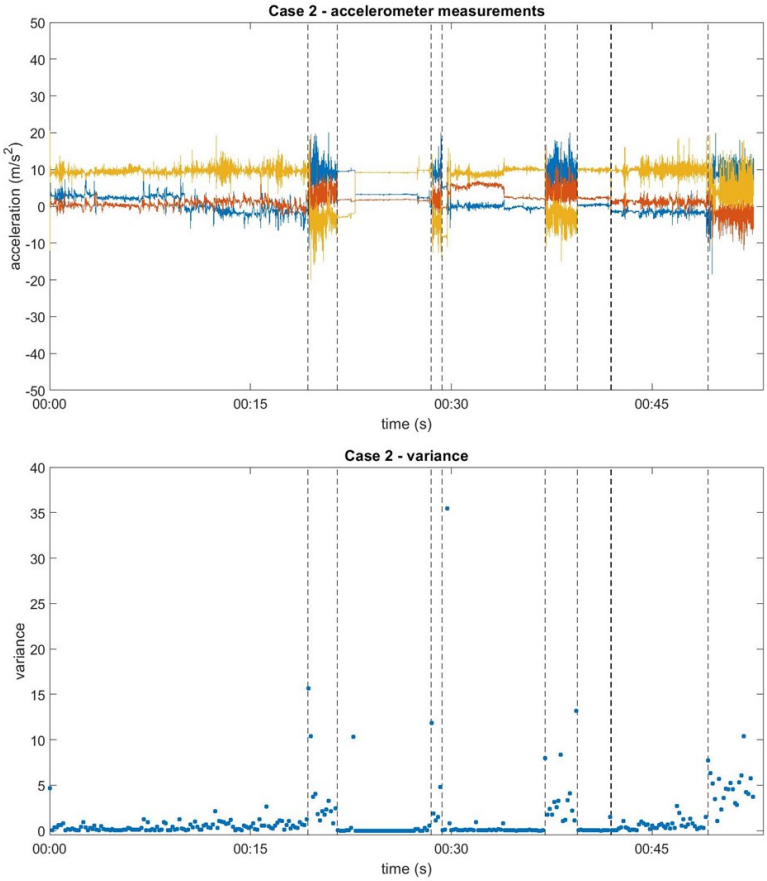


Figure B.1: *Acceleration* in three directions as measured by the accelerometer and *variance* of the magnitude of the linear acceleration for a window of 10 seconds for Case 2.

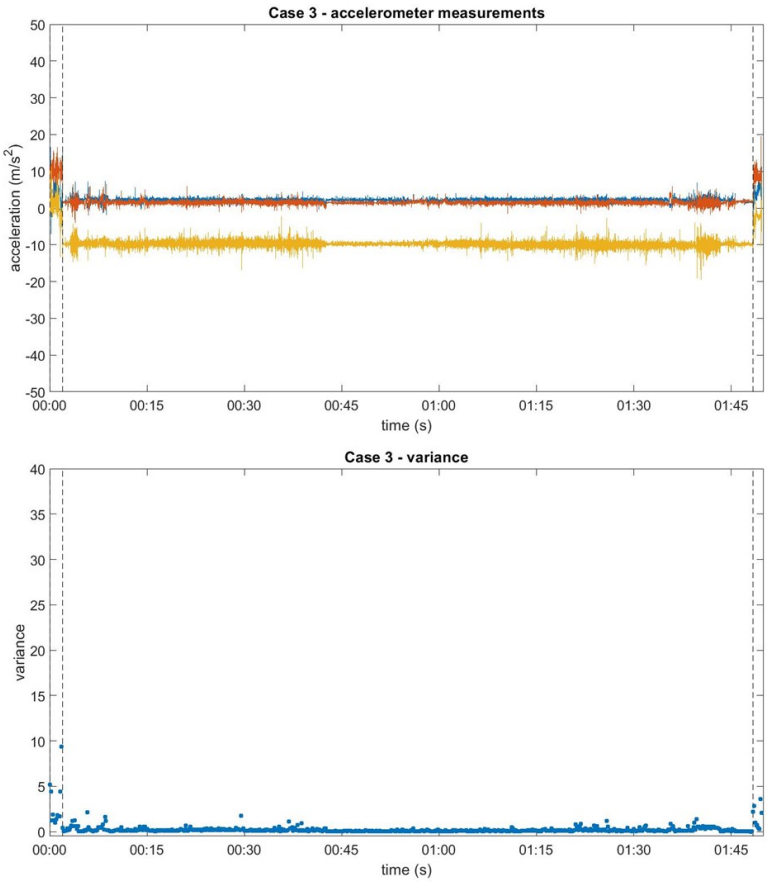


Figure B.2: *Acceleration* in three directions as measured by the accelerometer and *variance* of the magnitude of the linear acceleration for a window of 10 seconds for Case 3.

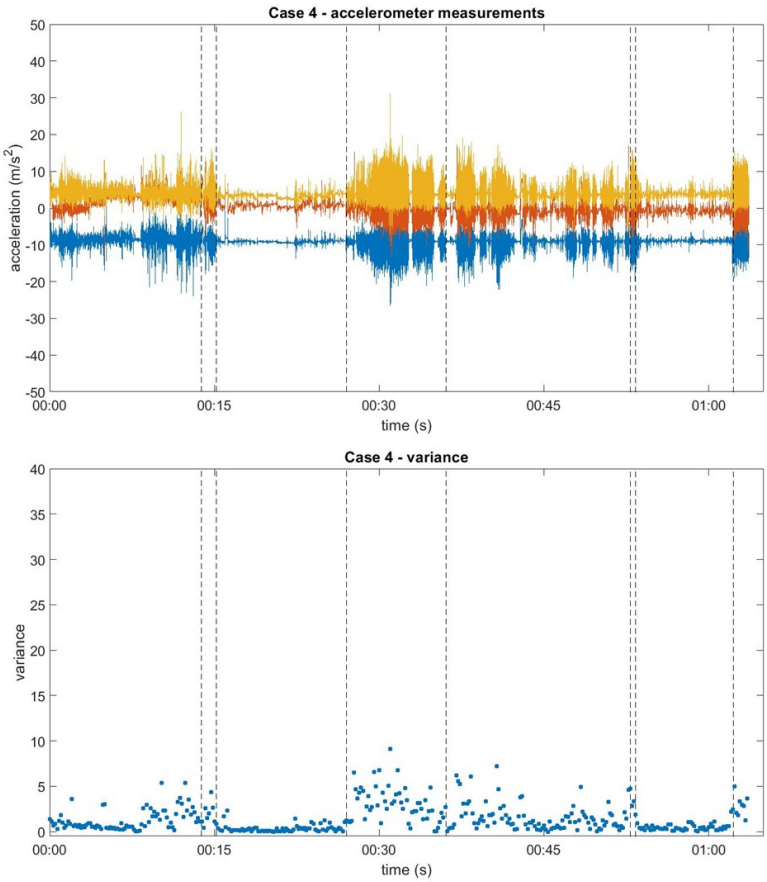


Figure B.3: *Acceleration* in three directions as measured by the accelerometer and *variance* of the magnitude of the linear acceleration for a window of 10 seconds for Case 4.

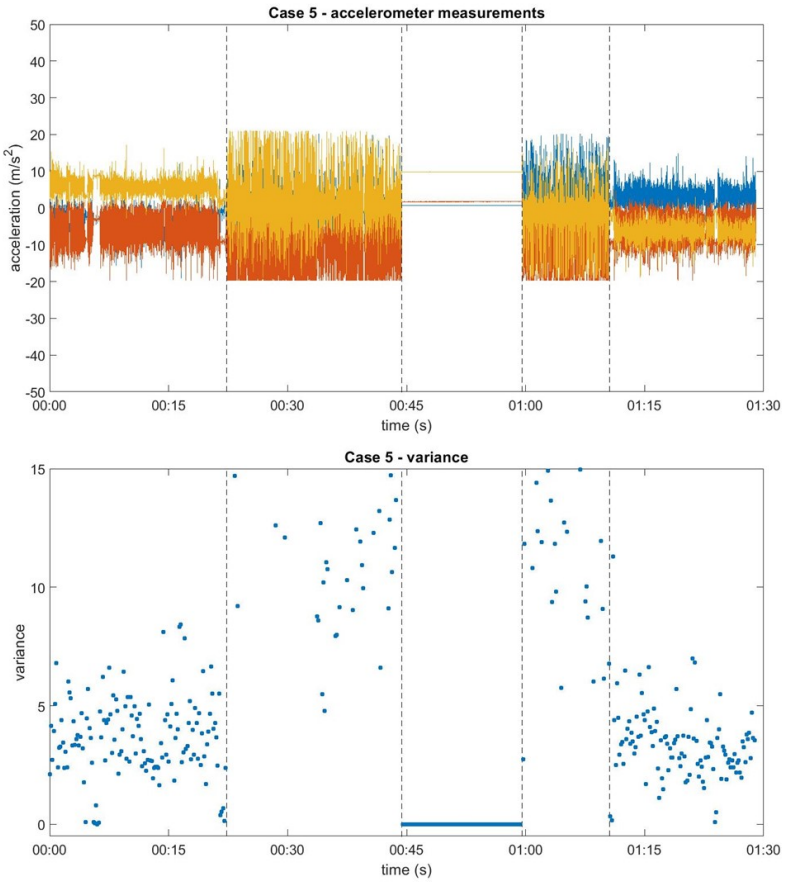


Figure B.4: *Acceleration* in three directions as measured by the accelerometer and *variance* of the magnitude of the linear acceleration for a window of 10 seconds for Case 5.

Appendix C

Results

Ovr	Mult	Ext 1	Ext 2	Ext 3	Ext 4	ACC	ACC_{REL}	$TIME_{high}$
•						66.89	71.846	33.445
•		•				85.011	88.968	33.445
•			•			66.89	71.846	33.445
•		•	•			85.011	88.968	33.445
•				•		66.89	71.846	33.557
•		•		•		85.011	88.968	33.557
•			•	•		66.89	71.846	33.557
•		•	•	•		85.011	88.968	33.557
•					•	74.832	77.75	34.452
•		•			•	80.201	73.637	34.452
•			•		•	74.832	77.75	34.452
•		•	•		•	80.201	73.637	34.452
•				•	•	74.832	77.75	34.452
•		•		•	•	80.201	73.637	34.452
•			•	•	•	74.832	77.75	34.452
•		•	•	•	•	80.201	73.637	34.452
	•					75.951	77.68	32.886
	•	•				83.445	83.195	34.564
	•		•			75.951	77.68	32.886
	•	•	•			83.557	83.269	35.011
	•			•		76.174	77.921	33.669
	•	•		•		84.228	83.805	36.913
	•		•	•		76.174	77.921	33.669
	•	•	•	•		84.228	83.805	36.577
	•				•	86.801	86.537	34.452
	•	•			•	90.716	85.208	36.577
	•		•		•	86.801	86.537	34.452
	•	•	•		•	90.716	85.208	39.038
	•			•	•	86.689	86.463	34.452
	•	•		•	•	90.94	85.355	40.716
	•		•	•	•	86.689	86.463	34.452
	•	•	•	•	•	91.387	85.649	45.414

Table C.1: Results Case 1

Ovr	Mult	Ext 1	Ext 2	Ext 3	Ext 4	ACC	ACC_{REL}	TIME_{high}
•						31.646	39.052	4.4304
•		•				24.367	27.983	5.6962
•			•			31.646	39.052	4.4304
•		•	•			24.367	27.983	5.6962
•				•		31.646	39.052	4.4304
•		•		•		24.367	27.983	5.6962
•			•	•		31.646	39.052	4.4304
•		•	•	•		24.367	27.983	5.6962
•					•	31.962	39.707	6.3291
•		•			•	22.152	25.602	6.3291
•			•		•	31.962	39.707	6.3291
•		•	•		•	22.152	25.602	6.3291
•				•	•	31.962	39.707	6.3291
•		•		•	•	22.152	25.602	6.3291
•			•	•	•	31.962	39.707	6.3291
•		•	•	•	•	22.152	25.602	6.3291
	•					31.646	38.874	5.3797
	•	•				31.962	39.748	7.5949
	•		•			31.646	38.874	5.3797
	•	•	•			32.911	40.602	8.2278
	•			•		31.646	38.874	5.6962
	•	•		•		32.278	40.125	7.9114
	•		•	•		31.646	38.874	5.6962
	•	•	•	•		33.228	40.979	8.5443
	•				•	28.481	35.348	5.6962
	•	•			•	24.051	29.774	7.9114
	•		•		•	28.481	35.348	5.6962
	•	•	•		•	24.051	29.774	8.2278
	•			•	•	29.114	36.103	6.962
	•	•		•	•	24.684	30.529	8.8608
	•		•	•	•	29.114	36.103	6.962
	•	•	•	•	•	24.684	30.529	9.1772

Table C.2: Results Case 2

Ovr	Mult	Ext 1	Ext 2	Ext 3	Ext 4	ACC	ACC_{REL}	$TIME_{high}$
•						1.6692	7.7698	0
•		•				0.45524	7.1429	0
•			•			1.6692	7.7698	0
•		•	•			0.45524	7.1429	0
•				•		1.6692	7.7698	0
•		•		•		0.45524	7.1429	0
•			•	•		1.6692	7.7698	0
•		•	•	•		0.45524	7.1429	0
•					•	2.5797	1.3323	0
•		•			•	0	0	0
•			•		•	2.5797	1.3323	0
•		•	•		•	0	0	0
•				•	•	2.5797	1.3323	0
•		•		•	•	0	0	0
•			•	•	•	2.5797	1.3323	0
•		•	•	•	•	0	0	0
	•					6.0698	3.1348	0
	•	•				8.953	4.6238	0
	•		•			6.0698	3.1348	0
	•	•	•			8.1942	4.232	0
	•			•		6.0698	3.1348	0
	•	•		•		8.953	4.6238	0
	•		•	•		6.0698	3.1348	0
	•	•	•	•		8.1942	4.232	0
	•				•	8.953	4.6238	0
	•	•			•	8.953	4.6238	0.91047
	•		•		•	8.953	4.6238	0
	•	•	•		•	9.8634	5.094	0.75873
	•			•	•	8.953	4.6238	0
	•	•		•	•	8.953	4.6238	0.91047
	•		•	•	•	8.953	4.6238	0
	•	•	•	•	•	10.015	5.1724	0.75873

Table C.3: Results Case 3

Ovr	Mult	Ext 1	Ext 2	Ext 3	Ext 4	ACC	ACC_{REL}	TIME_{high}
•						48.077	48.812	8.3333
•		•				76.923	77.323	8.3333
•			•			48.077	48.812	8.3333
•		•	•			76.923	77.323	8.3333
•				•		48.077	48.812	8.3333
•		•		•		76.923	77.323	8.3333
•			•	•		48.077	48.812	8.3333
•		•	•	•		76.923	77.323	8.3333
•					•	64.744	65.217	10.897
•		•			•	74.359	74.253	10.897
•			•		•	64.744	65.217	10.897
•		•	•		•	74.359	74.253	10.897
•				•	•	64.744	65.217	10.897
•		•		•	•	74.359	74.253	10.897
•			•	•	•	64.744	65.217	10.897
•		•	•	•	•	74.359	74.253	10.897
	•					46.154	45.849	8.3333
	•	•				58.974	58.062	8.3333
	•		•			46.154	45.849	8.3333
	•	•	•			58.974	58.062	8.3333
	•			•		46.154	45.849	8.3333
	•	•		•		58.974	58.062	8.3333
	•		•	•		46.154	45.849	8.3333
	•	•	•	•		58.974	58.062	8.3333
	•				•	63.462	62.527	7.6923
	•	•			•	67.949	66.826	16.026
	•		•		•	63.462	62.527	7.6923
	•	•	•		•	67.949	66.826	16.026
	•			•	•	66.667	65.951	16.667
	•	•		•	•	71.154	70.251	25
	•		•	•	•	66.667	65.951	16.667
	•	•	•	•	•	71.154	70.251	25

Table C.4: Results Case 4

Ovr	Mult	Ext 1	Ext 2	Ext 3	Ext 4	ACC	ACC_{REL}	TIME_{high}
•						53.745	64.848	35.768
•		•				54.869	66.847	35.955
•			•			53.745	64.848	35.768
•		•	•			54.869	66.847	35.955
•				•		53.745	64.848	35.955
•		•		•		54.869	66.847	36.142
•			•	•		53.745	64.848	35.955
•		•	•	•		54.869	66.847	36.142
•					•	52.809	63.277	36.142
•		•			•	51.311	61.005	36.142
•			•		•	52.809	63.277	36.142
•		•	•		•	51.311	61.005	36.142
•				•	•	52.809	63.277	36.142
•		•		•	•	51.311	61.005	36.142
•			•	•	•	52.809	63.277	36.142
•		•	•	•	•	51.311	61.005	36.142
	•					55.431	66.046	36.142
	•	•				56.18	67.282	36.33
	•		•			55.431	66.046	36.142
	•	•	•			55.993	67.145	36.517
	•			•		54.12	65.09	36.142
	•	•		•		54.869	66.326	36.33
	•		•	•		54.12	65.09	36.142
	•	•	•	•		54.682	66.189	36.517
	•				•	54.12	63.973	36.33
	•	•			•	53.558	62.874	36.704
	•		•		•	54.12	63.973	36.33
	•	•	•		•	53.558	62.874	37.453
	•			•	•	53.371	63.426	36.33
	•	•		•	•	52.809	62.327	36.704
	•		•	•	•	53.371	63.426	36.33
	•	•	•	•	•	52.809	62.327	37.453

Table C.5: Results Case 5