Vehicle Sensor and Event Detection through Smartphone Acceleration Data

Fahrzeug Sensor und Ereignis Detektion mit Hilfe von Smartphone Beschleunigungsdaten

Master-Thesis von Mike Matthias Smyk aus Offenbach am Main

Juli 2017

TECHNISCHE UNIVERSITÄT DARMSTADT

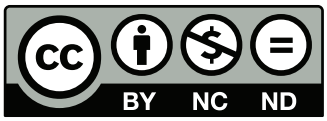
Department of Computer Science
Telecooperation Group

Vehicle Sensor and Event Detection through Smartphone Acceleration Data
Fahrzeug Sensor und Ereignis Detektion mit Hilfe von Smartphone Beschleunigungsdaten

Vorgelegte Master-Thesis von Mike Matthias Smyk aus Offenbach am Main

1. Gutachten: Prof. Dr. Max Mühlhäuser
2. Gutachten: Sebastian Kauschke

Tag der Einreichung:



Erklärung zur Abschlussarbeit gemäß §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Mike Matthias Smyk, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Darmstadt, den 10. Juli, 2017

(Mike Matthias Smyk)

Acknowledgments

I would like to thank my supervisor Sebastian Kauschke and the Telecooperation Group of TU Darmstadt, to provide me the basic conditions for my thesis and for the help during the project.

I would also like to thank Benjamin Ryder from ETH Zürich, for the excellent mentoring and for offering me the opportunity to work on this topic. Thank you for the plenty of your time during our regular calls and for your advices.

Thanks to Advisori FTC GmbH, for providing me financial security during my thesis time and for offering me the possibility to completely concentrate on the thesis, without other worries.

Special thanks to Kai Steinert, Christos Votskos and Lydia Weber for proofreading and for your advices during the project.

Thanks to Kai for spending plenty of time in the "Dungeon" with me and for closing tabs in the right moments. Thanks to Lydia for being there and withstand also bad moods. Thanks to my family, for all the support I got in all my life that has brought me to where I am now.

Last but not least I would like to thank all the people who were involved in my life during this thesis and throughout my whole studies. Thanks for the best time of my life, for years full of positive experience and full of good spirits. I will miss it.

Abstract

Modern vehicles contain sensors tracking the state of the car. The recorded data is used for internal diagnostics and error reporting. Sharing and aggregating this data in a network could increase traffic safety by warning drivers or monitoring road quality. But vehicle sensor data is restricted and not standardized. To overcome restrictions, a smartphone could be used as a proxy from the real world to the network.

In this thesis, vehicle sensor data is used to label acceleration data of a free positioned smartphone. It is investigated, whether vehicle system activations can be estimated with smartphone acceleration data. Events that are identified are Anti-lock Braking System (ABS), Electronic Stability Control (ESC), vehicles acceleration, and heavy braking.

The results show that estimating events in a driving context is hardly achievable. The main problem is to transform acceleration vectors in the vehicle's coordinate system, since the smartphone is freely positioned.

Although, detecting vehicle sensors and driving events through smartphone data is a promising approach to overcome the lack of information resulting from restrictions of the vehicle's sensor communication. Activations can not yet be accurately detected by acceleration data and more research for more accurate sensor readings of the smartphone has to be performed.

Zusammenfassung

In modernen Autos sind viele Sensoren verbaut. Die Daten, die die Sensoren produzieren, werden lediglich für die interne Diagnose und Fehlerauswertung benutzt. Allerdings könnten diese Daten einen Mehrwert für die Vehrkehrssicherheit oder die Straßenqualität bieten. Hierfür müssen die Daten in einem Netzwerk zur Verfügung stehen, um andere Fahrer zu warnen oder Auswertungen durchzuführen. Jedoch stehen die Daten aus dem Fahrzeug nur begrenzt zur Verfügung, da die interne Kommunikation nicht standardisiert ist und die auslesbaren Daten von den Autoherstellern begrenzt wird. Um diese Beschränkung zu umgehen wird ein Smartphone benutzt. Smartphonedaten können in einem Netzwerk zur Verfügung gestellt werden, da die meisten Smartphones bereits mit dem Internet verbunden sind.

In der vorliegenden Arbeit werden aufgenommene Autodaten dazu benutzt, Beschleunigungsdaten eines frei positionierten Smartphones zu kennzeichnen. Es wird untersucht, ob die Aktivierung von Fahrzeugsystemen erkannt wird. Konkret wird untersucht ob erkannt wird, ob das Antiblockiersystem (ABS) oder Elektronische Stabilitätsprogramm (ESP) aktiv waren, wie schnell das Fahrzeug beschleunigt hat und ob ein starkes Bremsmanöver stattgefunden hat.

Das Hauptproblem bei der Detektion der genannten Ereignisse sind die vom Smartphone bereitgestellten Transformationsvektoren. Die Beschleunigungsvektoren müssen zunächst in das Auto-Koordinatensystem transformiert werden. Die Transformationen sind jedoch ungenau und zu selten aktualisiert. Autosensoren und Fahrereignisse mit dem Smartphone zu erkennen ist ein vielversprechender Ansatz, die Restriktionen bei der Datenauslesung zu umgehen. Allerdings können die Sensoren und Ereignisse nicht genau genug erkannt werden, aufgrund der ungenauen Smartphone Sensoren.

Contents

Figures and Tables

List of Figures

List of Tables

1 Introduction

Modern vehicles contain sensors tracking the state of the car. These sensors generate a huge amount of data that is used only for internal diagnostic and error reporting. Studies indicate that this data can be further used for driving behavior predictions (Ruta et al., 2010), determination of street conditions (Mangan et al., 2003) and traffic situation detection (Zheng et al., 2014). Those high-level generated information can be used for early warning systems that can prevent dangerous situations if they are shared. Subsequent drivers can be warned, digital traffic signs can be updated, and in extreme cases emergency services can be pre-warned. Moreover, shared car data enables to maintain statistics, for example, safety system activations point out particularly dangerous locations. Thus, these statistics can be used to improve traffic safety. Vehicle data can also be used for predictive maintenance. Changes in the patterns of the data may indicate upcoming serious problems of the car. Hence, it is important to further investigate usage and the share of vehicle data.

Volkswagen starts to equip the first models with pWLAN (personal Wireless LAN) in 2019 (Volkswagen, 2017). It enables cars to communicate and exchange information. "Examples would include a car making an emergency stop or the on-board sensors detecting black ice. Within a few milliseconds, this information can be shared with the local environment, allowing other road users to react to this risky situation appropriately." (Volkswagen, 2017) pWLAN is based on IEEE 802.11p standard. It is the first approach of a standardized inter-vehicle communication. Since it will be started to be integrated in 2019, it will still take a long time until the prevalence of equipped cars increases. Hence, an approach to utilize today's available data sources is required.

Data in a car is distributed over the CAN bus (Farsi et al., 1999). The CAN bus was specified by Bosch in 1991 to lower the amount of cable in a car (Bosch, 1991). Getting CAN bus data today is expensive. The CAN bus signal is not standardized and, therefore, differently encoded for every make of car or even between models of the same manufacturer. Hence, building a network of connected cars sharing CAN bus information is challenging these days. The network requires additional hardware in all cars and a collaboration of many companies to support this new hardware. Establishing a new standard that enables every new car to be automatically part of the network is cost consuming and would require a lot of time and political effort. Instead, available hardware can be used.

Instead of connecting the cars directly, they can be connected through smartphones that are carried inside the car. Smartphones are widely common devices, are comparably easy to con-

nect, and unite a multiplicity of sensors[1]. Dependent on the model, a smartphone contains environmental sensors (barometer, photometer, and thermometer), position sensors (orientation sensors and magnetometer) and motion sensors (accelerometer, gyroscope, and rotational vector sensors).

Still, the information of the car's system has to be transferred from car to smartphone. Again, additional hardware would be required to get sensor information from the car. Instead, the smartphone could be used as a proxy between car and network. To overcome the need of additional hardware to distribute information about car system activations, sensors of the smartphone are used to detect these activations. Therefore, a system is needed to detect car system activations.

In forefront of this thesis, a data set was gathered by researchers of ETH Zürich and University of St. Gallen (Ryder et al., 2016) in an uncontrolled field study. During the study, drivers were provided with a smartphone and a mount. While driving, data from car (CAN bus) and data from smartphone (accelerometer, gyroscope, magnetometer, Global Positioning System (GPS)) were simultaneously recorded.

The goal of this thesis is to check, whether smartphone acceleration data can be used to recognize specific situations. These situations include Anti-lock Braking System (ABS) or Electronic Stability Control (ESC) activations, heavy braking, and the current real acceleration of the car. The provided data set can be used for labeling the data to train classifiers or regressors, respectively. It is investigated whether it is possible to estimate these events only with smartphone acceleration data.

1.1 Thesis Outline

The thesis is structured as follows: Chapter 2 gives an overview of the current literature on event detection in cars through CAN bus data and smartphone acceleration data. Chapter 3 introduces the terms and concepts used for this thesis. Chapter 4 presents the data that is used for the experiments and explains some challenges occurring with the data. Chapter 5 overviews the pre-processing steps done on the smartphone acceleration data. Chapter 6 covers the experiments done in the project, the experiments are introduced and discussed. Finally, chapter 7 concludes the project and provides an overview of possible ongoing work.

[1] `https://developer.android.com/guide/topics/sensors/sensors_overview.html` (March 25, 2017)

2 Event Detection in a Driving Environment

The previous chapter provides an introduction of the problem tackled in this thesis. This chapter summarizes the current state of research in detecting events in a driving environment.

2.1 Detection Through CAN Bus Data

Since Bosch specified the CAN bus (Bosch, 1991) to lower the amount of cable in a car, a great amount of work has been created to utilize the resulting data.

(Mangan et al., 2003) develop a road slope estimation algorithm using only the CAN bus data. The data is used to estimate the parameters of a car model built by the authors. Using the adapted model, the road slope is computed. Since parameters, such as the *braking coefficient*, are adjusted using *braking tests*, the generality of the model remains unclear.

CAN bus data is also used to provide information on the current status of the driver. (Zaldivar et al., 2011) designed a smartphone application that sends accident details via short message or e-mail or makes an automated phone call to an emergency service. Therefore, the vehicle's velocity is monitored as well as the airbag triggers. The car data is gathered by a Bluetooth On-Board Diagnostics (OBD)-II connector. The acceleration estimation using the car's velocity data is compared to the acceleration estimation using the smartphones GPS data and the accelerometer information, respectively. Acceleration information based on the smartphone's G-force sensor is disregarded. The authors mention, the range of values supported by the sensor is too low to distinguish between accidents and events, such as a falling smartphone. In this thesis, it is assumed that the smartphone is mounted such that noise from handling the phone or other external factors than forces from car is prevented.

(Ruta et al., 2010) combine the smartphone data with OBD-II data, which is gathered by a Wifi connector. Additionally, context information, such as weather condition, is added. The combined data is used to semantically annotate relevant safety events, which are used to provide recommendations for a safer drive. By using a formal language, rules are derived that define how to avoid risks produced by, for example, fog. The system suggests the driver to perform safety measures, such as activating fog lights or reducing speed. Although, the labeling process and the rule derivation are not further stated, the work of Ruta et al. demonstrates how to combine and use several data sources (car, smartphone and web).

There are market-ready OBD-II adapters, such as PACE[1] or Vgate®[2]. These OBD-II adapters are capable to read the current information of a car and send it to a smartphone via Wifi or Bluetooth. This information includes engine speed, vehicle speed, battery voltage, gear, and others. Moreover, trouble-codes are received and analyzed. The PACE adapter also detects crashes and contacts emergency service in case of an accident.

Other start-ups perform predictive maintenance on CAN bus data. For example, COMPREDICT[3] uses the built in sensors of a car for predictive maintenance. Since the information of OBD-II is restricted, COMPREDICT use the CAN bus signal, which contains all possible sensor signals of the car. Hence, they state to directly work in cooperation with car manufacturers to be able to process the CAN bus data (personal communication, January 31, 2017). Yet, predictive maintenance only affects a car internally and does not incorporate other cars in the environment and does not require a network.

Building a CAN bus network is not realistically obtainable as it requires hardware upgrades for shipped cars. It also requires car manufacturers to provide their protocols as every CAN bus is encoded uniquely. Instead, smartphone data could be used as a proxy from real world events to the network. If it is possible to detect events by smartphone data, there is no need for additional hardware in the car. Getting the smartphone data is cheap and deploying a marketable solution is easy, using existing application distributors. The number of smartphone user is still increasing. In 2017, 78% of the German people used a smartphone (Bitkom, 2017). Moreover, a smartphones are already connected via mobile Internet. Having that many available data sources being, theoretically, already connected to each other or rather to the Internet leads to the question, whether smartphone data is as useful as the CAN bus data.

2.2 Detection Through Smartphone Data

Smartphone acceleration data is used in a variety of fields and tasks. For example, it is used to detect human activity like sitting, walking straight, walking up- or down-stairs (Khan et al., 2010) or to distinguish, whether a person is shopping or taking the bus (Lee and Cho, 2011). It is also used successfully for walk distance estimation (Shih et al., 2012) or for fall detection (Vo et al., 2012).

In car driving environments, smartphone acceleration data is also used for various inferences. (Chu et al., 2014) use acceleration data of a smartphone, which is carried in the user's right pocket, to decide whether the user is the driver or a passenger. Also traffic light scheduling is estimated by making use of the high prevalence of smartphones (Zheng et al., 2014). For electric

[1] `https://www.pace.car/` (January 20, 2017)
[2] `http://www.vgate.com.cn` (January 20, 2017)
[3] `http://compredict.de` (February 06, 2017)

vehicles, smartphone acceleration data is used to estimate battery consumption (Alvarez et al., 2014).

Not only an indication of occupants or traffic is given by smartphone data, also road properties are estimated. (Fazeen et al., 2012) estimate street bumps, potholes and overall road properties (rough, uneven, or smooth) using acceleration data. The smartphone is placed in different locations in the car, in which the axes of the coordinate system are always aligned with the cars' coordinate system. (Mukherjee and Majhi, 2016) classify different types of road bumps using acceleration in z-direction of a mounted smartphone also with aligned coordinate axes. Another traffic and road condition estimation system called *Wolverine* is developed by (Bhoraskar et al., 2012). Here, the data is gathered with the phone being in some unknown position and orientation. A re-orientation algorithm is applied in one second data windows, to align the coordinate axes of the phone with the car. For each event, the data preparation is the same. First, K-means clustering (with $K = 2$) is applied to the data windows to classify the data points in *Bumpy* and *Smooth* parts. Some points are relabeled manually based on observation. The labeled data points are used to train a Support Vector Machine (SVM) to predict test data points.

An uncontrolled field study is done by (Sharma et al., 2015). The data that is used to predict road surface condition and driving behavior is gathered by an app (*S-Road Assist*) collecting data from accelerometer, gravity, magnetometer and location (GPS) sensor. During data recording, the smartphone is placed in any arbitrary position and orientation. To ensure the coordinate systems of the smartphone and the car are aligned, again, a *re-orientation* algorithm is applied every second over a data window.

Smartphone data is also used to detect accidents. By combining data of multiple sensors of a smartphone, including the acceleration information, traffic accidents are detected automatically to reduce the time from accident to emergency call. The acceleration data is combined with additional information of the other sensors available in a smartphone, like camera or microphone, to reduce false positives and provide more context information for the rescue forces (Thompson et al., 2010; White et al., 2011).

Instead of detecting car accidents after they occurred, systems are developed to detect accidents in advance. Collision warning systems, such as *iOnRoad* (Botzer et al., 2017), use a variation of sensors provided by a smartphone. *iOnRoad* is an Augmented Reality (AR) smartphone application that uses the camera, GPS, and the acceleration sensors to detect dangerous driving situations. Even though the user acceptance is still low, driving assistance systems are a promising application for smartphone sensor data.

To suggest safer driving styles, the behavior of the driver is analyzed. (Johnson and Trivedi, 2011) use the gyroscope, accelerometer and device Euler angle rotation values to distinguish between 12 types of driving behavior, such as aggressive or non-aggressive right or left turn. They use Dynamic Time Warping (DTW) for maneuver classification. It is stated that it is dif-

ficult to differentiate a left turn from a U-turn using only the accelerometer or gyroscope data. Combining data of the two sensors improves differentiation. The combined data set consists of x-axis rotation rate, y-axis acceleration and pitch. To determine the start and end of a maneuver a Simple Moving Average (SMA) of the rotational energy around the x-axis for a window of size k is used. The maneuver is set to begin when the SMA is higher than a threshold. (Johnson and Trivedi, 2011) conclude their system to be a viable system for driver style recognition. Data recorded from smartphone sensors is able to detect movements with similar quality compared to data recorder using the CAN bus. In contrast, (Paefgen et al., 2012) note that smartphones may be a less reliable data source for driving behavior estimation due to pose uncertainty and noise. They compare smartphone acceleration data to data from a mounted Inertial Measurement Unit (IMU). The data is recorded in a controlled field study. With this data, a driving behavior estimator is developed, similar to (Johnson and Trivedi, 2011). Furthermore, the rate of occurrence of driving events on the predefined test route is investigated. The investigation shows the data to be highly dependable on the smartphones position in the car. Three positions are compared: (a) face-up on the co-driver's seat, (b) the middle console between driver's and co-driver's seat, and (c) a smartphone holder attached to the front windshield. Highest correlation between IMU data and smartphone data is reached for case (c). However, also for case (a), which seems to be the *weakest* case, a substantial correlation can be registered. For the presented investigation by (Paefgen et al., 2012) only acceleration data of the smartphone is contributed. The gyroscope data is also recorded and it is mentioned that the algorithm may be extended by gyroscope or GPS data to eliminate invalid or inconsistent events. However, (Johnson and Trivedi, 2011) mention, maneuver recognition only works by using a combination of acceleration and gyroscope data, which puts (Paefgen et al., 2012) findings into perspective.

There is various literature showing that smartphones are a viable source for data to detect driving behavior or events.

(Mitrovic, 2005) uses the acceleration data that is labeled manually to infer driving events like right/left turn or right/left on roundabout. The data is transformed into feature space and a codebook of size 16 is generated. A codebook is a collection of characteristic feature vectors that are gained by clustering algorithms. An event is represented as a sequence of codebook entries. For every event, a Hidden Markov Model (HMM) is trained on that codebook. (Daptardar et al., 2015) also use an HMM for detection of events, whereas the HMM only detects whether an event is present or not. The classification is done by a Random Forest classifier. A comparison of mobile data and CAN bus data in driving event recognition is performed by (Sathyanarayana et al., 2012). For data acquisition, a car is equipped with many additional sensors and a portable device that is fixed to a mount. A SVM works best for estimation of maneuvers with the data of the portable device and a k-Nearest Neighbors (k-NN) classifier in combination with Latent Dirichlet Analysis (LDA) gives best results for the CAN bus data. The estimation using the portable device is concluded to outperform the estimation using the CAN

bus data, which shows that using smartphone data for driving event detection is a promising approach. (Cervantes-Villanueva et al., 2016) develop a system to estimate the current state of the car, where possible states are parking, parked, stopped, and driving. For data acquisition, the smartphone is fixed and its coordinate system is aligned to the car's coordinate system. Two hierarchical classifier are used for state classification. A Random Forest is used as a classifier, which outperforms a SVM. Instead of a smartphone, (Wu et al., 2016) use a fixed IMU for driving event detection. The authors state to achieve an overall performance of 93% accuracy at which the distinguished events are extreme cases in acceleration. Here, a SVM outperforms a decision tree and a k-NN classifier.

Instead of a fixed smartphone, (Eren et al., 2012) use a freely positioned and oriented device. The authors classify the driving behavior in safe and unsafe driving. The best results are gained with a Bayes Classification algorithm in combination with DTW. They state that Bayes Classification performs better than Random Forest. (Vaitkus et al., 2014) classify driving style in public transportation with freely placed smartphones. The driving behavior is distinguished in aggressive and normal driving. (Zheng and Hansen, 2016) grade driving behavior, i.e. cluster extracted features in grade classes. To label the data GPS information is used. As features, three time-domain and three frequency-domain features per axis are used, as proposed by (Lu et al., 2010). The dimensionality of the feature space is reduced with Principal Component Analysis (PCA). For classification a SVM is used.

The current research concentrates on high level driving assistance, driving behavior analysis, and driver classification, which may be a valuable source of information for insurance companies. There is still little work on low level detection of car driving events, such as (Siegel et al., 2015) detects wheel imbalances.

This thesis investigates which low-level events are detectable with smartphone data.

3 Fundamentals

The previous chapter summarizes the current state of research in detecting events in a driving environment. This chapter introduces the concepts and techniques used in this thesis.

3.1 Signal Processing

To work on acceleration data or on any sensor data, respectively, it has to be preprocessed. Since sensor data includes noise of the sensor itself, noise reduction has to be applied. To reduce noise in data, a low-pass filter can be applied.

3.1.1 Low-Pass Filter

The low-pass filter filters high frequency parts of a signal and lets low frequency parts pass (Kaiser and Reed, 1977). An example signal is shown in Figure 3.1a. The signal is sampled with 100 Hz by the function

$$f(t) = \sin(1.5 * 2\pi * t) + 1.5\cos(9 * 2\pi * t) + 0.5\sin(12 * 2pi * t) \tag{3.1}$$

Note, that the sinus and the cosine functions can be written as

$$f(t) = A\sin(2\pi f\, t + \gamma) \tag{3.2}$$

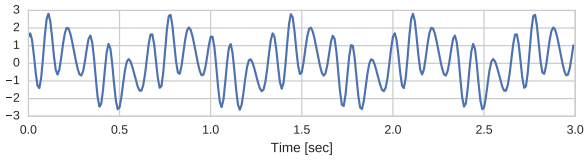
$$f(t) = A\cos(2\pi f\, t + \gamma) \tag{3.3}$$

where A is the amplitude, f is the frequency of the wave, and γ is the phase.

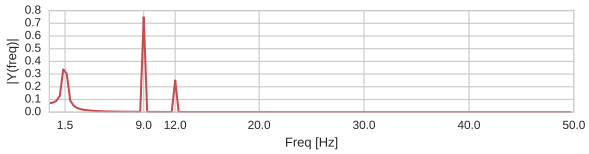
Comparing Equations (3.2) and (3.3) to Equation (3.1) there are three superimposed waves with frequencies of 1.5, 9, and 12 Hz.

Fast Fourier Transformation (FFT) is an efficient algorithm to calculate the Discrete Fourier Transformation (DFT). DFT is used to obtain the frequency distribution of a signal (Heideman et al., 1984).

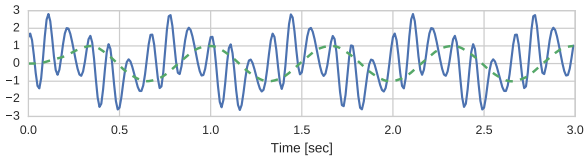
Figure 3.1b shows the frequency distribution of the signal in Figure 3.1a. It shows three peaks of frequencies at 1.5, 9, and 12 Hz as expected. Goal of the low-pass filter is to eliminate the high frequency parts of the signal. A cutoff frequency has to be defined. The frequencies higher than the cutoff are suppressed. In the example, the cutoff frequency is set to 4 Hz.



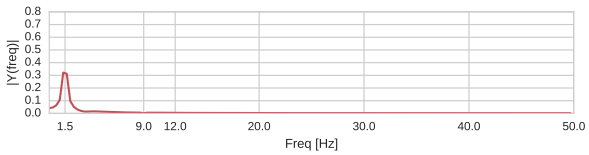
(a) Example signal. Signal is sampled with 100 Hz and consists of three superimposed functions with different frequencies (Equation (3.1))



(b) Frequency distribution of the signal in (a). It shows three peaks at 1.5, 9, and 12 Hz



(c) The solid line is the signal as in (a) the dotted line shows the filtered signal. Only the low frequency part of the signal remains



(d) Frequency distribution of the filtered signal in (c). Only the peak at 1.5 Hz remains, the other frequencies are filtered out

Figure 3.1: Example for an applied low-pass filter. The Figure shows a noisy signal (a) and its frequency distribution (b) and the filtered signal (c) with correspondent frequency distribution (d)

An basic implementation of a low-pass filter is the moving average filter. Here, a data point at time T is recalculated by the average of the last N data points.

$$x_T^{\text{new}} = \sum_{t=0}^{N} \frac{x_{T-t}}{N} \tag{3.4}$$

Hence, high frequencies get averaged out.

3.2 Transforming Smartphone Data

During this work data is used produced in the real world by an arbitrary positioned smartphone. If the smartphone's position and orientation is not fixed, the recorded acceleration vectors have to be transformed such that they reference to the car's coordinate system. Otherwise, two acceleration vectors of two differently orientated smartphones are not comparable. For instance, without transformation the resulting accelerometer vectors of two contrariwise orientated smartphones in the same car produce different data. In the following, the required transformation basics are introduced.

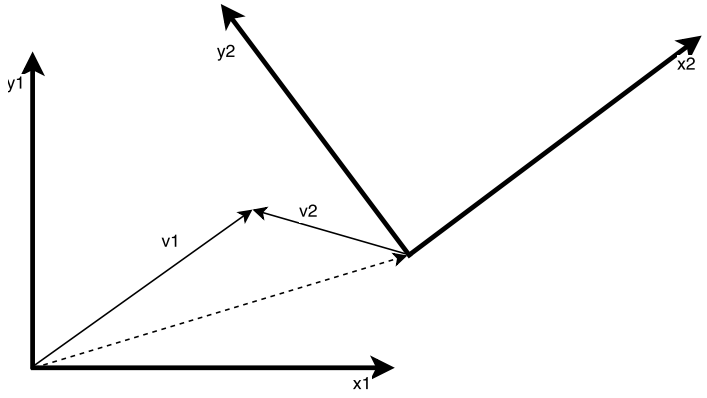


Figure 3.2: Vector transformation example. The thick arrows are the basis vectors of the coordinate systems. The thin solid lines are the vectors v1 and v2 which both represent the same point with different reference coordinate systems. The thin dotted line depicts the transformation of coordinate system 1 to 2.

3.2.1 Vector Transformation

A vector is always defined in relation to basis vectors. These basis vectors build up a coordinate system. Basis vectors can differ between systems. Figure 3.2 illustrates a two-dimensional example of two vectors with different reference coordinate systems representing the same point. One could see the example as coordinate system 1 (represented by basis vectors *x1*, *y1*) as the coordinate system of a camera and coordinate system 2 (represented by basis vectors *x2*, *y2*) as the coordinate system of a room photographed with the camera. To state where in the room an object with camera coordinates v1 is arranged, the transformation of camera coordinates (v1) to room coordinates (v2) has to be known. If the room coordinates of the object are known it can be compared across multiple images.

The same problem occurs with acceleration data. Here, the vectors are three dimensional. The acceleration values of the recorded data denote the acceleration direction of the smartphone to itself. In the first place, there is no information of the acceleration direction of the system (the car). Hence, the acceleration vectors have to be transformed, such that they represent the acceleration of the car.

To transform vectors from one coordinate system to another, transformation matrices are used. A transformation matrix for three dimensional vectors is a $\mathbb{R}^{4\times4}$ matrix. The transformation matrix is applied to a vector by matrix vector multiplication. Therefore, the vector has to be extended to a homogeneous vector, i.e. a fourth dimension is introduced, which represents the

scale of the vector. If transformed from a euclidean vector to homogeneous vector, one is chosen as the scale value. A vector (x, y, z) becomes $(x, y, z, 1)$ as homogeneous vector. The identity transformation T_{id} is defined by an 4×4 identity matrix. The last column of the matrix denotes the translation of the vector and the upper left $\mathbb{R}^{3\times3}$ submatrix the rotation around the x, y, and z axis. Rotations of an angle α around x, y, and z axis are defined by the following rotation matrices:

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}, R_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix}, R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.5)$$

Transformations can be composed and inverted. Vector transformation is associative, but not commutative. To compose transformations, the transformation matrices have to be multiplied by each other. For example, to rotate a vector first around the x axis, then the y axis and lastly around the z axis, the transformations have to be applied consecutively:

$$T_z(\alpha_z) * T_y(\alpha_y) * T_x(\alpha_x) * v = v^* \quad (3.6)$$

where T_z,T_y, and T_x are the transformation matrices that rotate the vector v around z, y, and x axis, and v^* is the rotated vector. T_w with $w \in \{x, y, z\}$ is built as $T_w = \begin{pmatrix} R_w(\alpha_w) & 0 \\ 0 & 1 \end{pmatrix}$ with R_w defined in Equation (3.5).

Instead of representing transformations by matrices, transformation can be represented by quaternions. Quaternions are defined in $\mathbb{R}^4$. They span a four dimensional vector space. Quaternions consist of a real part and an imaginary part that consists of three numbers. They are represented as $(a, b\text{i}, c\text{j}, d\text{k})$ with a as the real part and i, j, k the imaginary part. Same as transformation matrices, quaternions are associative but not commutative. Quaternions provide a more compact notation and are faster to compute than matrices (Shoemake, 1985). A subset of quaternions that describe rotations on the unit sphere are unit quaternions. Unit quaternions are all quaternions with norm 1, i.e. $||q|| = 1$, representing rotations around the origin. Quaternions may also be converted to rotation matrices. Given a quaternion of form $q = (a + b\text{i} + c\text{j} + d\text{k})$, the corresponding rotation matrix is

$$R = \begin{pmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2bd + 2ac \\ 2bc + 2ad & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab \\ 2bd - 2ac & 2cd + 2ab & a^2 - b^2 - c^2 + d^2 \end{pmatrix}$$

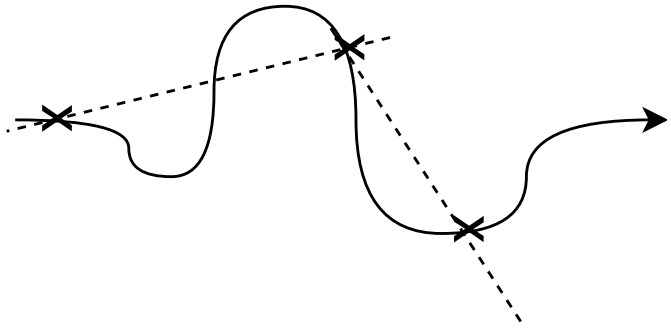


Figure 3.3: Worst case scenario for direction of travel estimation. The solid line is the way of travel, 'x' mark recorded GPS coordinates and the dotted line the estimations of direction of travel.

A vector in three dimensional space $v = (v_x, v_y, v_z) = v_x\mathrm{i} + v_y\mathrm{j} + v_z\mathrm{k}$, i.e. real part equals zero, can be rotated by a quaternion q by

$$v' = q\,v\,q^{-1} \tag{3.7}$$

where v' is the rotated vector.

3.2.2 Bearing

There is no information on the direction of travel in the raw acceleration values. The smartphone only measures the acceleration in reference to itself. To transform the acceleration such that it points towards direction of travel, additional information is required.

The bearing angle denotes the angle of direction of motion and geometric north. The angle can be calculated by two consecutive GPS points. If two geographic positions are known, the angle of the line connecting the two GPS positions and geometric north can be computed.

This is so far the only indicator of the direction of travel. But GPS is inaccurate (approximately 5 meter inaccuracy under perfect conditions (Wing et al., 2005)) and the frequency of GPS recordings is relatively low. Hence, changes of the direction of travel are recognized delayed or not at all, if the direction changes several times between two consecutive bearing angle computations. Figure 3.3 shows a worst case scenario for estimations of direction of travel. Even interpolation of the bearing angle would not yield any improvement of the estimations. Nevertheless, the bearing angle is the only information given about the direction of travel. Without it, acceleration could not be distinguished from deceleration.

3.3 Machine Learning

In this thesis, machine learning techniques are used. The following section introduces the basic concepts and definitions of machine learning and the algorithms used in this thesis.

3.3.1 Definitions

Machine learning combines techniques to find patterns in data. The computer is *learning* these patterns and applying it to new, unseen data to estimate the outcome of this new data.

In the following, some basic concepts of machine learning are introduced.

Supervised and Unsupervised Learning Problems

There are two main distinctions of machine learning problems: Supervised and unsupervised learning.

The goal of supervised learning is to learn a mapping from input to output, given a set of input-output pairs $D = \{(x_i, y_i)\}_{i=1}^{N}$ where x_i are the input vectors, y_i the output values, and N the number of data points (Murphy, 2012). Therefore, the target value has to be measurable. Since the true values are known, the learning algorithm can estimate its current error on the data. An example for supervised learning is an e-mail spam filter. The algorithm gets several examples labeled by the user as Spam and tries to predict the label for new examples.

In an unsupervised learning problem, the target value is unknown to the algorithm. This can be helpful, if the target value is not measurable. Hence, the goal of unsupervised learning is to find patterns in data, whereas the data $D = \{x_i\}_{i=1}^{N}$ consists only of the input vectors x_i (Murphy, 2012). For example, when the task is to assign a customer of some shop to a group of customers, it is not known, which customer belongs to which group or how the groups are defined. Without expert knowledge, the algorithm builds groups and assigns the customer to one. When there is no information on the target value, there is also no error indication for the learning algorithm. Nevertheless, the information found by the learning algorithm to distinguish the groups can be utilized to derive key features.

There is also a combination of supervised and unsupervised learning called semi-supervised learning. It comes in handy, when the data has to labeled manually by some expert. Labeling data manually is time-consuming. Hence, the expert labels only a part of the data (supervised) and the gained information is used to label the rest of the data automatically (unsupervised).

Regression and Classification Problems

Supervised machine learning problems can be distinguished in two main inference classes: regression and classification.

Figure 3.4a depicts an example machine learning task. 'x' marks sampled data points from an unknown underlying function or distribution.

The goal of regression is to estimate a correlation between variables to model an underlying function, so that for every new unseen data point the output value is inferred. Figure 3.4b shows

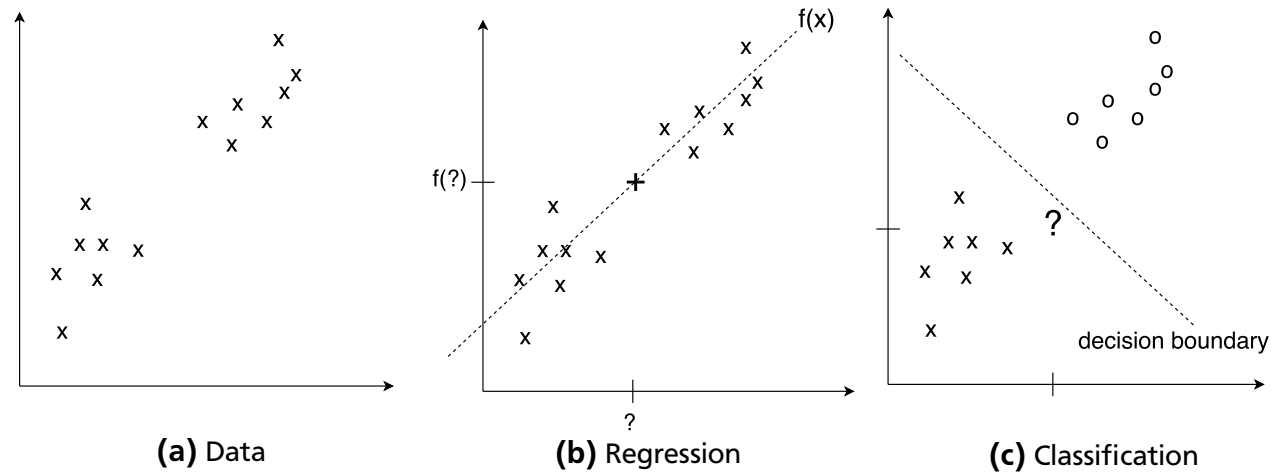


Figure 3.4: Classification and regression. Sketch of a supervised machine learning problem. (a) depicts the available data. 'x' are the data points. (b) shows one possible regression solution f(x) to decide which value the new data point ('?') is assigned. The dotted line is the derived function. In (c) there are two classes 'x' and 'o', the dotted line symbolizes one possible decision boundary. The new data point ('?') is assigned to class 'x'.

one possible solution of the regression problem. The dashed line depicts a potential model, that is learned by a regression algorithm. Using the learned model $f(x)$, the output value of a new data point ('?') is estimated, as illustrated by the '+'.

Figure 3.4c shows the same data points. In this case, the given data points are assigned to classes, marked by 'x' and 'o'. In classification, the goal is to find a decision boundary that separates the training data points of the two classes, such that a new unseen data point can be assigned to one of the classes. The dashed line depicts one possible decision boundary. The new data point '?' is assigned to class 'x'. There is more than one possible decision boundary that can be inferred from the data. Unless the underlying distribution is known, the optimum decision boundary can not be stated.

Over-fitting

Over-fitting is a major problem in machine learning. Occam's Razor is a principle, which states that a model needs to be as simple as possible to explain the data. Over-fitting is the violation of this rule, when the model is more complex than it needs to be (Hawkins, 2004). Figure 3.5 depicts an example regression task. The Figure shows four polynomials of different order $M \in \{0, 1, 3, 9\}$. The higher the order of the polynomial is, the more complex the model gets. All polynomials are fitted to the data, such that the Mean Squared Error is minimized. The plots show that the model error on the data that is used to fit the model is lowest with order $M = 9$. However, the shape of the underlying function, depicted as green line, appears best for order

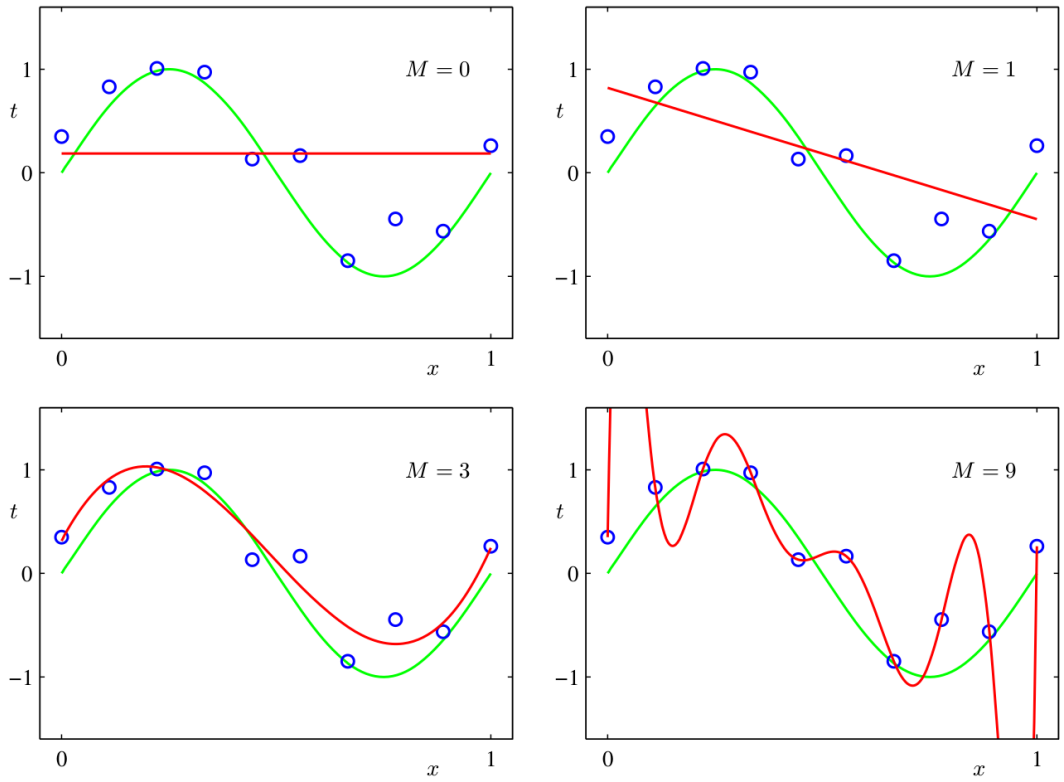


Figure 3.5: Plots of polynomials with order M, shown as red curves, fitted to data set, shown as blue circles, sampled by a disturbed function, shown as green curve. Polynomials are fitted, such that Mean Squared Error is minimized. (Bishop, 2006, p.7)

$M = 3$. Hence, the error for a new unseen data point is lower on the less complex model, i.e. the complex model does not generalize.

This is also known as the Bias-Variance trade-off (Geurts, 2002). The simplest model ($M = 0$) is not flexible enough to reflect the underlying function, this problem is called *bias*. The most complex version ($M = 9$) generates different models of every different set of training data, since it over-fits on it. In this case, the model has too much *variance*. Hence, the model with order $M = 3$ provides the best *bias-variance tradeoff*.

Train, Test, and Validation Set

Over-fitting occurs if the model is too complex and the parameters are optimized only on one data set. To avoid over-fitting, the data set is split into three separate subsets. The three subsets are train, test, and validation set. Train and test set are used to optimize the parameters of the model, i.e. train the model. The model is fitted on the train set and parameters are adapted by maximizing the performance on the test set. Hence, the model is adapted, such that the performance on an *unseen* set is increased.

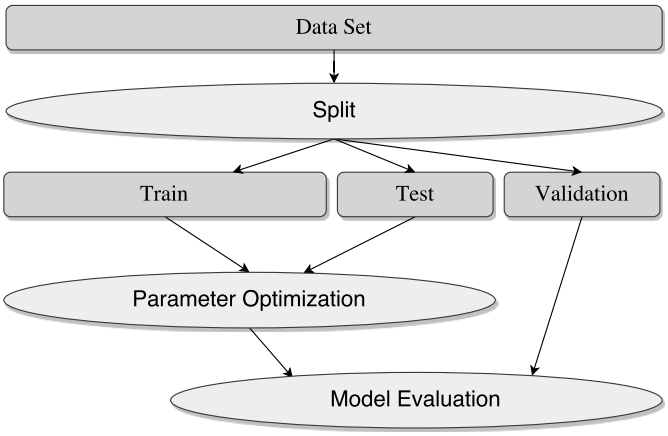


Figure 3.6: Illustration of train, test, and validation split of a data set

To evaluate the obtained model, the validation set is used. It is very important to use the validation set as a *hold-out* set and to not incorporate it in train phase. Otherwise no evidence of generalization of the model can be given. Figure 3.6 illustrates the procedure.

It has to be noted, that splitting the data in train, test, and validation set does not guarantee the model generalizes for new unseen data. The US army, for example, tried to train a model to detect tanks, but due to bias in the data, only learned to distinguish cloudy days from sunny days (Dreyfus and Dreyfus, 1992). However, under the assumption that the data is not biased, splitting data into a train, a test, and a validation set is a common approach to use data to train and evaluate a model.

3.3.2 Cross-Validation

In case there is no dedicated train and test data set, a common approach for model learning is to use *k-fold cross validation* (Arlot and Celisse, 2010). Instead of solely splitting the data into a train and a test data set, it is split into k equally sized splits or folds, respectively. Again, beforehand a evaluation data set has to be extracted for model selection. The k folds are used for parameter optimization. The optimization is done k times, such that in every iteration $k-1$ splits are used as training data and one split is used as test. This way, every split is used at least once as test set. After k iterations either the best model is used, as indicated by the evaluation data set, or an average of the models, depending on the method.

3.3.3 Decision Tree Classifier

The basic idea of the Decision Tree classifiers is to divide a complex decision into a union of simple decisions (Safavian and Landgrebe, 1991). Thus, a decision tree splits one dimension of feature space in every level.

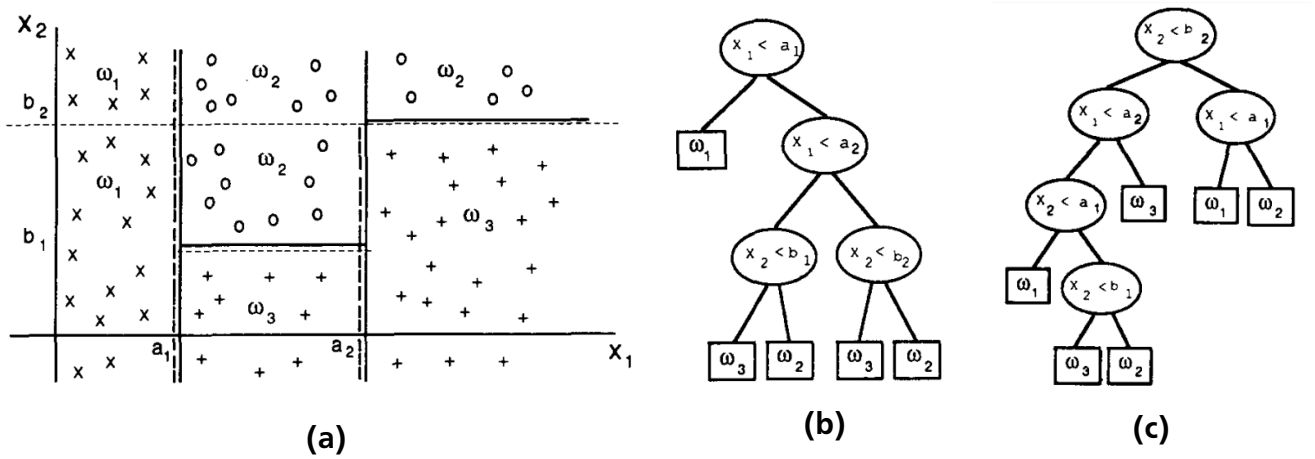


Figure 3.7: Decision Tree example. (a) shows the data in two dimensional space. Data consists of samples of three different classes marked with x, +, and o. Two possible decision boundaries are shown, the solid and the dashed line. (b) shows the decision tree for the solid line decision boundaries and (c) the decision tree for dashed lined decision boundaries. (Safavian and Landgrebe, 1991)

Figure 3.7 shows an example classification task and two possible resulting Decision Trees. The data is present in two dimensional space. In each level of the decision trees, one of the dimensions is split in two parts. The deeper the tree becomes, the more specific the partition gets. The leaves specify the class that the decision tree infers for the data points.

The parameters, which have to be adapted, of a Decision Tree classifier are the maximal number of features to be considered for the splits, the maximal depth of the tree[1], the minimum number of samples required in a node to split it, and the minimum number of samples required to be in a leaf node.

3.3.4 Random Forest

Instead of training a single decision tree, the idea of Random Forest is to use an ensemble of weak Decision Trees (Breiman, 2001). During train phase, the training data is randomized and random samples are used to train multiple Decision Trees. Data points are assigned to a class by majority vote of the Decision Trees. Figure 3.8 depicts the pipeline of Random Forests.

The parameters of a Random Forest classifier to adapt are similar to the Decision Tree ones. Additionally, the number of Decision Trees in the forest has to be adapted.

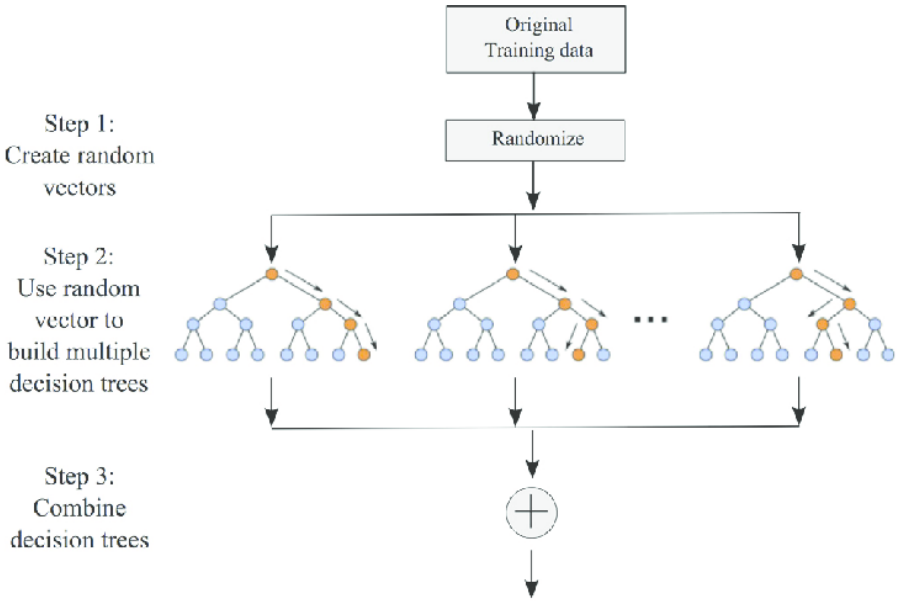
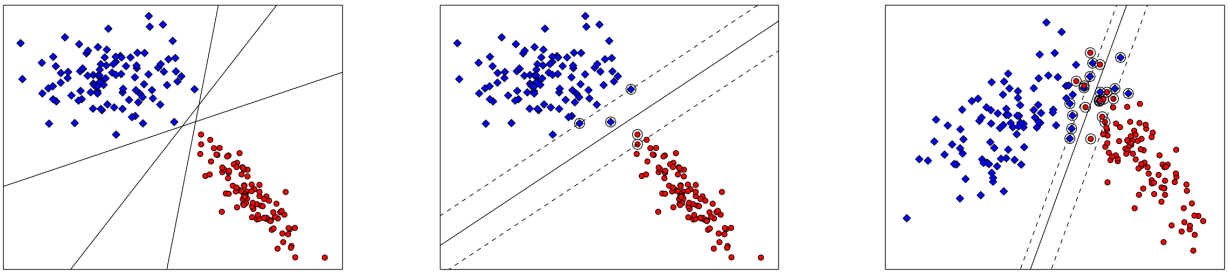


Figure 3.8: Random Forest pipeline. First the training data is randomly sampled. The random vectors are used to build an ensemble of weak Decision Trees. The Decision Trees are combined by majority vote. (Malekipirbazari and Aksakalli, 2015)



(a) **(b)** **(c)**

Figure 3.9: SVM example. The markers define data points of two different classes. The solid lines are decision boundaries and the dashed lines are the margins. The circled markers flag the support vectors. (a) There are infinitely many decision boundaries that separate the two classes perfectly, (b) decision boundary that maximizes the margin together with support vectors, (c) decision boundary for a dataset that is not linearly separable.

3.3.5 Support Vector Machine

The Support Vector Machine (SVM) is initially introduced by (Boser et al., 1992) for perfectly separable data and later enhanced by (Cortes and Vapnik, 1995) for data that is not perfectly separable. The goal of a SVM is to find a decision surface that maximizes the margin between training instances and the decision surface to classify data. It maps the input vectors in a high dimensional feature space with a non-linear mapping. For the construction of the decision function only a subset of the training vectors is required, called the *support vectors*. SVM is generally a binary classifier, for which the class labels are $y \in \{1,-1\}^N$.

The primal formulation of a SVM is defined by

$$\begin{aligned} \min_{w,b,\zeta} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{N} \zeta_i \\ s.t. \quad & y_i(w^T \phi(x) + b) \quad \geq 1 - \zeta_i, \\ & \zeta_i \quad \geq 0 \text{ and } i = 1, \ldots, N \end{aligned} \tag{3.8}$$

where w are the weights of the data points, $\phi(\cdot)$ is the feature transformation of inputs x, ζ is the slack variable to allow misclassification, C is the weight on misclassification.

The decision surface has the form

$$f(x) = \sum_{i=1}^{N} y_i \alpha_i k(x_i, x), \tag{3.9}$$

where α_i is the weight of the support vector in the feature space and $k(\cdot)$ is a kernel function.

3.3.6 Linear Least Squares Regression

The goal of regression is to model an underlying function to estimate the function value of unseen data points. In linear least squares regression, the function is modeled as a linear combination of the input variables (Bishop, 2006)

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D \tag{3.10}$$

where $\mathbf{x} \in \mathbb{R}^D$ denote the input vector, $\mathbf{w} \in \mathbb{R}^{D+1}$ is the weight vector.

[1] deeper trees lead to over-fitting, since every data point can be represented as a separate leaf, which is undesired, since it does not generalize to unseen data

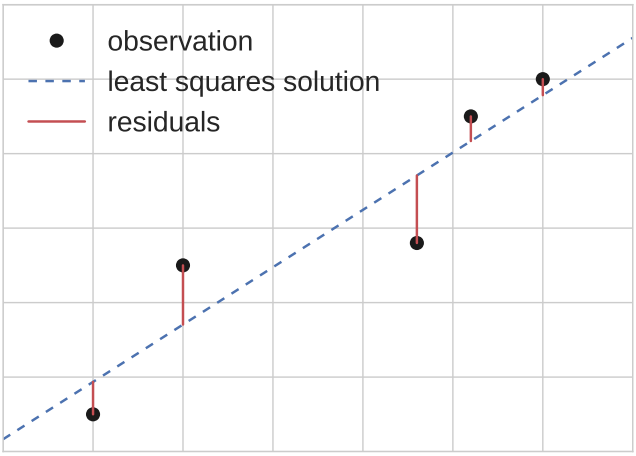


Figure 3.10: Linear Least Squares example. The polynomial fit to the data has order one. The markers depict the observed data, the dashed line shows the model fit to the observations by linear least squares, and the solid lines show the residuals, i.e. the error of observed value to estimated value. Least squares minimizes these residuals.

The sum-of-squares-error function is defined by

$$E(\mathbf{w}) = \sum_{n=1}^{N} (t_n - y(\mathbf{x}_n, \mathbf{w}))^2 \tag{3.11}$$

where t_n is the target value of the n-th input vector $\mathbf{x}_n$.

Least squares regression adapts the weights, such that the error E is minimized. The deviation of target and estimated value is also called residual. Hence, the error E can also be defined as the sum of squared residuals. Figure 3.10 depicts an example of a regression problem solved by linear least squares. The depicted learned model minimizes the square of residuals. Whereas the target values are the observed values.

3.3.7 Decision Tree Regression

A regression tree is organized analogous to decision tree classifiers. Compared to decision tree classifier, the leaves do not define classes but values. The estimated signal is not continuous, since the number of leaves is limited. The number of possible values that are estimated depends on the number of leaves and, hence, on the depth of the tree. The inferred signal has a step shape.

Figure 3.11 depicts an example inference task solved with a decision tree regression model. Two instances of decision tree regressions are shown, one with maximal depth of the decision tree set to three and the other set to six. The decision tree with maximal depth set to six is much

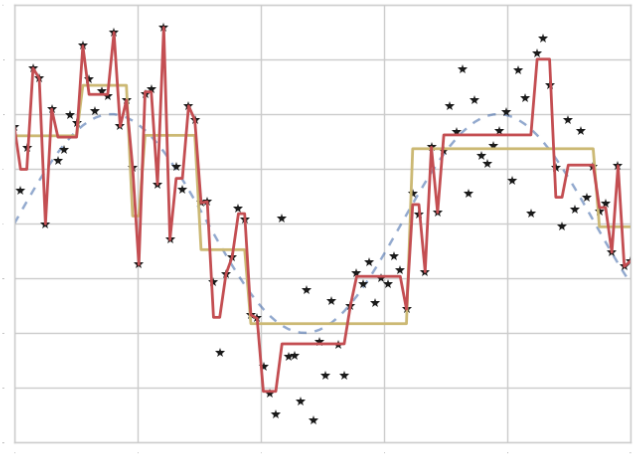


Figure 3.11: Decision Tree Regression example. The dashed line depicts the underlying true function, the '*' are noisy observations, the yellow line are estimates performed by decision tree regression with maximal depth set to 3 and the red line are estimates performed by decision tree regression with max depth set to 6

more adapted to the data than the one with depth three. The deeper the tree gets the more values it can represent, however, it also leads further to over-fitting.

The parameters to adapt are the same as for the decision tree classifier.

3.3.8 Feature Generation

Machine learning models can not be trained with raw data. Instead, features have to be created and selected as not all generated features add value to the outcome of the prediction model.

There are many ways to generate features. The kind of features and the way they are generated differs, depending on the task, the data, and the machine learning algorithm used for inference.

In the following, the feature generation for the experiments in this thesis are presented.

To generate features, the data is windowed using a sliding window, i.e. the time series data is sliced into equally sized fractions. Thereby, every fraction has overlapping data points with the neighboring one. In literature, the window sizes used for smartphone acceleration data vary between one and three seconds, depending on the use case. (Bhoraskar et al., 2012) state that a window size of one second would be reasonable for their traffic and road condition estimator, whereas, (Enev et al., 2016) find a window size of three seconds with an overlap of 25% to fit best for driver detection.

Several values are computed for every window as features. First, the magnitude (Equation (3.12)) is computed (Das et al., 2010). The magnitude is the absolute value of the acceleration vector.

$$\bar{a} = \sqrt{x^2 + y^2 + z^2} \tag{3.12}$$

The gradient of all values (acceleration in x, y, and z direction and magnitude) is calculated (Murphey et al., 2009; Enev et al., 2016). The derivative of acceleration is called jerk. For every acceleration direction, the magnitude, and the jerk, the minimal and maximal values are taken as features. Mean and standard deviation are calculated for all values (Bhoraskar et al., 2012; Mitrovic, 2005; Lu et al., 2010; Sathyanarayana et al., 2012; Sharma et al., 2015). (Lu et al., 2010) suggest to use the mean-crossing-rate as a feature, i.e. the rate that the value is crossing the mean value. Several differences are also taken as features, like the difference between mean and maximum, mean and minimum, maximum and minimum (Sathyanarayana et al., 2012).

3.3.9 Metrics

There are several metrics to measure the quality of a model. The metric is chosen, depending on the class of problem and on the goal of the inference.

In the following, *TP* (True Positive) denote the examples that are correctly classified as true, *FP* (False Positive) the examples that are incorrectly classified as true. Similarly, *TN* (True Negative) and *FN* (False Negative) denote the correctly and incorrectly classified negative examples. *P* denote all the positive examples and *N* all the negative examples.

For classification, accuracy, precision, and recall are the most basic metrics. Whereby, accuracy specifies the ratio of correctly classified instances to the overall number of instances.

$$acc = \frac{TP + TN}{P + N} \tag{3.13}$$

Precision indicates the ratio of correctly as true classified data points over the number of all instances classified as true.

$$precision = \frac{TP}{TP + FP} \tag{3.14}$$

Recall specifies the ratio of correctly classified true examples over the number of all positive examples. Recall is also known as the *True Positive Rate*.

$$recall = \frac{TP}{TP + FN} \tag{3.15}$$

Instead of considering precision and recall separately, the F_1 score incorporates both of them.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \tag{3.16}$$

The metrics presented above are used for classification problems. When the quality of a regression task needs to be determined, other metrics have to be considered. One of the most common quantitative performance metrics for regression is the Mean Squared Error (MSE) (Wang and Bovik, 2009). The MSE computes the mean error of all estimations, whereby due to the squaring the small errors are weakened and the large errors are magnified.

$$MSE = \frac{1}{N} \sum_{i}^{N} (x_i - x_i^*)^2 \tag{3.17}$$

where x_i is the estimated value, x_i^* the true value, and N the number of data points.

3.3.10 Parameter Optimization

Every machine learning model depends on parameters, whereby, the number and kind of the parameters depends on the model. In most cases, the parameters have a huge influence on the quality of the model. Hence, it is important to optimize the parameters.

For optimization, a cost function is needed. The goal of the optimization is to find the best value of the cost function. If the gradient of the cost function is present, gradient descent should be performed. On gradient descent, the cost and the current gradient is computed. The next parameter set is chosen, based on the gradient. One disadvantage of gradient descent is the possibility of finding a local optimum. To overcome this problem, optimization can be started with several different initial parameter settings. In case the gradient of the cost function can not be computed or there are too many local optima, grid search can be performed. In forefront of the optimization, a grid of all parameter settings to test is prepared. The costs for all of these settings are computed and the parameter setting with minimal cost is chosen as best parameter set.

3.4 Car Systems

In this thesis, activations of car safety systems are estimated. The investigated systems are the Anti-lock Braking System (ABS) and the Electronic Stability Control (ESC). The following section introduces the mode of operation of these two systems.

3.4.1 Anti-lock Braking System

ABS is part of the safety system of a car. It supports the driver during brake events, where one or more wheels would lock without intervention, to decrease stopping distance.

The first ABS system was the 1952 Dunlop Maxaret, which prevented aircraft wheels from being locked during brake (Veloso and Fixson, 2001). The first commercial car ABS system was introduced in 1984 by Bosch and ITT-Teves (Veloso and Fixson, 2001).

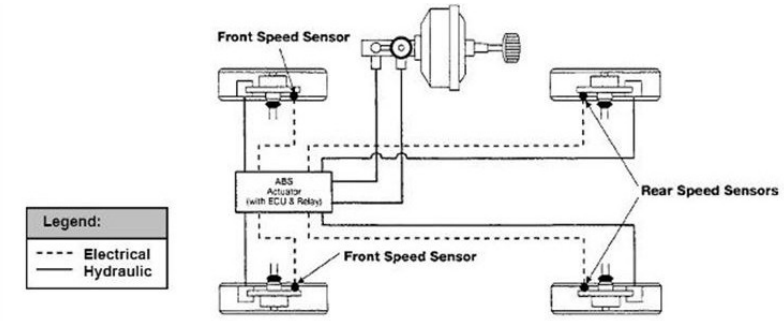


Figure 3.12: Sketch of an anti-lock braking system (England, 2014)

An ABS system consists of an electrical control unit, electrical and hydraulic connections, and wheel speed sensor at every wheel. Figure 3.12 depicts a sketch of the structure of an ABS system. The control unit checks the wheel speed. The speed of the individual wheel can vary due to slip because of the underground. If one wheel sharply decelerates, the control unit releases the brake of the affected wheel by releasing brake fluid of the particular brake.

The brake is released and tightened again up to 15 times per second (Nice, 2000). The releasing and tightening can be felt by the driver in terms of a pulsing brake pedal.

3.4.2 Electronic Stability Control

ESC stabilizes the car by specific activation of the brakes and engine management. The safety system assists the driver to keep the vehicle on the intended path (Liebemann et al., 2004).

If the car loses control, i.e. one or more wheels are moving in a different speed than calculated from steering input and turning angle, brakes are activated actively and engine power is adapted (Lie et al., 2006).

Figure 3.13 depicts the general procedure of an ESC system. First, the desired course is calculated by taking steering angle and wheel speed into account. In parallel the actual course is computed with the lateral acceleration and the yaw rate. The difference of the two courses

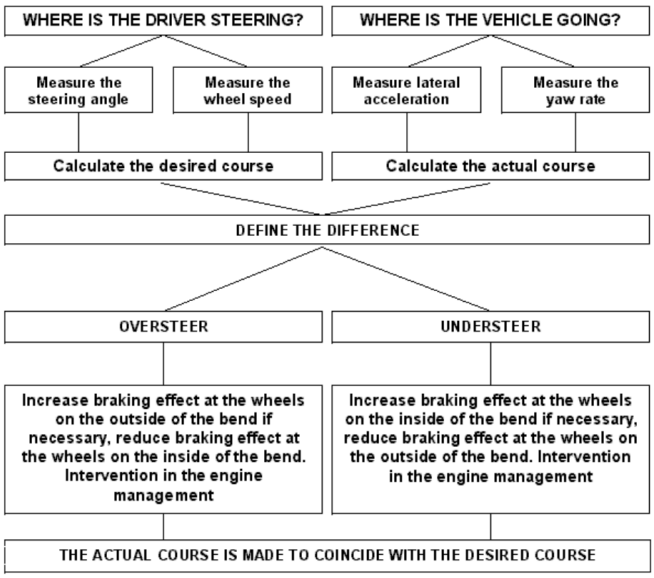


Figure 3.13: Principal of Electronic Stability Control operation. (Sferco et al., 2001)

is determined and oversteering and understeering are handled accordingly. Whereby, when taking a curve, understeering causes the car to not turn enough and skid off the road and when oversteering occurs, the car turns too sharply and tends to spin.

4 Data

The previous chapter introduces the current state of research and the fundamentals of this work. This chapter presents and describes the data that is used in this work.

In advance of this project data was recorded. There are two sources of data: The Controller Area Network (CAN) bus of a car and an Android smartphone. Both sources are joined using their time stamps.

4.1 Data Source

The data was recorded during an uncontrolled field study in forefront of this thesis by researchers of ETH Zürich and university of St. Gallen (Ryder et al., 2016). For this study, 72 Chevrolet Captiva of the *Touring Club Schweiz* were equipped with an OBD-II dongle. An OBD-II dongle can be used to record the CAN bus data, if access of the CAN bus is not restricted. Note, that messages of the CAN bus are not standardized and may vary even between models of a car (Ryder et al., 2016). Additionally, all drivers were provided with an Android smartphone, that was linked with the dongle via Bluetooth, and a mount. The dongle transmits the CAN messages to the smartphone, which then sends the data, including the CAN messages and the smartphone's internal sensor readings, to a server via GSM. The data was gathered over a period of seven months.

The data is stored in a Cassandra[1] data base.

4.2 Controller Area Network (CAN) Bus Data

The CAN bus delivers data of all sensors in a car with a resolution of 8 bit. A list of recorded signals is shown in Table 4.1. The data is gathered with a maximum rate of 30 Hz. System activations like ABS or ESC are recorded when they are triggered on or off, i.e. there is a data entry with time stamp when the system is activated and one more entry with time stamp when it is deactivated again.

The vehicle acceleration signal is saved in two's complement. To realize negative values in hardware, the two's complement is common practice. Since hardware can not code a minus sign, the most significant bit is utilized to encode negative values. When the value is just converted in a decimal number, without considering the two's complement, the value jumps from zero

[1] `http://cassandra.apache.org/` (March 05, 2017)

Signal	Unit	Range
Anti-lock Braking System (ABS)	–	off, on
Brake Pedal Position	%	0 – 100
Brake Pressure		$\mathbb{R}^+$
Engine Speed	RPM	$\mathbb{R}^+$
Electric Stability Control (ESC)		off, on
Fuel Consumption	L/100km	$\mathbb{R}^+$
Fuel Level	Liter	$\mathbb{R}^+$
Gear	–	park, reverse, neutral, drive
Lateral Acceleration	m/s^2	$\mathbb{R}$
Lights	–	off, on, unsure, highbeam
Longitudinal Acceleration	m/s^2	$\mathbb{R}$
Odometer	km	$\mathbb{R}^+$
Outside Temperature	°C	$\mathbb{R}^+$
Steering Wheel Angle	deg	$\mathbb{R}$
Traction Control System (TCS)	–	off, on
Throttle Pedal Position	%	0 – 100
Vehicle Acceleration	m/s^2	$\mathbb{R}$
Wheel Slip Status		No Slip, Negative Slip, Positive Slip
Wheel Speed Back Left	km/h	$\mathbb{R}$
Wheel Speed Back Right	km/h	$\mathbb{R}$
Wheel Speed Front Left	km/h	$\mathbb{R}$
Wheel Speed From Right	km/h	$\mathbb{R}$
Yaw Rate	deg/s	$\mathbb{R}$

Table 4.1: List of recorded CAN bus signals

to maximal value. Hence, a value of the affected signals has to be processed before applying any algorithms on them. To convert a value from two's complement to a continuous signal, the maximum value has to be subtracted from all values higher than half of the maximum. Thus,

$$x_* = \begin{cases} x - \text{maxval}, & \text{if } x \geq \frac{\text{maxval}}{2} \\ x, & \text{otherwise} \end{cases} \tag{4.1}$$

where x is the value in two's complement with a range of $[0; \text{maxval}]$, maxval is the maximum value of the signal, and x_* is the new signed value with a new range of $[-\frac{\text{maxval}}{2}; \frac{\text{maxval}}{2}]$.

4.3 Smartphone Acceleration Data

Data of a smartphone is simultaneously recorded. A smartphone contains many sensors. Three of them are used for the records of the dataset used in this project: accelerometer, magnetometer, and GPS.

Signal	Unit	Dimensionality	Avg. Freq.
Acceleration	m/s^2	$\mathbb{R}^3$	50 Hz
Bearing	deg	Scalar	1 Hz
GPS Altitude		Scalar	1 Hz
GPS Speed	m/s	Scalar	1 Hz
Rotation	quaternion	$\mathbb{R}^4$	5 Hz

Table 4.2: List of recorded smartphone signals

The accelerometer values are directly utilized for inference. Whereas, the magnetometer is used to compute the quaternions that are required for the transformation of the accelerometer vectors from smartphone coordinate system to world coordinate system. The GPS values are used to compute the speed, the altitude, and the bearing angle, which is used to rotate the acceleration vectors from world coordinate system to the car's coordinate system.

Table 4.2 presents an overview of recorded smartphone signals and their frequencies. Accelerometer values are recorded with an average frequency of 50 Hz. The quaternion rotation vector is recorded with an average frequency of 5 Hz, the GPS speed and altitude are recorded with an average frequency of 1 Hz, and the bearing angle is recorded with an average frequency of 1 Hz.

4.3.1 Coordinate Systems

Acceleration data highly depends on the orientation of the smartphone. For example, holding a phone horizontally while moving forward produces different records than holding it vertically and performing the same movement. When moving a smartphone, the orientation of the coordinate system of the sensor is fixed, but in relation to the world coordinate system it is changing. In the present case, the reference coordinate system is the car's coordinate system. To match the car's coordinate system, every acceleration vector has to be transformed. The data set provides quaternions for the phone acceleration data. The quaternions are provided by the Android API's module: *Rotation Vector Sensor* (Google, 2017). The provided quaternions transform the acceleration vector to a geometric reference coordinate system. The geometric coordinate system is approximately pointing to the east (x-axis), to the geomagnetic North Pole (y-axis), and perpendicular to the ground plane upright to the sky (z-axis) (Google, 2017).

After transformation into world coordinate system, a vector still has to be reoriented to the coordinate system of the car. Therefore, GPS readings are used. With two consecutive GPS data points the angle of movement to geomagnetic north pole is computed. Hence, the so called *bearing angle* determines the direction of motion.

4.3.2 Compare Smartphone Acceleration to Car Acceleration

Figure 4.1 shows the rotated phone acceleration data. All values are in the car's coordinate system. The acceleration in x and y direction is compared to the longitudinal and the lateral acceleration data, respectively. In z direction the gravity ($9.81 m/s^2$) is subtracted. In the shown example, the bearing angle is not applied to x and y acceleration. Hence, the lateral and longitudinal acceleration are switched (on y and x axis).

Figure 4.2 shows a scheme of the coordinate systems. The coordinate system of the car is arranged, such that the z-axis points upwards orthogonal to the ground, the y-axis points in driving direction, and the x-axis points to the right in the direction of driving forward. Hence, longitudinal acceleration is applied to y-axis of the car and lateral acceleration is applied to the x-axis of the car. The data provides evidence, that the current car coordinate system is rotated by -90 degree around z-axis in reference to world coordinate system. Since the smartphone acceleration vectors are transformed into world coordinate system and longitudinal acceleration is applied in negative x direction and lateral acceleration in y direction. Moreover, there is evidence for the car taking a turn near timemilli 100,000, where the peak of lateral acceleration appears in the smartphone's x acceleration. Again, this is caused by the fact that the smartphone acceleration values are defined in world coordinate system and not referring to direction of travel. This is a good example for the need of the bearing angle, since the shown data segment is not comparable to a segment containing exactly the same pattern of acceleration, but driven in another direction. The lateral acceleration switching from y to x direction provides a first insight into this problem.

4.3.3 Sampling Rate

The time of recording between data points is not equidistant. Figure 4.3 shows the distribution of time distances. between recorded acceleration data points. (Das et al., 2010) state that is is caused by the Android API. A value is only saved on an *onSensorChanged()* signal, i.e. a sensor value is only stored on change. While this is reasonable to save memory space, it leads to inconsistent sampling rates.

To cope with the described problem, (Das et al., 2010) perform re-sampling, which changes the sampling rate of the data using interpolation. A rate is chosen and the data is fit to this rate using interpolation. (Das et al., 2010) suggest linear interpolation for smartphone acceleration data. Linear interpolation calculates a linear function between two data points and uses this function to compute the desired re-sampled point. The Nyquist-Shannon sampling theorem (Shannon, 1949) states that in order to reconstruct a signal sufficiently, the sampling rate has to be at least twice the frequency of the original signal, i.e., to re-sample the data present at 50

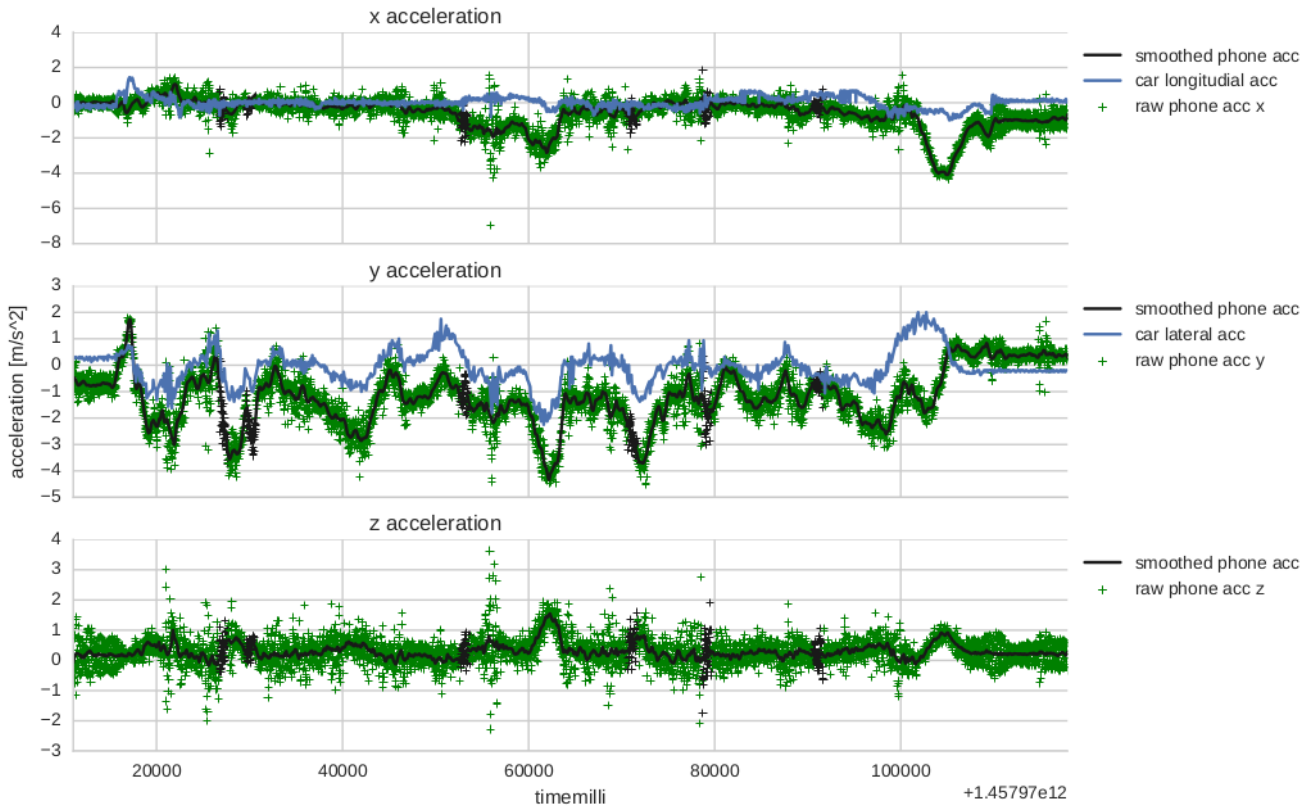


Figure 4.1: Plot of phone acceleration data in x, y, and z direction. In z direction, gravity is subtracted. The data is in the car's coordinate system. The x-axis shows the timestamps in milliseconds. The green markers are raw acceleration points, the black lines illustrate the smoothed acceleration data. The blue lines show corresponding data from the CAN bus.

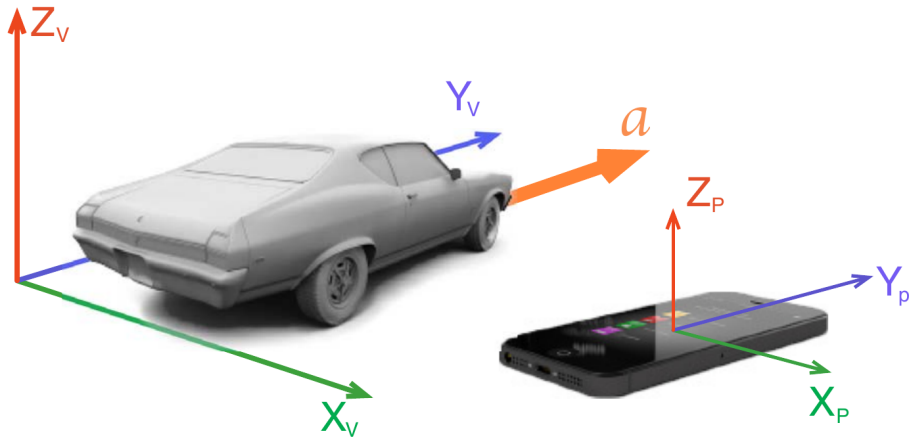


Figure 4.2: Illustration of car and smartphone coordinate system (Han et al., 2014)

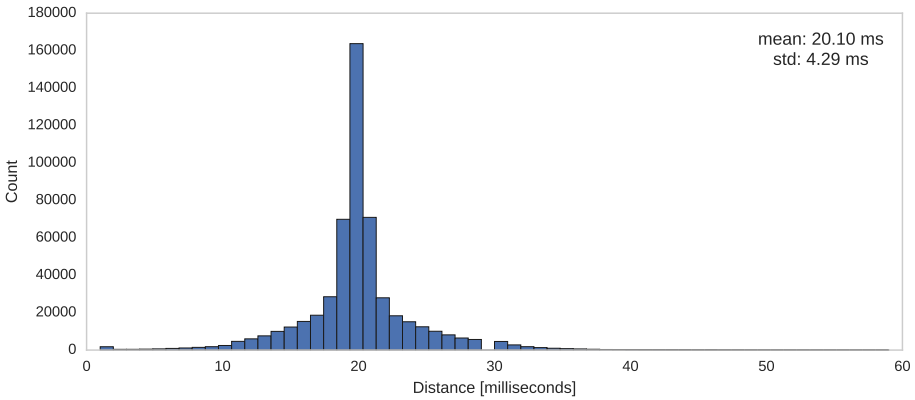


Figure 4.3: Distance distribution of the recorded smartphone acceleration data points in time. The duration between one recorded data point and the next one is calculated. The Figure depicts the histogram of all durations smaller than 60 ms, binned in 60 bins.

Hz without any information loss the new sampling rate has to be at least 100 Hz. (Das et al., 2010) use a sampling rate of 125 Hz for their smartphone acceleration data.

The drawback of re-sampling is the doubling of data. Without a gain of information, the data gets larger, due to the Nyquist-Shannon sampling theorem. In this project, the data is windowed. Considering the inconsistent sampling rate, there are possibly occurring different numbers of data points in the windows. Thus, the number of data points in windows varies. Preliminary experiments show that 96% of the windows include 50 ± 1 data points. Hence, for this project, the inconsistent sampling rate is not taken into account, but it is important to notice, especially for other use cases.

4.4 Challenges

The data was not pre-processed in forefront of this project. The following challenges occurred during pre-processing.

4.4.1 Merge Different Data Sources

Since all the sensors are recorded with different frequencies, the first challenge is to merge the data points of these sensors. Therefore, for every accelerometer data point a corresponding data point of the other sensors has to be found. Hence, the sensors' data is interpolated.

Interpolate Quaternions

Linear interpolation is used on the transformation vectors, i.e. the quaternions. These quaternions are linearly interpolated using SLERP (Spherical Linear intERPolation) (Shoemake, 1985). The implementation of the *pyquaternion*[2] package of SLERP is used. SLERP interpolates the ro-

[2] `http://kieranwynn.github.io/pyquaternion/` (March 11, 2017)

Figure 4.4: Comparison of rotated x-acceleration values. The upper subplot shows the acceleration values rotated with quaternions interpolated with nearest neighbor interpolation. The lower subplot shows the acceleration values rotated with quaternions interpolated with SLERP.

tation of a vector between two quaternions. Figure 4.4 shows the rotated acceleration values with SLERP and without SLERP, only using nearest neighbor interpolation. As is depicted, SLERP interpolation introduces additional noise to the data. SLERP is useful for one rotation between two quaternions but not for consecutive quaternion rotations (Shoemake, 1985). Since quaternions describe a rotation on a sphere, it is not specified in which direction the interpolation takes place. If the direction of interpolation is not specified, the interpolated value may not rotate the vector in the desired manner. As a consequence rotated vectors are jumping because of missing rotation direction. The result is depicted in Figure 4.4.

Furthermore, in a real system it is not possible to interpolate between the current quaternion and the succeeding one. Hence, nearest neighbor interpolation is used for the quaternions, which also leads to smoother data than SLERP.

4.4.2 Missing Data

The GPS sensor did not work as intended during several stages of recording for unknown reason. In some arbitrary periods, the GPS location is not changing for several seconds or even minutes. Since the GPS location is required for the bearing angle of the phone's movement (Section 3.2.2) this data is discarded. Data without corresponding bearing angle can not be properly preprocessed. This affects 15% of the smartphone acceleration data.

4.4.3 Sparse Event Occurrence

Systems, such as ABS, are only activated if needed. During a normal drive the quantity of ABS activations is very low. In the present data the average ABS activation number per day and driver is 0.76. This results in an unbalanced data set, since the amount of data points containing an 'on' label is very low.

The problem of the unbalanced classes can be addressed by setting the weight of the data points with an 'on' label higher than the 'off' labeled data points. Another solution is to down-sample the data. I.e. for every positive example 10, 100, or 1000 negative examples are randomly picked for the data set.

Further, it is important to notice that some evaluation metrics are not practical for an unbalanced class problem. For example, accuracy is not suitable. If there are 10,000 false and 10 true examples, predicting all data points as false leads to a accuracy of 99.99%, but this is not a desired result, if the goal is to classify all points correctly. Instead, metrics like the F-Score are more expressive. The F-Score does not measure the accuracy, but an combination of precision and recall. In the mentioned example, the F-Score is zero, so the classifier assigning all false has the lowest score, compared to the accuracy of almost 100%.

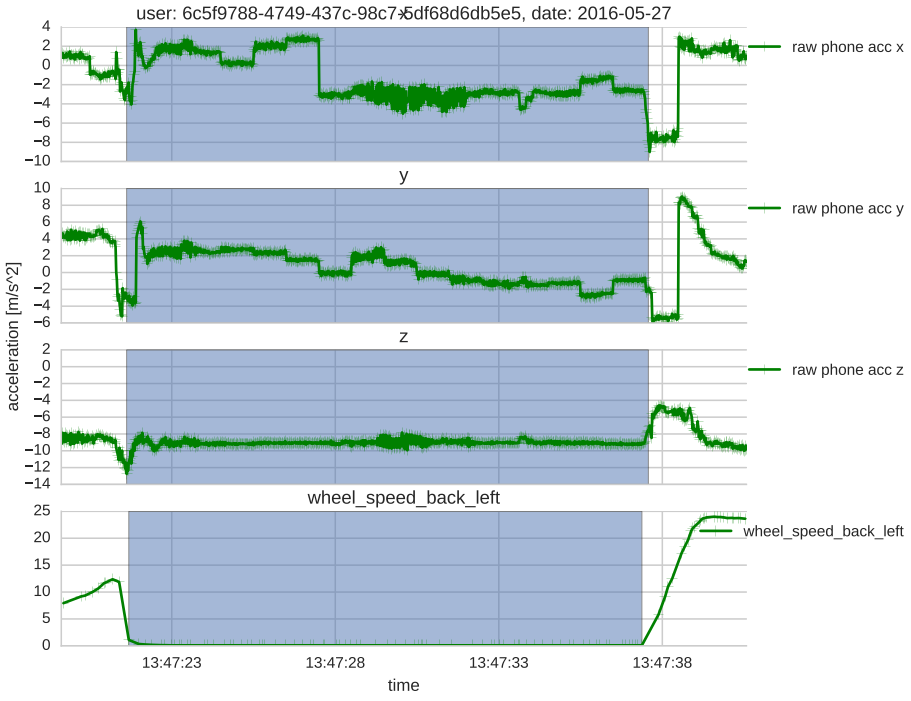


Figure 4.5: Wrongly marked ABS activation. The first three subplots show the x, y, and z acceleration of the phone rotated in car coordinate system. The green lines are acceleration values in m/s^2. The bottom subplot shows the speed of the back left wheel in km/h. The blue area marks the ABS activation as stored in the CAN bus data. Metadata of this example is ID: 6c5f9788-4749-437c-98c7-5df68d6db5e5, Time: 2016-05-27 13:47:20

4.4.4 Inexplicable Data

There is still some data that has to be filtered. An example is show in Figure 4.5. Here, the ABS activation is shown, which is for unknown reason marked incorrectly. It may be, that there was a problem during data recording. However, such data has to be filtered, since it influences the overall performance. As it can be seen in Figure 4.5, most of the labeled time there is no identifiable braking occurring To filter such data, other information has to be taken into account, such as the wheel speed of the vehicle. In this project, data is filtered if the average wheel speed is below 5 km/h for at least one second.

5 Signal Processing of Smartphone Acceleration Data

The previous chapter described the data used during the project and the challenges that arose. Before anything can be estimated by means of the smartphone acceleration data, the smartphone acceleration data has to be pre-processed. The steps of the pre-processing are described in the following chapter. Figure 5.1 depicts an overview of the data pre-processing pipeline.

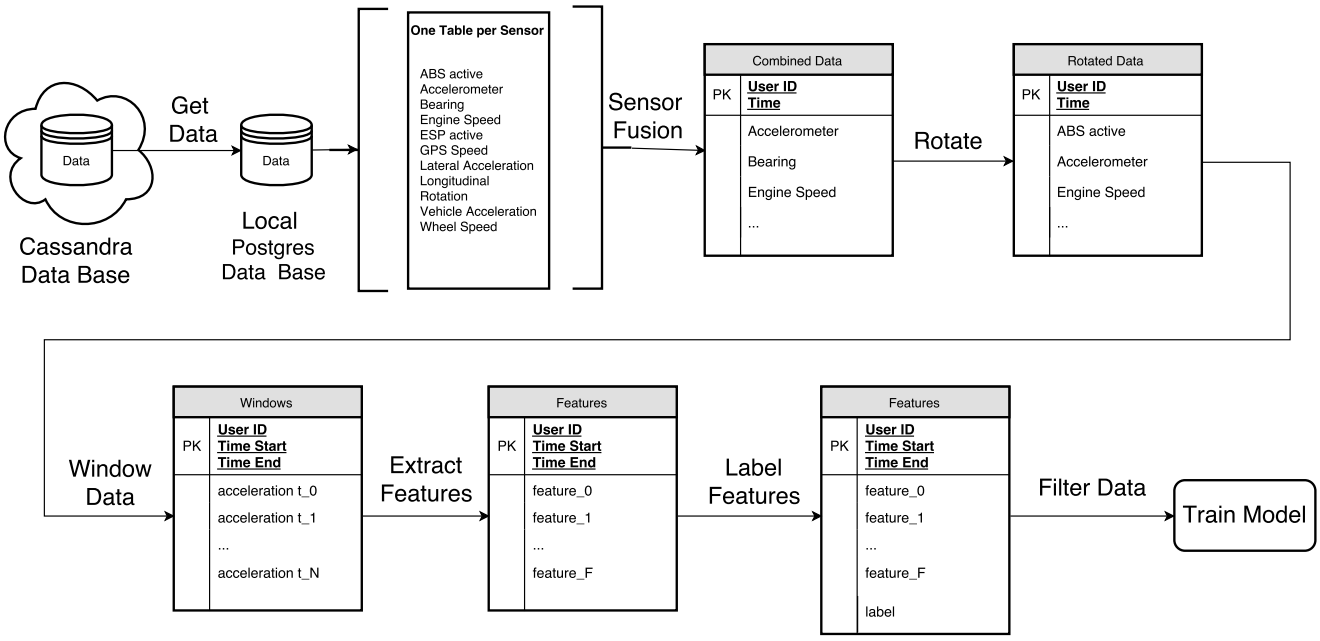
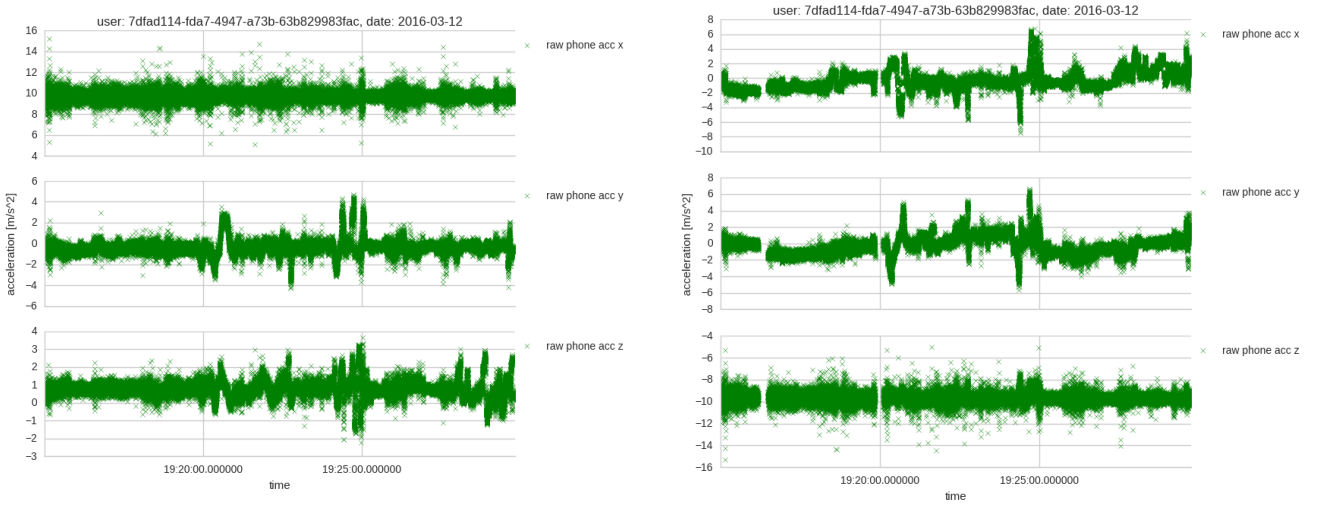


Figure 5.1: Data Pipeline

5.1 Sensor Fusion

As described in Section 4.3, the smartphone produces data by several sensors recorded with different frequencies. To process the data of those sensors, the data has to be merged. For every data point a time stamp is saved. This time stamp is used to merge the data. For every smartphone acceleration data point the most recent corresponding data point of every other sensor is identified.



(a) Before transformation

(b) After transformation

Figure 5.2: Unrotated and rotated smartphone acceleration data. (a) depicts the raw acceleration data recorded by smartphone and (b) shows the same data with quaternion transformation and bearing angle applied.

5.2 Rotation

The acceleration data, recorded by the smartphone, contains the acceleration related to the smartphone itself. Since during the recording of the data the smartphone could be placed in any arbitrary direction, one can not directly derive the car's acceleration. Therefore, the smartphone data has to be rotated, such that it is in the coordinate system of the car (see Section 3.2).

Transformation quaternions, transforming smartphone data from smartphone coordinate system to world coordinate system, and the bearing angle, transforming from world coordinate system to car coordinate system, are available. Figure 5.2 depicts raw smartphone acceleration data and the same data with applied transformation. As indicator of working transformation, the z acceleration can be considered. Only gravity should influence the acceleration in z direction. In the illustrated example, gravity appears in positive x direction. In contrast, after transformation gravity appears in negative z-direction as desired. In the data after transformation, gaps arise. These gaps are caused by missing bearing angles, since GPS was not working and without GPS the bearing angle is not computable. Hence, this data is filtered.

5.3 Windowing

A single data point itself does not contain enough information to learn anything, since the data is continuous. To identify the characteristics of the continuous signal, such as changes in gradient or frequency, ranges of data points have to be considered. Thus, the data needs to be grouped

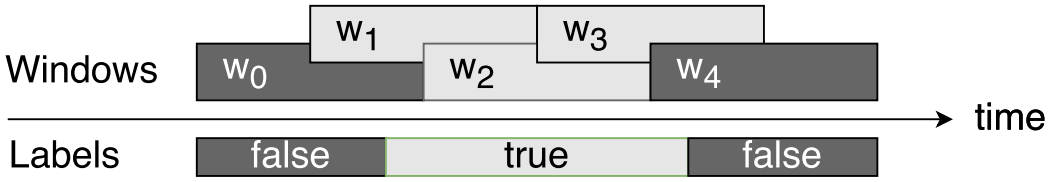


Figure 5.3: Example for window - label problem. Light gray windows labeled true, dark windows labeled false. True, false assignments of the windows are exemplary.

and is, therefore, split into windows. Using the time stamps of the data points, the data is joined to overlapping windows of a specific duration. The windows are overlapping, since otherwise characteristics located at the border of two windows get lost. Inside these windows, statistical values, such as the mean or standard deviation, are computed to describe the window.

5.4 Feature Extraction

After the data is split into windows, characteristic values that describe the particular window are computed. These values are called *features*. For every window, the features are computed and composed into a feature vector. Hence, the windows are transformed from time space into feature space. The feature vectors are then used for the learning procedures.

5.5 Labeling

After the features are extracted for the windows, the windows have to be labeled. The information of the labels comes from the CAN bus data. In the CAN bus data, the true signals of the car's sensors are saved, for example, when the ABS system was activated. If ABS is activated an 'on' signal is recorded with corresponding time stamp. When ABS is deactivated again an 'off' signal is recorded. This information is used to label the windows.

When a continuous signal is split into windows and it needs to be labeled, it happens that a part of the window is in the true label and another part of the window is in the false label. Figure 5.3 sketches the problem. True and false assignments of the windows are exemplary shown in the figure. In the example about 80% of window w_1 contains true labeled time and window w_0 contains only 5% of true labeled time. Hence, w_0 is labeled as false and w_1 is labeled as true. In this example the minimum amount of true labeled time is set to 10%. To label a window as true, a specific amount of time of the window needs to be inside the true labeled time. This amount of time is a free parameter.

5.6 Filter Data

The transformation of the smartphone acceleration data (Section 5.2) does not work properly at some points. The transformation of the acceleration vectors relies on three different sensors (gy-

roscope, magnetometer, and GPS), which all introduce noise to the resulting signal. Especially the magnetometer, which is very important for the three dimensional vector transformation from phone coordinate system to world coordinate system, is very sensitive for surrounding electro magnetic waves (Abbott and Powell, 1999). Furthermore, the sensors are all recorded in different frequencies, all of them slower than the accelerometer values, which causes drift to the data.

To overcome the problem of coping with improperly rotated data, the mean value of the z-axis is considered for a first filtering. Since a car is not expected to fly or fall, i.e. to have high accelerations on the z-axis, all windows of which the mean z-acceleration value differs more than 10% from actual gravity (9.81 m/s^2) are removed from the data set.

Even if vector transformation is successful, there is data that is not desired to be in the data set. As depicted in Figure 4.5 on page 33, smartphone acceleration data can is recorded even when the vehicle did not move. Since for this project only events of a moving car are investigated, data that is recorded while the car did not move is filtered. Hence, the wheel speed is taken into account. If the average wheel speed in a window is below 5 km/h, the window is removed from the data set.

6 Event Detection

The previous chapter introduces the pre-processing steps that are applied to the data. After pre-processing, the data is used for inference. This chapter describes the experiments performed during this project. First, the features used for classification are presented. Subsets of the overall data are used for the experiments which are introduced in the second part of the chapter. In the third part, the experiments are introduced. Lastly, results are discussed.

6.1 Features

For the classification tasks, features are extracted from the continuous data. Therefore, first the data is split into windows of one second, with half a second overlap. Then, for every window the features are computed.

First, the magnitude (Equation (3.12)) of the acceleration is calculated. Then, the jerk of the magnitude and the accelerations in x, y, and z direction are computed. For each of those (jerk(magnitude), jerk(x), jerk(y), jerk(z), magnitude, x, y, z) the following values are calculated:

- minimum
- maximum
- mean
- standard deviation
- mean crossing rate
- maximum − mean
- mean − minimum
- maximum − minimum

This results in (8 × 8) 64 features for every window.

6.2 Data Subsets for Analysis

The implementation and the analysis is performed on subsets of data. For debugging purposes it is assumed that all the data fits into memory and can be computed in a reasonable amount of time.

Three subsets are gathered from the data.

One-User-Set

One set contains data of one user over 13 days. The intension is to check, whether the models generalize over time for one user. The ID of this user is "7dfad114-fda7-4947-a73b-63b829983fac" and the days for the data set are between the 12th of March, 2016 and the 28th or March, 2016. On four days of this period, there is no data of this driver. This data set contains 5,435,567 smartphone acceleration data points and a driving duration of about 30 hours. Which results in about 216,000 feature vectors of 1 second windows with an overlap of 0.5 seconds. In the following this data set is called *One-User-Set*.

Many-User-Set

The other set contains the data of 7 users with data of one or two days of each user. The users and dates are picked randomly, the only restriction is that there is any data on a particular day. A list of users and corresponding dates used for this data set is shown in Table 6.1. This data set contains 6,116,371 smartphone acceleration data points and a driving duration of 34 hours and 20 minutes. This results in about 245,000 feature vectors. In the following this data set is called *Many-User-Set*.

ID	Date
7dfad114-fda7-4947-a73b-63b829983fac	06/21/2016
7dfad114-fda7-4947-a73b-63b829983fac	06/29/2016
444d6e50-30ef-4d5f-b90f-764ba6013d70	03/05/2016
56badd13-1ed2-478b-95a0-673383d6e600	06/19/2016
11ce1604-6182-4231-bac3-3b083d0aaa05	02/25/2016
11ce1604-6182-4231-bac3-3b083d0aaa05	06/04/2016
b02531b8-3e40-4bb5-a763-7973503da1c1	03/22/2016
9142a8ec-d074-4a5c-a680-563f2dff7aae	03/10/2016
9142a8ec-d074-4a5c-a680-563f2dff7aae	05/13/2016
02e05e12-7ddf-4b7b-8fd9-37059c6be4ad	03/30/2016

Table 6.1: List of user IDs and corresponding dates used for the *Many-User-Set*.

Test-Track-Set

All the data is tagged with GPS coordinates. From these coordinates it is evident, that some of the data was recorded during a drive on a test track. Data from a test track may have several disadvantages for this project. For example, the the car's systems have potentially been deactivated during drive for demonstration purposes. It is not possible to deduce from the data if a system in the car is deactivated by user. But, test track data also has advantages. On a test track, the driving behavior is more extreme. A more extreme driving behavior leads to

more distinguishable data. Moreover, on a test track the safety systems of a car should be demonstrated. Consequently, the number of activations is much higher than during a normal drive. For this data set, the data of one driver of one day during a drive on a test track is used. The user ID of the user is "52707bc6-fc83-49fd-97cb-eb7fe7479962" and was recorded on the 22nd of March, 2016. It contains 814,628 smartphone acceleration data points and a driving duration of about 3 hours and 30 minutes. This results in 25,687 feature vectors. In the following this data set is called *Test-Track-Set*.

6.3 Events

In this section, the individual experiments are described. The subsections are organized in description, implementation, results, and interpretation.

6.3.1 Anti-lock Braking System Events

This section is concerned with estimating whether the ABS system is active or not during a normal drive. Therefore, the data is split into windows of one second duration with an overlap of half a second and the features are extracted, as described in Section 6.1.

This is a binary classification task. Whereby the inputs are the features calculated by the smartphone acceleration data and the output is the state of the ABS system ({'on', 'off'}).

The One-User-Set contains 26 ABS events, which results in 91 of 216,000 feature vectors labeled as 'on'. The Many-User-Set contains 46 ABS events and 188 of 245,000 feature vectors are labeled as 'on'. The Test-Track-Set contains 94 ABS events and 389 of 25,687 feature vectors are labeled as 'on'.

Implementation

The data is pre-processed and split into windows as described in Chapter 5. The windows are labeled using the CAN bus data. Whereas, every window is labeled as 'on' if at least 10% of the window's time is part of the labeled time stamp. The amount of labeled time in a window may be an additional parameter to optimize.

To ensure that there are no overlapping windows, train and test set are split by user and day. For the One-User-Set the train set contains seven days and the other six days are in the test set. The Many-User-Set is split similar. Here, two users and their data of one day each are in the test set. For the Test-Track-Set a test time is specified. For cross-validation the separate folds are not further specified, which means that there may be overlapping windows in two different folds.

The windows are filtered due to the mean_z value of the feature vectors, to cope with wrongly transformed acceleration vectors (Section 5.6). Table 6.2 illustrates the number of feature vectors in train and test set after filtering.

Data Set	Train	Test
One-User-Set	28	8
Many-User-Set	86	39
Test-Track-Set	40	24

Table 6.2: Number of feature vectors labeled as ABS 'on' in the train and the test set for the particular data sets after filtering

Three classification algorithms are compared, which are used in literature for similar tasks using smartphone acceleration data: Random Forest (Daptardar et al., 2015; Cervantes-Villanueva et al., 2016), Decision Tree (Lu et al., 2010), and SVM (Chu et al., 2014; Wu et al., 2016). The code is written in Python 2.7. The scikit-learn 18.1 (Pedregosa et al., 2011) implementations of the classifiers are used for the experiments.

First, the parameters are optimized by cross-validation grid search on the train set with F_1 score as target value. The test set is not used for optimization, it serves as a hold out set to evaluate the quality of the model. For the SVM the parameter optimization is performed on a subset of the data to reduce the computation time of the problem and fasten the optimization. The subset contains all the 'on' examples of the train set and a sample of the 'off' examples.

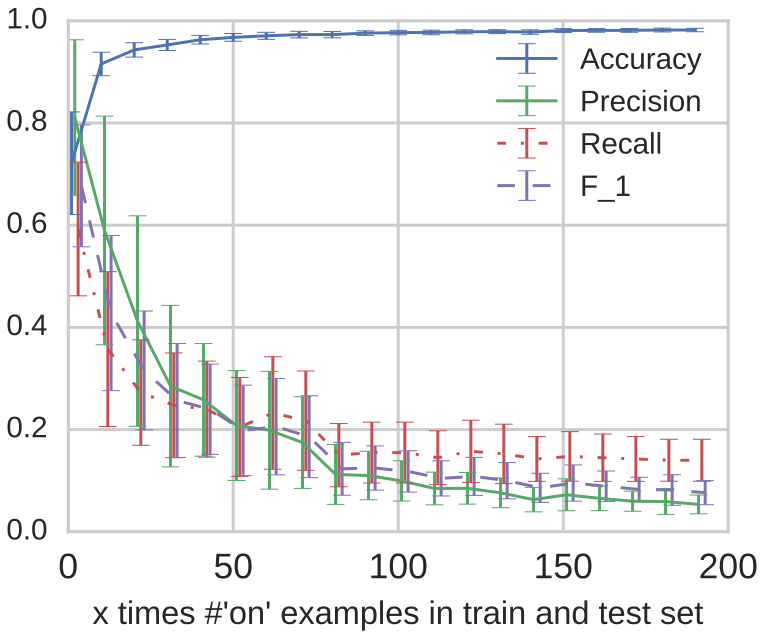


Figure 6.1: Metric values of ABS event detections with different proportion of true / false labels on One-User-Set. The x-axis indicates the amount of 'off' labeled feature vectors in relation to 'on' labeled vectors. For example, 20 on the x-axis denotes for every 'on' labeled feature vector that there are 20 'off' labeled vectors. Every amount is executed 50 times with different 'off' labeled samples in training and test set. The lines show the mean of these 50 trials and the bars the standard deviation. The lines are shifted for purpose of presentation.

Since this is an unbalanced classification problem the calculations of the metric values is performed on a subset of the data to reduce the imbalance. Hence, the data sets are sub-sampled, such that for every 'on' example a certain number of 'off' examples are sampled from the data. First, the different ratios of 'off' samples are tested on the One-User-Set and is chosen such that there is a good trade-off between precision and recall. Since in this project precision and recall are equally important, a good trade-off is defined as the point, where precision and recall are identical. For improved comparability, the ratio is optimized for one data set and used for all estimations. The gained ratio is used for the computations of the metrics. Whereas, every estimation is performed 100 times with different samples of the 'off' examples in train and test set. But, for every algorithm and every run the sampled data points are always the same, by setting the *Random Seed* and iterating it from 1 to 100.

Results

Tables 6.3 to 6.5 show the optimized parameters for the particular algorithms. The models are trained using the shown parameters.

The presented results are calculated on the test set. Table 6.6 shows an overview of the metric values for the data sets. The values in the table are the mean values of 100 runs of training and testing on different random sub-samples. Every run is performed on the 'on' labeled features and for each 'on' labeled feature vector 40 'off' labeled feature vectors are sampled from the data set. The ratio of 40 to one is chosen based on the comparison shown in Figure 6.1. At the ratio of 40 the mean of precision and recall are equal. Without sub-sampling, the metric values are almost zero with the exception of the accuracy which is almost one because of the imbalance, since assigning all data points to the 'off' class gives the best accuracy, but zero F1 score.

Figure 6.1 shows the tested ratios of 'off' examples to 'on' examples. It shows that precision, recall, and F1 score are decreasing with increasing ratio of 'off' labeled feature vectors, whereas, the accuracy converges to one.

The results in Table 6.6 show that the estimations work best for the Test-Track-Set and worst for the Many-User-Set. The accuracy is very high for all test runs, as expected for an unbalanced classification problem. The best result is gained on the Test-Track-Set using the Random Forest and the worst results on the Many-User-Set using a SVM. In general Random Forest performs best.

Dataset	max_depth	max_features	max_leaf_nodes	min_impurity_split	min_samples_leaf	min_samples_split	min_weight_fraction_leaf	n_estimators	class_weight
One-User-Set	4	16	None	10^{-7}	1	2	0	10	balanced
Many-User-Set	4	4	None	10^{-7}	8	2	0	100	balanced
Test-Track-Set	None	16	None	10^{-7}	16	2	0	10	balanced

Table 6.3: Optimized parameters - ABS detection - Random Forest

Dataset	max_depth	max_features	max_leaf_nodes	min_impurity_split	min_samples_leaf	min_samples_split	min_weight_fraction_leaf	presort	class_weight
One-User-Set	11	auto	None	10^{-7}	1	2	0	False	balanced
Many-User-Set	9	None	None	10^{-7}	1	2	0	False	balanced
Test-Track-Set	None	16	None	10^{-7}	16	2	0	False	balanced

Table 6.4: Optimized parameters - ABS detection - Decision Tree

Dataset	C	kernel	decision_function_shape	class_weight
One-User-Set	1.9	rbf	ovr	balanced
Many-User-Set	1.5	rbf	ovr	balanced
Test-Track-Set	0.5	rbf	ovr	balanced

Table 6.5: Optimized parameters - ABS detection - SVM

Figure 6.2 shows the distribution of the computed feature vectors. For every feature the mean and standard deviation values are shown. It shows that on Test-Track-Set the mean values are best distinguishable. On the Many-User-Set all feature values have overlapping standard deviations and only some values have distinguishable mean values. The One-User-Set has few distinguishable mean values but for those, the standard deviation is not overlapping as much as on the Many-User-Set.

Data Set	Method	Accuracy	Precision	Recall	F1-Score	TP (P)	FP (N)
One-User-Set	RF	0.92	0.6	0.36	0.43	1.9 (8)	6.1 (320)
	DT	0.96	0.18	0.11	0.13	0.9 (8)	5.4 (320)
	SVM	0.90	0.39	0.21	0.26	0.02 (8)	5.6 (320)
Many-User-Set	RF	0.90	0.10	0.40	0.16	15.7 (39)	144.5 (1560)
	DT	0.82	0.06	0.37	0.10	14.6 (39)	235.8 (1560)
	SVM	0.89	0.01	0.04	0.02	1.4 (39)	136.7 (1560)
Test-Track-Set	RF	0.98	0.52	0.86	0.65	20.6 (24)	19.6 (960)
	DT	0.98	0.58	0.68	0.62	16.5 (24)	12.3 (960)
	SVM	0.98	0.53	0.63	0.58	15.2 (24)	13.5 (960)

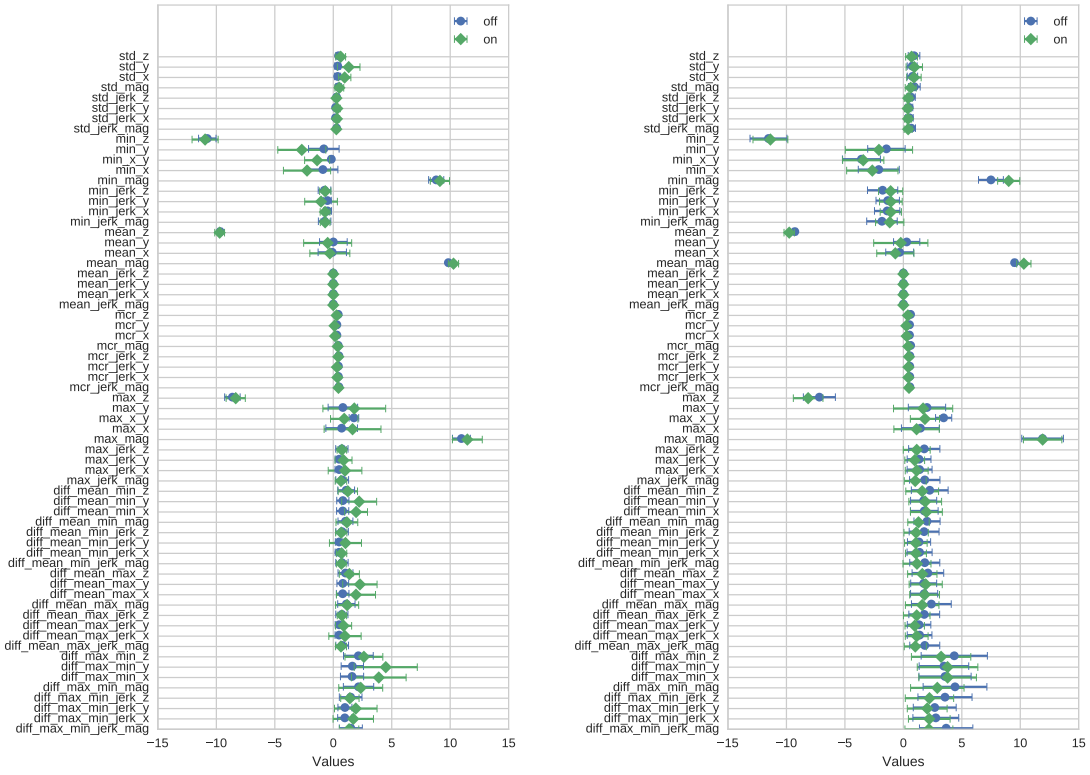
Table 6.6: Metric values on test set of ABS detection experiments. Values are mean values of 100 random sub-samples of the train and the test data set, where for every 'on' example 40 'off' examples are randomly sampled from the particular data set. In the last two columns, TP denote the number of true positives, i.e. correctly as 'on' classified feature vectors, P the number of all 'on' labeled feature vectors in the test set. FP (False Positive) denotes the number of incorrectly 'on' labeled feature vectors and N the number of the 'off' labeled feature vectors in the test set.

Figure 6.3 shows an ABS activation during normal drive and on test track. Both subplots show a duration of 10 seconds. On the Test-Track-Set example, smartphone acceleration data during ABS activation clearly sets apart from the surrounding data points. On normal drive, the brake event is already lasting for some time until ABS gets activated. The smartphone acceleration data during ABS activation is not distinguishable from non-ABS data in this normal drive example.

Interpretation

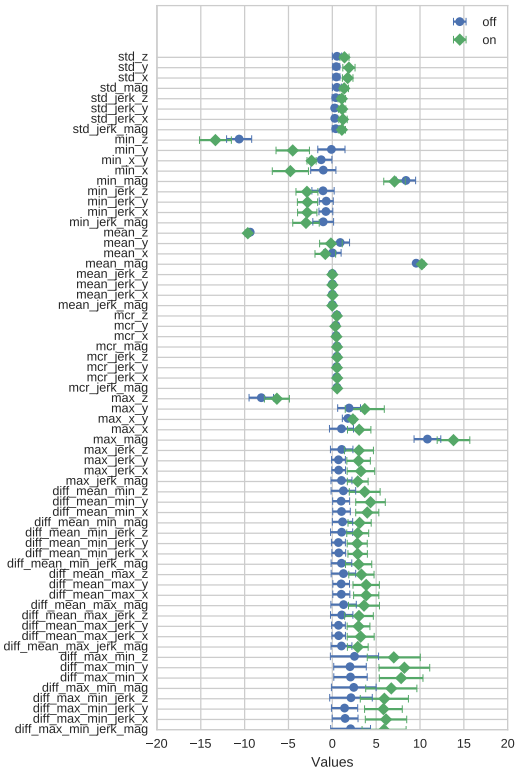
The results show that it is possible to estimate ABS events by use of smartphone acceleration data, but it does not perform well in the setting of this project.

The classification works only on sub-samples of the data. Since the classification problem is very unbalanced, even high weights on the underrepresented class examples do not improve performance. Artificially lowering the number of 'off' examples improves the classification performance significantly as it is depicted in Figure 6.1. Table 6.2 highlights the unbalance,



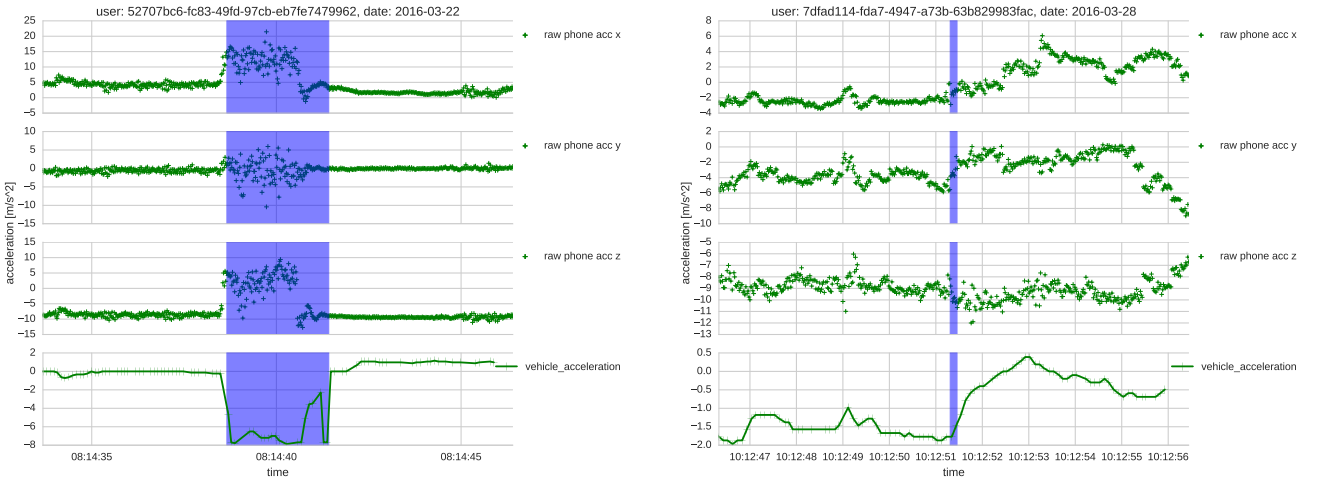
(a) One-User-Set

(b) Many-User-Set



(c) Test-Track-Set

Figure 6.2: Comparison of feature value distributions of ABS off and on labeled feature vectors on the data sets. The features are on the y axis and their values on the x axis. The markers show the mean value and the bars the standard deviation.



(a) Test-Track-Set **(b)** One-User-Set

Figure 6.3: Example ABS activation on the test track (a) and during a normal drive (b). The green markers and line depict the acceleration in x, y, and z direction of the smartphone and the true vehicle acceleration. The blue shaded area marks the true ABS activation duration.

considering the number of feature vectors labeled as 'on' and the overall available feature vectors.

The values in Table 6.6 show, that the classification generalizes over time but not over users. On the Many-User-Set the highest F_1 score is 0.16 on the Random Forest. The Test-Track-Set performs best with all algorithms. On a test track the driver attempted extreme driving behavior. This circumstance is good for the classification. It is expected that ABS is activated during extreme situations. The data shows, during a normal drive ABS is also activated while steady decelerating, which is hard to detect using smartphone acceleration data. Figure 6.3 shows that in this example the collected data on the test track appears to be much more distinguishable from the surrounding data than during normal drive. On the normal drive the ABS gets activated at the end of the braking event. With smartphone acceleration data, only the changes in acceleration can be incorporated for classification. Figure 6.3(b) shows there is no evidence that ABS activations can be detected by acceleration. For example, vehicle acceleration at 10:12:47 compared to vehicle acceleration during ABS activation has same shape and so do the x and y acceleration of the smartphone. Hence, it is not in general possible to detect ABS activation.

6.3.2 Electronic Stability Control Events

Similar to the ABS event detection, it is tried to estimate ESC events. The data is split into windows of one second duration with an overlap of half a second and the features are extracted, as described in Section 6.1. ESC event detection is a binary classification task. The inputs of the

classifier are the feature vectors calculated using the windows of the smartphone acceleration data and the output is the state of the ESC system ({'on', 'off'}).

The One-User-Set contains 19 ESC events, which results in 50 of 216,000 feature vectors labeled as 'on'. The Many-User-Set contains 12 ESC events and 36 of 245,000 feature vectors are labeled as 'on'. The Test-Track-Set contains 50 ESC events and 267 of 25,687 feature vectors are labeled as 'on'.

Implementation

The implementation is based on the implementation of the ABS event detection.

The data is pre-processed and split into windows as described in Chapter 5. The windows are labeled using the CAN bus data. Whereas, every window is labeled as 'on' when at least 10% of the windows time is part of the labeled time stamp. If the approach works in general, the amount of labeled time in a window may be an additional parameter to optimize

To ensure that there are no overlapping windows, train and test set are split by user and day. In the One-User-Set the train set contains seven days and the other six days are in the test set. The Many-User-Set is split similar. Here, two users and their data of one day each are in the test set. For cross-validation the separate folds are not further specified, which means that there may be overlapping windows in two different folds.

Three classification algorithms are compared, that are used in literature for similar tasks using smartphone acceleration data: Random Forest (Daptardar et al., 2015; Cervantes-Villanueva et al., 2016), a Decision Tree (Lu et al., 2010), and a SVM (Chu et al., 2014; Wu et al., 2016). The code is written in Python 2.7. The scikit-learn 18.1 (Pedregosa et al., 2011) implementations of the classifiers are used for the experiments.

The windows are filtered due to the mean_z value of the feature vectors, to cope with wrongly transformed acceleration vectors (Section 5.6). Table 6.7 illustrates the number of feature vectors in train and test set after filtering.

Data Set	Train	Test
One-User-Set	11	20
Many-User-Set	22	2
Test-Track-Set	50	36

Table 6.7: Number of feature vectors labeled as ESC 'on' in the train and the test set for the particular data sets after filtering

Train and test set are split by day and user, to avoid overlapping windows to be included in test and train set. Combined with the low occurrence of ESC events, train and test splits can be unbalanced.

First, the parameters are optimized by cross-validation grid search on the train set with F_1 score as target value. The test set is not used for the optimization. For the SVM the parameter optimization is performed on a subset of the data to reduce the complexity of the problem and fasten the optimization. The subset contains all the 'on' examples of the train set and a sample of the 'off' examples.

Since this is a unbalanced classification problem the calculations of the metric values are performed on a subset of the data to reduce the imbalance. Hence, the data sets are sub-sampled, such that for every 'on' example a certain number of 'off' examples are sampled from the data. For improved comparability, the ratio of 40 to one is chosen as for the ABS event detection.

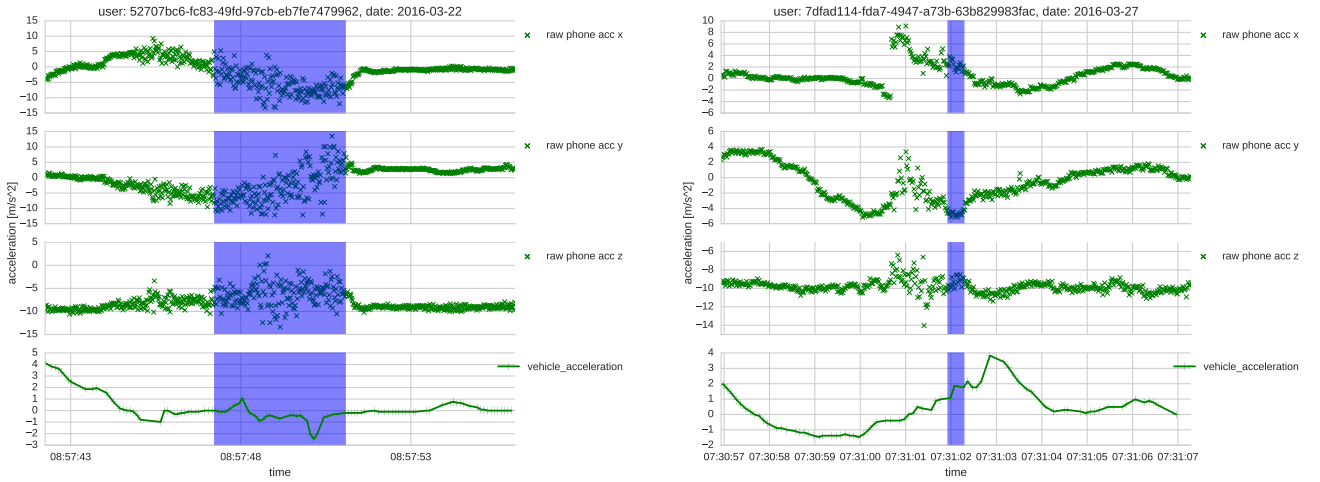
Results

Tables 6.9 to 6.11 show the optimized parameters for the particular algorithms. The models are trained using the shown parameters.

The presented results are calculated on the test set. Table 6.8 shows an overview of the metric values for the data sets. The values in the table are the mean values of 100 runs of training and testing on different random sub-samples. Every run is performed on the 'on' labeled features of the data set and for every 'on' labeled feature vector 40 'off' labeled feature vectors are sampled from the data set as for ABS event detection. Without sub-sampling the data the metric values are almost zero with exception of accuracy which is almost one because of the imbalance, since assigning all data points to 'off' class gives best accuracy but zero F1 score.

Data Set	Method	Accuracy	Precision	Recall	F1-Score	TP (P)	FP (N)
One-User-Set	RF	0.98	0.49	0.33	0.38	6.6 (20)	6.1 (800)
	DT	0.97	0.24	0.22	0.22	4.3 (20)	11.4 (800)
	SVM	0.96	0.03	0.02	0.02	0.4 (20)	10.9 (800)
Many-User-Set	RF	0.98	0	0	0	0 (2)	0.08 (80)
	DT	0.97	0.25	0.17	0.19	0.2 (2)	0.8 (80)
	SVM	0.98	0	0	0	0 (2)	0.3 (80)
Test-Track-Set	RF	0.99	0.66	0.82	0.73	29.4 (36)	15.4 (1440)
	DT	0.98	0.68	0.71	0.69	25.5 (36)	12.1 (1440)
	SVM	0.98	0.64	0.70	0.67	25.1 (36)	14.1 (1440)

Table 6.8: Metric values on test set of ESC detection experiments. Values are mean values of 100 random sub-samples of the train and the test data set, where for every 'on' example 40 'off' examples are randomly sampled from the particular data set. In the last two columns, TP denote the number of true positives, i.e. correctly as 'on' classified feature vectors, P the number of all 'on' labeled feature vectors in the test set. FP (False Positive) denotes the number of incorrectly 'on' labeled feature vectors and N the number of the 'off' labeled feature vectors in the test set.



(a) Test-Track-Set **(b)** One-User-Set

Figure 6.4: Example ESC activation on test track (a) and during a normal drive (b).The green markers and line depict the acceleration in x, y, and z direction of the smartphone and the true vehicle acceleration. The blue shaded area marks the true ABS activation duration.

The values in Table 6.8 show that the approach works best on the Test-Track-Set. It does not work on the Many-User-Set, here, the results are almost all zero. Only the decision tree generalizes a bit over the users with optimized parameters. On the One-User-Set Random Forest performs best and SVM achieves a score of almost zero. The accuracy is always almost one.

Figure 6.5 shows the distribution of the computed feature vectors. For every feature, the mean and standard deviation values are shown. It shows that on Test-Track-Set and One-User-Set the mean values are distinguishable, whereas, on the Test-Track-Set the mean value of more features are separated than on One-User-Set. On the Many-User-Set all feature values have overlapping standard deviations and only some values have distinguishable mean values.

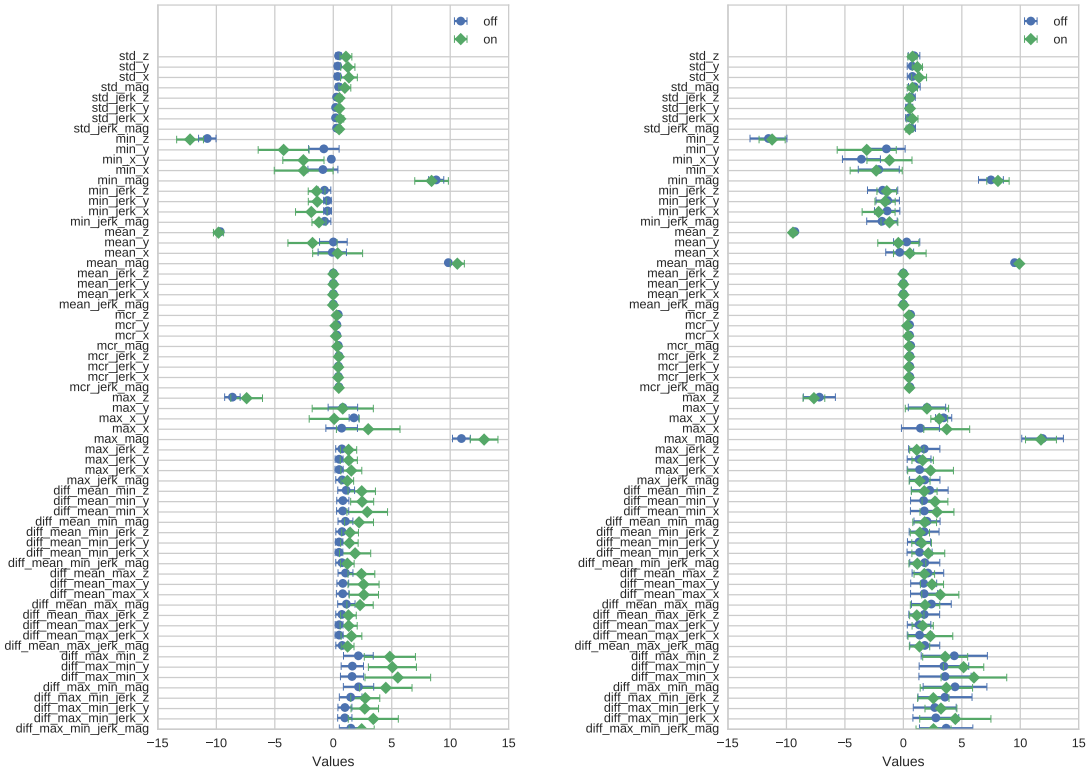
Figure 6.4 depicts one ESC activation during normal drive and one activation on test track. On the Test-Track-Set example the smartphone acceleration data during ESC activation clearly sets apart from the surrounding data points. The vehicle acceleration of these two examples shows that ESC may be activated during acceleration and during deceleration.

Interpretation

The results are similar to ABS detection results. Again, all algorithms perform best on Test-Track-Set and worst on Many-User-Set. On Many-User-Set the algorithms are not capable to classify ESC events. Only the Decision Tree is able to determine at least some ESC activations. Taking the feature distributions from Figure 6.5 into account this trend can also be seen. The mean values of the features of the Many-User-Set are worst separable in comparison to the distributions of the other data sets.

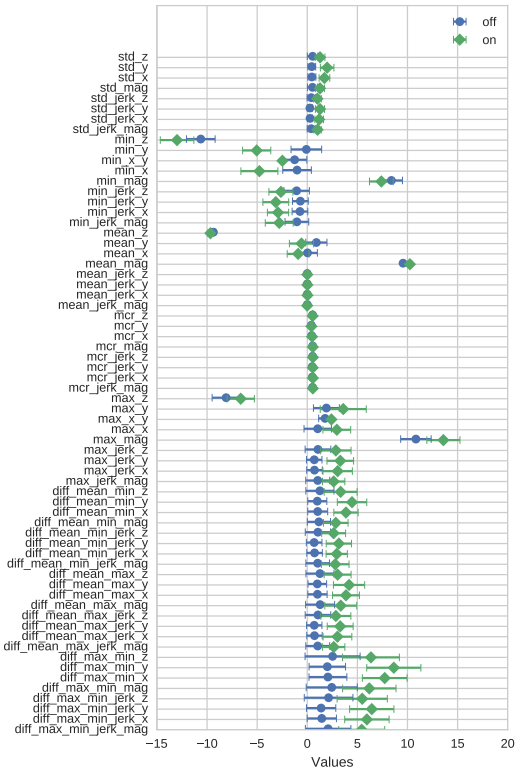
In contrast to ABS, which is activated mostly during brake events, ESC is activated in acceleration and deceleration situations as 6.4 shows. ESP is expected to be activated during lateral accelerations, which is present in the shown examples. Especially in 6.4a the acceleration is changing on x and y axis, whereas it stays constant in z direction. Only the standard deviation of the data is changing in z direction. In 6.4b the deviation of data is not changing during ESC activation.

The feature distribution shown in 6.5 reflects the results. Best distinguishable are the features of the Test-Track-Set, and worst distinguishable are features of Many-User-Set. Also the expected activation during lateral acceleration is illustrated in the feature distribution. The highest distance of means between the classes is present on the features in y direction. Also x directional features are separated similar good. Which is caused by imperfect transformation from smartphone coordinate system to car coordinate system..



(a) One-User-Set

(b) Many-User-Set



(c) Test-Track-Set

Figure 6.5: Comparison of feature value distributions of ESC 'off' and 'on' labeled feature vectors on the data sets. The features are on the y axis and their values on the x axis. The markers show the mean value and the bars the standard deviation.

Dataset	max_depth	max_features	max_leaf_nodes	min_impurity_split	min_samples_leaf	min_samples_split	min_weight_fraction_leaf	n_estimators	class_weight
One-User-Set	4	8	None	10^{-7}	1	2	0	10	balanced
Many-User-Set	4	16	None	10^{-7}	16	2	0	100	balanced
Test-Track-Set	None	4	None	10^{-7}	8	2	0	10	balanced

Table 6.9: Optimized parameters - ESC detection - Random Forest

Dataset	max_depth	max_features	max_leaf_nodes	min_impurity_split	min_samples_leaf	min_samples_split	min_weight_fraction_leaf	presort	class_weight
One-User-Set	7	None	50	10^{-7}	1	2	0	False	balanced
Many-User-Set	7	5	50	10^{-7}	1	10	0	False	balanced
Test-Track-Set	12	5	50	10^{-7}	1	2	0	False	balanced

Table 6.10: Optimized parameters - ESC detection - Decision Tree

Dataset	C	kernel	decision_function_shape	class_weight
One-User-Set	1.7	rbf	ovr	balanced
Many-User-Set	1.9	rbf	ovr	balanced
Test-Track-Set	1.0	rbf	ovr	balanced

Table 6.11: Optimized parameters - ESC detection - SVM

Dataset	max_depth	max_features	max_leaf_nodes	min_impurity_split	min_samples_leaf	min_samples_split	min_weight_fraction_leaf	presort
One-User-Set	3	None	None	10^{-7}	5	10	0	False
Many-User-Set	7	None	None	10^{-7}	5	2	0	False
Test-Track-Set	9	None	None	10^{-7}	5	10	0	False

Table 6.12: Optimized parameters - Vehicle Acceleration - Regression Tree

6.3.3 Vehicle Acceleration

The actual acceleration of the car is inferred. This is a continuous regression task, where the inputs are the smartphone acceleration data points and the output is the acceleration of the vehicle. As an error measure, the mean squared error is used.

Implementation

In contrast to estimations described above (ABS and ESP), the data is not windowed in forehand, since it is tried to estimate a continuous signal. Instead, the data is smoothed with a moving average of 50 data points, which is approximately one second at a frequency of 50 Hz. For the estimations, a moving window is used, whereby, every data point is described by itself and 25 prior data points. Additionally, the mean and standard deviation of the moving window are appended to the vector. The vectors, that are constructed in the described manner, are applied to a linear regression algorithm and a regression tree.

For linear regression, the *LinearRegression* implementation of scikit-learn 18.1 (Pedregosa et al., 2011) is used and as regression tree the *DecisionTreeRegressor* of the same package is utilized. The parameters are optimized using cross validation grid search.

As the Baseline, the test values are all predicted to be the mean vehicle acceleration of the train set.

First, the regression is trained on a small slice of 10 seconds of data and tested on the same slice. Note, that this does not state anything on the quality of the regression. Thus, it can be checked whether it is possible at all to approximate the vehicle acceleration with the smartphone acceleration by a linear regression.

Results

Table 6.12 on page 53 shows the optimized parameters for the particular data sets and models.

Figure 6.6 depicts an example slice of vehicle acceleration data. The figure shows the actual vehicle acceleration values (blue dotted line) and the estimated values (green markers). Note, for this plot the estimator was trained and tested on the presented slice. There is no statement on the quality of the estimation. However, this figure shows that there is a linear relationship between smartphone acceleration data and vehicle acceleration in general, since otherwise the linear regression could not be able to adapt to the data. It has still to be checked whether the estimations can be generalized over time and different user.

Table 6.13 depicts an overview of the mean squared errors of the particular data sets and the algorithms. On the Test-Track-Set and the One-User-Set both algorithms perform similar and

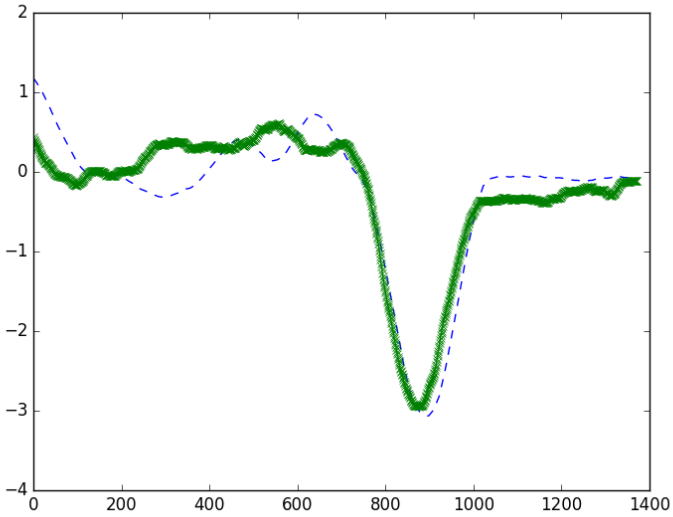


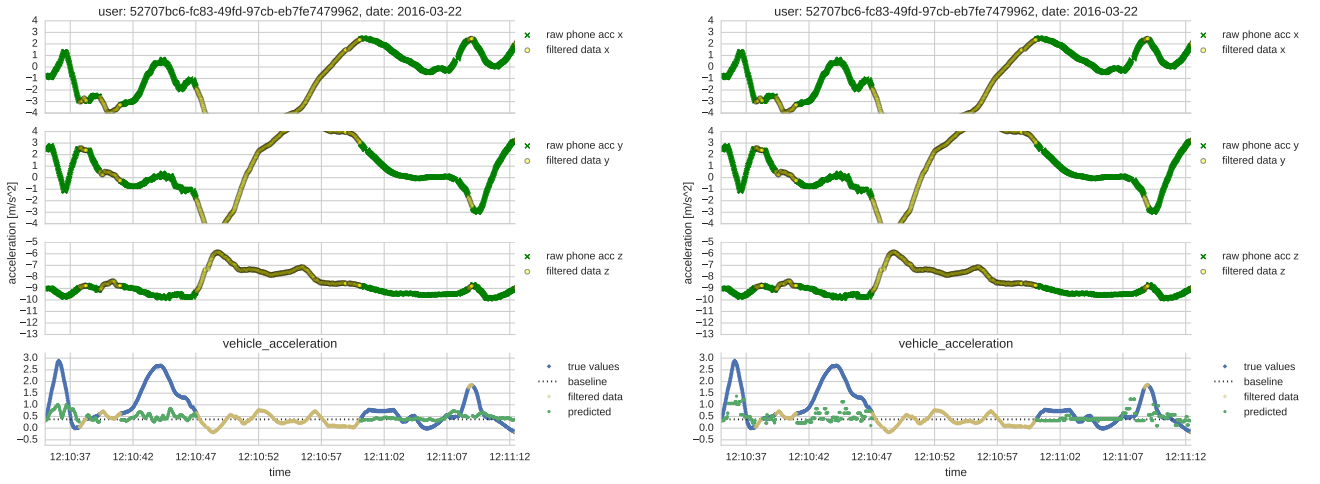
Figure 6.6: Example regression on small slice of data using linear regression. On the x-axis is the index of the data point and on the y axis is the vehicle acceleration (m/s^2). The dotted blue line is the actual vehicle acceleration and the green markers are the estimated values. This slice is trained and tested on the same data set. Note, this is no evidence for generalization of the method. But, it shows that there is a linear relationship between smartphone acceleration data and vehicle acceleration.

are slightly better than baseline. On the Many-User-Set the regression tree performs equal to baseline and the linear regression performs slightly better than baseline.

Data Set	Method	MSE
One-User-Set	Linear	0.175
	RTree	0.177
	Baseline	0.184
Many-User-Set	Linear	0.149
	RTree	0.152
	Baseline	0.152
Test-Track-Set	Linear	0.263
	RTree	0.267
	Baseline	0.288

Table 6.13: MSE on test set of vehicle acceleration detection experiments. The predictions are performed with Linear Regression (Linear), Decision Tree Regression (RTree), and as Baseline the mean output value of the train set is used.

Since the Test-Track-Set performs best, the results are exemplary shown on this set. Figure 6.7 depicts an example slice of smartphone acceleration values and corresponding vehicle acceleration. Estimations with linear model and with regression tree are shown. The plots in the figure also include filtered data, presented as yellow markers. The linear model produces



(a) Linear **(b)** Regression Tree

Figure 6.7: Example prediction on Test-Track-Set. The baseline in bottom plot is defined by the mean vehicle acceleration of the train set. The yellow dots show filtered smartphone acceleration data points. These points are filtered since the z-acceleration values differs more than 10% from gravity (9.81 m/s^2).

a continuous signal, whereas the regression trees estimations jump. Both estimators produce many estimations near baseline.

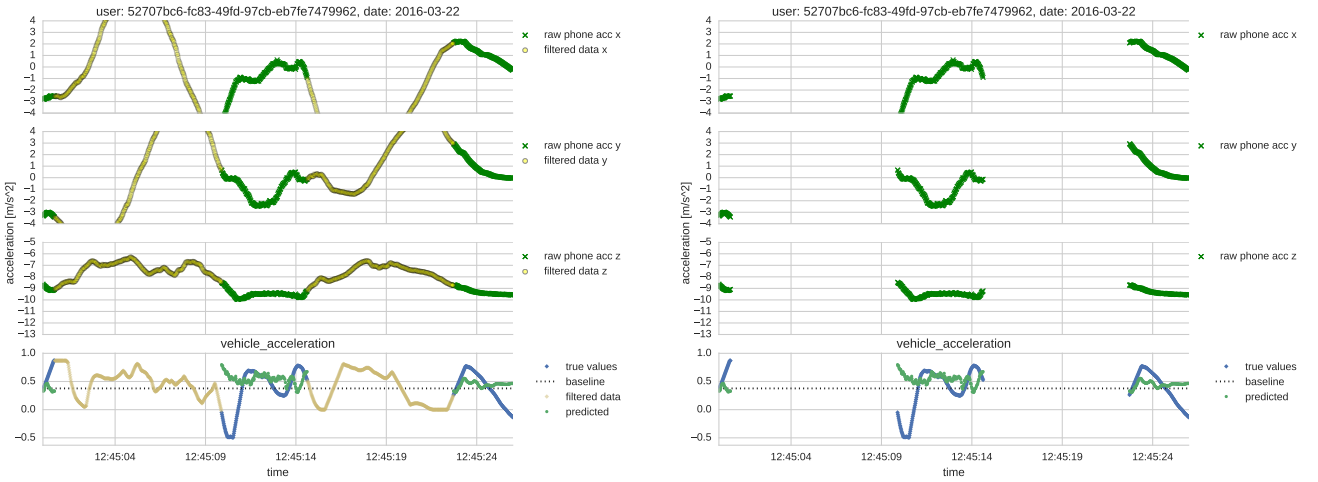
Filter data due to the z acceleration values filters about 30% of the data. The result is shown in Figure 6.8. Here, the acceleration data is shown with and without filtered data to illustrate the resulting signal used for regression.

Interpretation

The results show that on the present data it is not possible to infer the vehicles acceleration by use of the smartphone acceleration data accurately. All of the algorithms are only slightly better than the baseline, which only estimates the mean vehicle acceleration of the train set for every data point on the test set. On the Many-User-Set the regression tree performs equally to the baseline.

Figure 6.6 shows a linear correlation of smartphone acceleration to vehicle acceleration on a small time slice. In the Figure, it can be seen, that especially the extreme value is well reflected. But, as stated above, this is trained and tested on the same data set, so the algorithm can also just over-fit. It also shows, for small accelerations, the estimation performance is already relatively low in comparison to estimations around the minimum of the slice.

The results of Table 6.13 show that the regression models neither generalize over time nor over user. For the Test-Track-Set the regression algorithms work best. The Test-Track-Set contains data from one user and one day. I.e. most probably the smartphone was mounted once and kept in this position for the whole time of recording. Hence, possible artifacts from trans-



(a) With filtered data

(b) Without filtered data

Figure 6.8: Example prediction on Test-Track-Set with and without filtered data. The baseline in bottom plot is defined by the mean vehicle acceleration of the train set. (a) and (b) show the same slice of acceleration data. In both cases predictions are performed on same train and test set and with linear regression. (a) includes the filtered data (yellow dots). These points are filtered since the z-acceleration values differs more than 10% from gravity (9.81 m/s^2). (b) misses the filtered points to illustrate the resulting signal used for regression.

formation or smartphone itself are same for train and test set. Thus, regression performs best on Test-Track-Set. The more variance the data gets, in One-User-Set more days of one user and in Many-User-Set different user with different smartphones, the worse regression performs. For Many-User-Set it only performs equal to baseline.

Since estimations on Test-Track-Set perform best, some example estimations are shown in Figures 6.7 and 6.8. Figure 6.7 compares the estimations of linear regression and a regression tree. As the results of Table 6.13 indicate, both estimator's predictions are mainly located around baseline. Comparing the plot of the x acceleration value and the vehicle acceleration, a correlation of the shape can be observed. Nevertheless, the model is not capable to identify this graphically viewable correlation. Figure 6.8 illustrates the result of filtering data due to z acceleration value. The example slice shows, on one hand, that the z acceleration value is a good indicator for whether the transformation from smartphone coordinate system to car coordinate system did work. On the other hand, it makes clear how much data is filtered. The continuous signal becomes discontinuous. But, the example slice in Figure 6.8 also shows that at ranges, where data is filtered, the above mentioned *observable correlation* is vanished. Instead, there are some unreasonable acceleration values on x and y axis, which do not correspond to the shown vehicle acceleration.

In summary it can be said that with the present data, especially the present transformation values, it is not possible to perform qualitative inference of the car's actual acceleration.

6.3.4 Heavy Braking Events

A heavy braking event is defined in this work by a vehicle acceleration lower than -2 m/s^2. So instead of detecting specific events, that depend on many external factors or exact values, extreme values of the acceleration are detected. An attempt is done, to locate them by using only the smartphone acceleration data. The problem is handled as a binary classification task, where the inputs are the features calculated by the smartphone acceleration vectors and the output is the variable whether a heavy braking event occurred or not. The One-User-Set contains 587 heavy braking events, which results in 1390 out of 216,000 feature vectors labeled as a brake event. The Many-User-Set contains 3015 brake events and 1286 out of 245,000 feature vectors labeled as brake event. The Test-Track-Set contains 144 brake events and 411 out of 25,687 feature vectors labeled as brake event. Note, that it may happen that there are more events than windows, since the windows consists of smartphone acceleration data, which is filtered if there was no GPS signal and the events are computed by the raw vehicle acceleration signal.

Implementation

A data set containing the time stamps of the lowest acceleration during a brake is provided by the supervisor of this project. This data set was gathered by checking the point where the acceleration undershoots a value of -2 m/s^2 and the point when it overshoots a value of 0 m/s^2 again. The minimal value and its time stamp inside this period is defined as the moment of the heavy brake.

The duration of the brake event is set from 100 milliseconds before and 100 milliseconds after the point of the time stamp of the heavy brake. Since it is only intended to recognize the extreme values of the braking, all the windows containing data 5 seconds after and 5 seconds before the heavy brake event are deleted from the data set, if they do not contain a break event.

The acceleration data is split into windows of one second duration and 0.5 seconds overlap. If 10% of the window is inside the time of the brake event, the window is labeled as a brake event. All windows that have no interception with the brake event and are 5 seconds before or after the break event are deleted from the data set, as mentioned above.

The windows are filtered due to the mean_z value of the feature vectors, to cope with wrongly transformed acceleration vectors (Section 5.6). Table 6.14 illustrates the number of feature vectors in train and test set after filtering.

As classifiers a Random Forest, a Decision Tree, and a SVM are tested. The implementations of scikit-learn 18.1 (Pedregosa et al., 2011) are used for the experiments. The parameters are optimized by cross validation grid search on the train set.

Data Set	Train	Test
One-User-Set	145	180
Many-User-Set	212	74
Test-Track-Set	73	26

Table 6.14: Number of feature vectors labeled as brake event in the train and the test set for the particular data sets after filtering

As for ABS and ESC detection the classification problem is unbalanced. Hence, again the data sets are sub-sampled, such that for every 'on' example 40 'off' examples are randomly sampled from the data. All experiments are repeated 100 times with different random sub-samples.

Results

Tables 6.16 to 6.18 show the optimized parameters for the particular algorithms and data sets.

Data Set	Method	Accuracy	Precision	Recall	F1-Score	TP (P)	FP (N)
One-User-Set	RF	0.95	0.31	0.75	0.43	134.6 (180)	305.6 (7200)
	DT	0.95	0.25	0.57	0.35	103.0 (180)	307.9 (7200)
	SVM	0.93	0.21	0.74	0.33	132.6 (180)	501.2 (7200)
Many-User-Set	RF	0.85	0.13	0.91	0.24	67.5 (74)	435.0 (2960)
	DT	0.87	0.09	0.49	0.15	36.2 (74)	363.4 (2960)
	SVM	0.79	0.09	0.82	0.16	60.8 (74)	633.0 (2960)
Test-Track-Set	RF	0.96	0.25	0.32	0.28	8.4 (26)	24.9 (1040)
	DT	0.96	0.22	0.18	0.19	4.6 (26)	18.0 (1040)
	SVM	0.96	0.21	0.25	0.22	6.4 (26)	25.2 (1040)
Test-Track-Set unfiltered	RF	0.96	0.33	0.60	0.43	52.6 (87)	106.2 (3480)
	DT	0.95	0.24	0.34	0.28	30.1 (87)	97.4 (3480)
	SVM	0.96	0.27	0.44	0.33	38.4 (87)	103.9 (3480)

Table 6.15: Metric values on test set of brake event detection experiments. The values are the mean values of experiments with 100 random sub-samples of the train and the test data set. where for every 'on' example 40 'off' examples are randomly sampled from the particular data set. In the last two columns, TP denote the number of true positives, i.e. correctly as 'on' classified feature vectors, P the number of all 'on' labeled feature vectors in the test set. FP (False Positive) denotes the number of incorrectly 'on' labeled feature vectors and N the number of the 'off' labeled feature vectors in the test set.

Table 6.15 shows the metric scores of the particular data sets and the algorithms used for inference. The regression tree always performs best. Decision tree and SVM perform similar, but always worse than the random forest. One-User-Set and the unfiltered Test-Track-Set perform best. The Many-User-Set performs worst.

Dataset	max_depth	max_features	max_leaf_nodes	min_impurity_split	min_samples_leaf	min_samples_split	min_weight_fraction_leaf	n_estimators	class_weight
One-User-Set	None	16	None	10^{-7}	16	2	0	100	balanced
Many-User-Set	4	4	None	10^{-7}	1	8	0	100	balanced
Test-Track-Set	None	16	None	10^{-7}	16	2	0	10	balanced

Table 6.16: Optimized parameters - Brake Event detection - Random Forest

Dataset	max_depth	max_features	max_leaf_nodes	min_impurity_split	min_samples_leaf	min_samples_split	min_weight_fraction_leaf	presort	class_weight
One-User-Set	11	auto	None	10^{-7}	1	2	0	False	balanced
Many-User-Set	9	None	None	10^{-7}	1	2	0	False	balanced
Test-Track-Set	None	16	None	10^{-7}	16	2	0	False	balanced

Table 6.17: Optimized parameters - Brake Event detection - Decision Tree

Dataset	C	kernel	decision_function_shape	class_weight
One-User-Set	1.9	rbf	ovr	balanced
Many-User-Set	0.3	rbf	ovr	balanced
Test-Track-Set	1.8	rbf	ovr	balanced

Table 6.18: Optimized parameters - Brake Event detection - SVM

Figure 6.9 shows the distribution of the feature values. For every feature the mean and standard deviation value is shown for the 'off' and 'on' labeled feature vectors. Whereas, 'off' means *no heavy braking detected* and 'on' means *heavy breaking detected*. It shows that on Test-Track-Set the mean values are best distinguishable from 'off' to 'on'. On the unfiltered Test-Track-Set it is even better distinguishable. On the One-User-Set there are many overlapping values. The Many-User-Set has many distinguishable mean values with overlapping standard deviations.

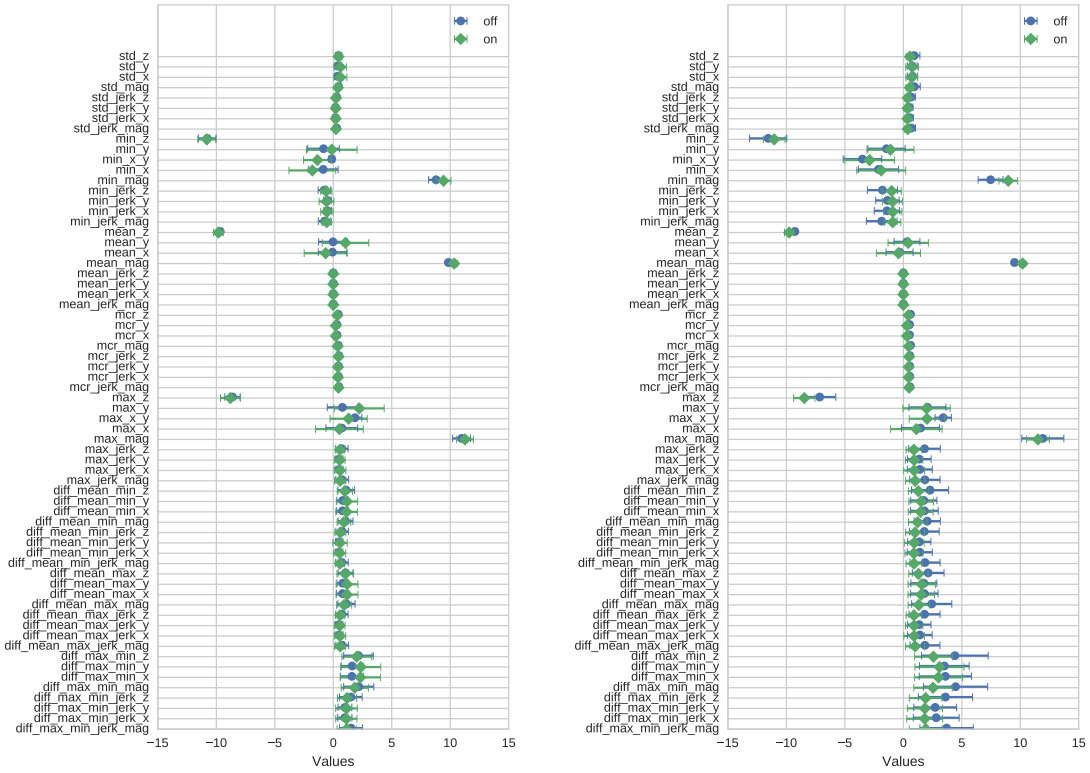
To detect brake events a data set is provided that specifies the moment in time of the highest deceleration (lowest acceleration). Figure 6.10 shows one example brake event label. It shows that the brake event label is not matching with the true lowest acceleration value in the displayed range.

Interpretation

It was assumed that heavy braking would be the best performing classification problem, since it was expected that, in general, the actual vehicle acceleration can be recognized on the smartphone acceleration. The heavy braking events only involve extreme (negative) acceleration values.

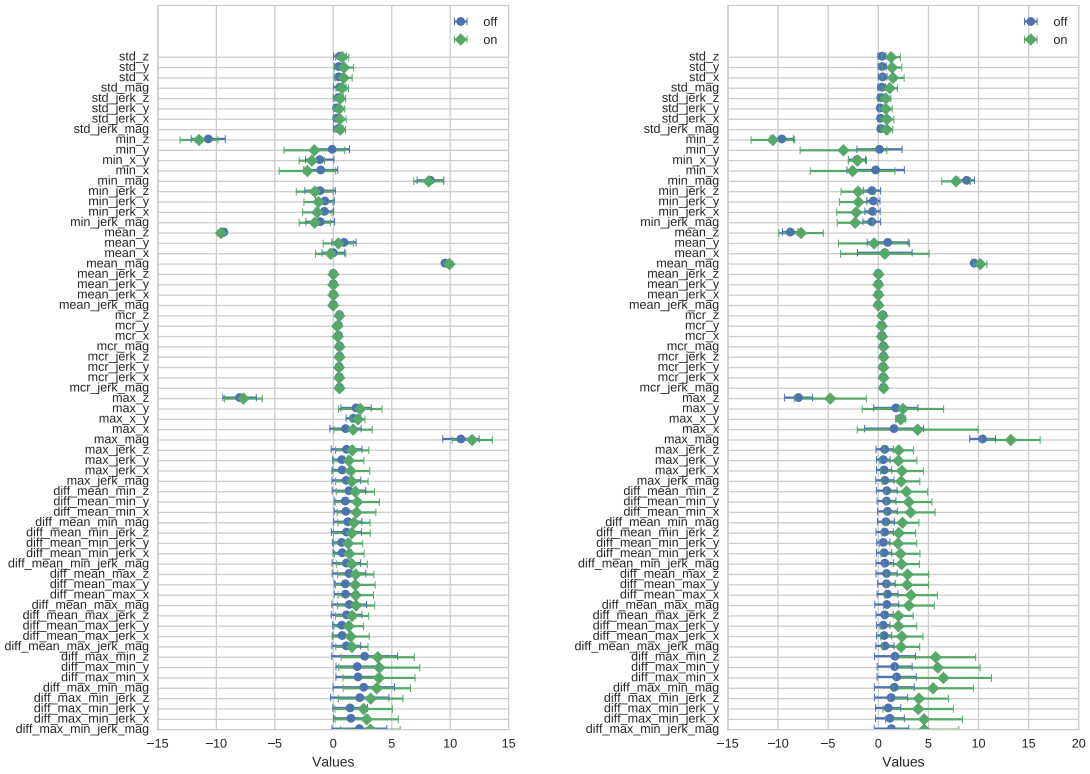
The results show that those negative acceleration values can not be detected as expected. Even with the Test-Track-Set, which in the other classification problems performed best, the results are not satisfying.

The problem is exemplary shown on the Test-Track-Set. Figure 6.9(c) and (d) show the feature value distribution of the Test-Track-Set, once with filtered data (c) and without (d). The data is filtered since the rotation of the smartphone acceleration vectors is not always performed correctly. As an indicator of the quality of the rotation, the mean value in z-direction of the window is used. If the *mean_z* value of the window deviates more than 10% of the gravity (9.81 m/s^2) the window is filtered, since the car is not expected to fly or fall. If the data is not filtered the mean value of the *mean_z* feature is shifted to the right. Without filtering it may happen that the vector is rotated wrong and features, that depend on a direction, are misinterpreted. The Test-Track-Set is a special case. During a drive on a test track the driver is driving more extreme than during a normal drive. Figure 6.3a on page 47 depicts another example for the unfiltered data performing better than filtered. The figure shows a brake event. During the brake the mounted smartphone strongly vibrates. The high variance of the data is the most distinct feature in this case. But, during the brake, the mean value of the z acceleration deviates from -9.81 m/s^2 to a higher value. Hence, this example is filtered in the implementation. Such cases can not be differentiated to cases where the transformation vector was wrong. The results in 6.15 point out that most of the filtered feature vectors are distinguishable by the classifier in this data set. However,



(a) One-User-Set

(b) Many-User-Set



(c) Test-Track-Set

(d) Test-Track-Set unfiltered

Figure 6.9: Comparison of feature value distributions of brake event off and on labeled feature vectors on the data sets. (d) shows the feature distribution of unfiltered Test-Track-Set. Here, the windows are not filtered on the basis of the mean_z value as in (c) In all plots the features are on the y axis and their values on the x axis. The markers show the mean value and the bars the standard deviation.

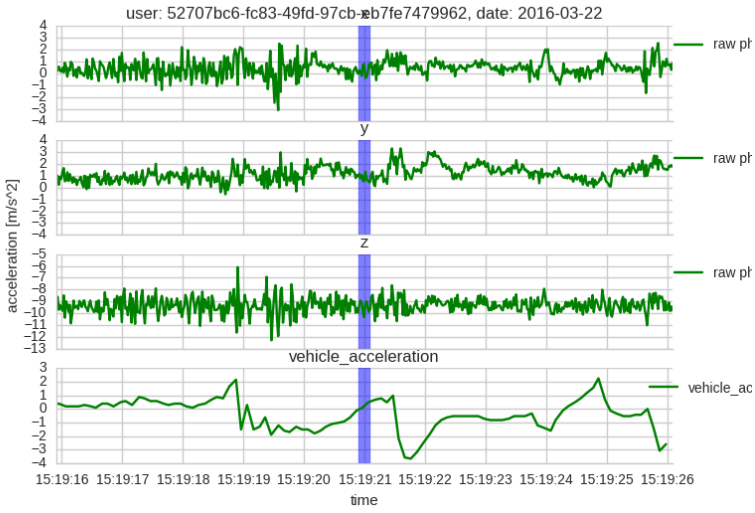


Figure 6.10: Wrong labeled brake event. Blue shaded area is the brake event. Lines are the acceleration values. Marker should be at the lowest point of acceleration (Approximately 15:19:21.8) but data is only with second precision, not millisecond

Figure 6.10 depicts a problem with the provided brake event data set. It shows that the label, as it is defined in the provided data set, does not match the minimum value of the vehicle's acceleration. The shift is resulted by the precision of the labels. Brake labels are provided with a secondly precision, whereas the acceleration data is covered in milliseconds. In the shown example, the brake event occurs at approximately millisecond 800 of the second. Hence, the label is shifted. The shift gets larger the farer away the true minimum of the brake event is from the last full second. Hence, some labels are shifted more than others.

Due to time restrictions of this project and the overall moderate performance of the other prediction experiments the mentioned problem of label precision is not further investigated. The prediction results match with the results seen during ABS and ESP detection. Hence, the results are not expected to improve much with a fixed labeling.

6.4 Discussion

It was tried to infer ABS and ESP activations, heavy braking events and the actual car acceleration. Other than in literature, the data was gathered using a free positioned smartphone during an uncontrolled field study. The results show, in general it is possible to estimate the car's acceleration by the smartphone. But it also shows that the gained model is not generalizable over users or over time of one user. The estimations only works on a sub-sampled set of the data, due to the highly unbalanced occurrence of 'on' and 'off' labels for classification problems. Sub-sampling the data improves the estimation performance. Without sub-sampling all learned classifiers predict the dominant class.

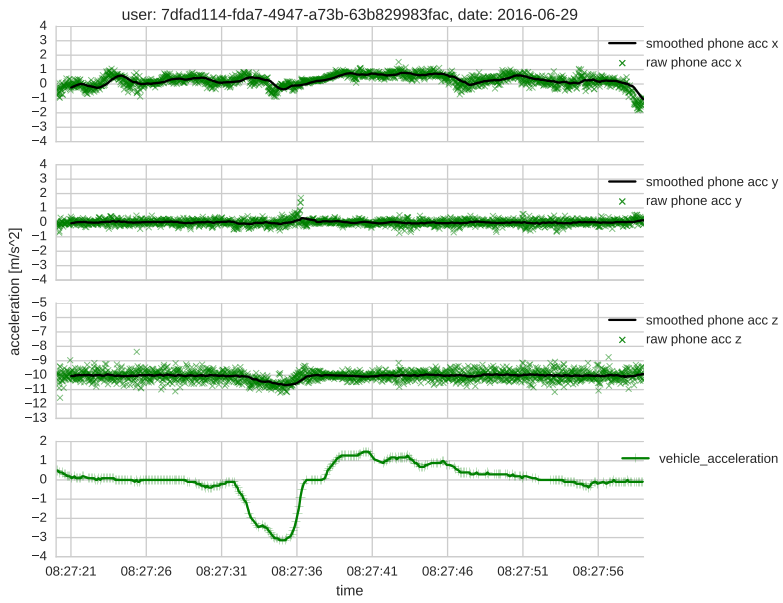


Figure 6.11: Example slice of smartphone data and vehicle acceleration to depict exemplary the difficulty for estimations arising from the smartphone acceleration data. On the x-axis is the time and on the y axis the acceleration values [m/s^2]. The upper three plots show the x, y, and z acceleration of the smartphone. Here, the green markers are the raw acceleration values and the black lines the smoothed values, smoothed with a sliding average of size 50 (one second). The forth plot shows the vehicle acceleration.

The most sensitive pre-processing step is the transformation of smartphone acceleration data from smartphone coordinate system to car coordinate system. First, acceleration data has to be transformed into world coordinate system and afterwards rotated to movement direction of the car. If the smartphone is fixed in the car and its position is known, these transformations can be calculated once analytically and then be applied to the data. If it is free positioned, one has to rely on the other sensors of the smartphone, which are the magnetometer and the GPS sensor. Without transformation, there would be no indication of the direction of the current acceleration. For example, data gathered from a phone mounted in landscape orientation provides no information for the data of a smartphone gathered in portrait orientation. Especially the information of the direction of the movement is lost. It would be not possible to differentiate between a brake and a speed up situation.

To overcome the problem of incorrect transformed data, the z-axis of the rotated smartphone acceleration vector is considered. It is assumed that, after successful transformation of data into world coordinate system, gravity is only applied in z-axis direction. Hence, it is assumed that if the acceleration on z-axis differs from real gravity, which is 9.81 m/s^2, transformation was not successful. Windows that's mean_z value differs more than 10% from real gravity are filtered from train and test set.

The transformation fails, especially in cases of fast movements and fast direction changes. Hence, take a turn or brake influences the transformation results. ABS activations mostly occur during brake, ESP most probably occur during a turn. The attempted estimations all take place whilst situations that influence the transformations. Hence, moments with the most distinguishable data get filtered out. These moments are not discriminable from situations the sensors were disturbed. The consequences of filtering can be inferred from Tables 6.2, 6.7 and 6.14, in which the number of feature vectors in train and test set after the filtering is shown. Compared to the number of feature vectors mentioned in the particular sections, the ratio of filtered feature vectors averages to over 50%. To overcome this problem data preparation has to be further investigated. For example, it could be tried to optimize transformation vectors. Instead of applying the transformation vectors provided by the smartphone, one could minimize the distance of z acceleration to gravity and thus transform the acceleration. Optimize every transformation of every acceleration vector requires a lot of computation time. Apart from that transformation quality could be improved by improving data quality while recording. Either the smartphone can be fixed, as it is done in most of related work. Or the smartphone has to be calibrated as suggested by (Almazán et al., 2013).

Even if the transformation of the smartphone acceleration vectors works, there are problems. Figure 6.11 depicts an example slice of rotated smartphone acceleration values and corresponding vehicle acceleration. The z-acceleration value around -10 indicates a working transformation of smartphone acceleration data to car coordinate system. Neither in raw data nor in smoothed smartphone acceleration data, the deceleration of the car is recognizable. The Figure shows a heavy brake event, where the vehicle acceleration is below -2 m/s^2. There is no indication for the brake event in the smartphone data. Thus, this example slice shows an inappropriate sample for brake event detection and for vehicle acceleration estimation.

It can be summarized that with the present data it is not possible to infer car system activations by use of a free positioned smartphone. The quality of the recorded transformation vectors is not adequate enough to infer event occurrences in the car.

7 Conclusion

The goal of this thesis was to investigate, whether low level events in a driving context can be inferred by using smartphone acceleration data.

Modern vehicles produce a huge amount of data. This data remains unused unless for internal diagnostics. To increase traffic safety and to get information of traffic or street condition, the data has to be shared in a network.

Volkswagen plans to start to equip new cars with a new standard for inter-vehicle communication in 2019 (Volkswagen, 2017). Until then, retrieving and sharing car data is expensive, since the internal communication is restricted and not standardized. Instead, smartphones can be used as a proxy from the real world to the network. Hence, instead of recording the car data explicitly, the sensor values can be estimated by a smartphone and sent to the network.

In the literature, smartphone acceleration data is used for several inference tasks, such as road property estimation (Fazeen et al., 2012) or driving behavior estimation (Sharma et al., 2015). In this thesis, low-level event detection is investigated. Anti-lock Braking System (ABS) and Electronic Stability Control (ESC) activations, the vehicle's current acceleration and heavy brake events are detected.

The data used for the inference was gathered in forefront of this thesis by researchers of ETH Zürich and university of St. Gallen (Ryder et al., 2016). Data of a car was recorded with smartphone acceleration data simultaneously. The data of the car is used to label the smartphone acceleration vectors.

To process the data, a pipeline is developed and implemented. First, the data is joined, since originally every sensor value is stored in a separate database table. Accelerometer vectors are transformed, such that they reference to the car's coordinate system, to be comparable to each other. The transformed data is windowed and features are extracted. The feature vectors are labeled and wrongly transformed data, as data where the vehicle did not move, is filtered from data set.

The derived data is used for inference. To test the models on generalization over time, generalization over user and on extreme situations, three data sets are sampled from overall data. One contains data of one user over several days, one contains data of 5 users and one or two days each, and one contains data recorded during a drive on a test track.

All inference attempts perform best on the test track data. Hence, ABS, ESC, brake and vehicle acceleration detection work best in extreme situations, since the activation data is better distinguishable than during normal drive. The results give evidence that the model generalizes better over time of one user than over different user. ABS and ESC detection perform similar.

Whereby, on the obtained sub data set, ESC detection does not generalize over users. Vehicle acceleration estimation performs slightly better than the baseline for all data sets. The baseline is defined as the mean vehicle acceleration of the train set. Heavy brake detection performed worse than ESP and ABS detection. But, there is a problem with the provided label set, which declares the brake events with second accuracy but rest of the data is given in millisecond accuracy. Hence, it occurs that the label and the actual brake event do not match. Due to time restrictions of this project, the brake event label problem is not further investigated. Considering the other experiment's results, especially the vehicle acceleration estimation, brake detection performance is not expected to improve much with fixed labeling.

To summarize, precise estimation of a vehicle's safety systems or accurate recognition of the vehicle's acceleration is not possible. On the one hand, the data is not accurate enough. Several problems occur using acceleration data of a free positioned smartphone. Transformation vectors are available in low frequency and transformation accuracy is disturbed by other electric devices in the car. The angle of direction of travel is based on GPS which is inaccurate and available in low frequency. In literature, experiments with fixed smartphones are predominate. On the other hand, safety systems, such as ABS and ESC, are activated during situations that are not recognizable in acceleration.

7.1 Future Work

To improve vehicle acceleration estimation and to further investigate vehicle system activation estimation, further research on smartphone acceleration vector transformation has to be performed. One big problem during this thesis were inaccurate transformations of smartphone acceleration vectors to car coordinate system. To be able to work with free positioned smartphones, these transformations have to be more accurate. First, more data could be recorded. In addition to the free positioned smartphone, a fixated smartphone could be installed in the car. This fixated smartphone should be positioned in a known location with a known orientation. With this data, transformation vectors can be evaluated and the quality of sensor readings can be estimated.

The present data set could be used to optimize transformation vectors. To prove whether transformation performed as expected, z-acceleration was taken into account. Assuming that there is no or only little acceleration in z-direction, it could be tried to optimize transformation such that the error of transformed acceleration to gravity minimizes. Moreover, instead of using the bearing angle, which is inaccurate, distances to vehicles longitudinal and lateral acceleration could be minimized.

Even though optimization approaches may be interesting for laboratory research, it is not practical for real live applications. Hence, more accurate smartphone sensors are required.

For the experiments in this thesis, features utilized in related literature are used to represent data windows. For further research, more features could be generated and tested. Or, instead of feature engineering, methods with automated feature generation could be applied. One method providing automated feature generation are Neural Networks. Neural Networks are applied to a variety of problems including signal processing (Cochocki and Unbehauen, 1993). Instead of defining features, important patterns are detected automatically. A drawback of Neural Network is that the resulting model can not be interpreted.

If overall quality of the transformation increases, more estimations in driving context could be performed by using smartphone acceleration data. When it is possible to detect vehicle acceleration accurately, more detailed inference could be conducted, such as estimating the vehicle's longitudinal or lateral acceleration. Also, the vehicle acceleration estimation can be used to improve vehicle speed estimation of GPS (Han et al., 2014). Other low level estimations could be throttle position or brake pedal position. It may also be tried to estimate the current wheel slip status. If GPS data is matched to a map and more environment information is added, road estimations could be performed, such as current slope or curvature.

The more sensor values are identifiable with smartphone data, the more independence from automobile manufacturers can be gained regarding information retrieval in a driving context.

Acronyms

ABS Anti-lock Braking System

AR Augmented Reality

CAN Controller Area Network

DTW Dynamic Time Warping

ESC Electronic Stability Control

GPS Global Positioning System

HMM Hidden Markov Model

IMU Inertial Measurement Unit

k-NN k-Nearest Neighbors

LDA Latent Dirichlet Analysis

OBD On-Board Diagnostics

PCA Principal Component Analysis

SMA Simple Moving Average

SVM Support Vector Machine

Bibliography

Abbott, E. and Powell, D. (1999). Land-vehicle navigation using GPS. *Proceedings of the IEEE*, 87(1):145–162.

Almazán, J., Bergasa, L. M., Yebes, J. J., Barea, R., and Arroyo, R. (2013). Full auto-calibration of a smartphone on board a vehicle using IMU and GPS embedded sensors. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1374–1380.

Alvarez, A. D., Garcia, F. S., Naranjo, J. E., Anaya, J. J., and Jimenez, F. (Fall 2014). Modeling the Driving Behavior of Electric Vehicles Using Smartphones and Neural Networks. *IEEE Intelligent Transportation Systems Magazine*, 6(3):44–53.

Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79.

Bhoraskar, R., Vankadhara, N., Raman, B., and Kulkarni, P. (2012). Wolverine: Traffic and Road Condition Estimation using Smartphone Sensors. In *Fourth International Conference on Communication Systems and Networks*, pages 1–6.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bitkom (2017). Anteil der Smartphone-Nutzer in Deutschland in den Jahren 2013 bis 2017. `https://de.statista.com/statistik/daten/studie/585883/umfrage/anteil-der-smartphone-nutzer-in-deutschland/`. Accessed: 27, June 2017.

Bosch, R. (1991). CAN Specification version 2.0. *Robert Bosch GmbH, Postfach 30 02 40, D-70442 Stuttgart*.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

Botzer, A., Musicant, O., and Perry, A. (2017). Driver behavior with a smartphone collision warning application – A field study. *Safety Science*, 91:361–372.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Cervantes-Villanueva, J., Carrillo-Zapata, D., Terroso-Saenz, F., Valdes-Vela, M., and Skarmeta, A. F. (2016). Vehicle Maneuver Detection with Accelerometer-Based Classification. *Sensors*, 16(10):1618.

Chu, H., Raman, V., Shen, J., Kansal, A., Bahl, V., and Choudhury, R. R. (2014). I am a Smartphone and I Know My User is Driving. In *Sixth International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–8.

Cochocki, A. and Unbehauen, R. (1993). *Neural Networks for Optimization and Signal Processing*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition.

Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3):273–297.

Daptardar, S., Lakshminarayanan, V., Reddy, S., Nair, S., Sahoo, S., and Sinha, P. (2015). Hidden Markov Model based Driving Event Detection and Driver Profiling from Mobile Inertial Sensor Data. In *2015 IEEE SENSORS*, pages 1–4.

Das, S., Green, L., Perez, B., Murphy, M., and Perring, A. (2010). *Detecting User Activities Using the Accelerometer on Android Smartphones*. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA.

Dreyfus, H. L. and Dreyfus, S. E. (1992). What Artificial Experts Can and Cannot Do. *AI & Soc*, 6(1):18–26.

Enev, M., Takakuwa, A., Koscher, K., and Kohno, T. (2016). Automobile Driver Fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016(1):34–50.

England, B. (2014). What is the Anti-Lock Brake System and How do I Maintain it? `https://www.britishamericanauto.com/car-repair-education-and-info-blog/bid/90514/What-is-the-Anti-Lock-Brake-System-and-How-do-I-Maintain-it`. Accessed: 27, June 2017.

Eren, H., Makinist, S., Akin, E., and Yilmaz, A. (2012). Estimating Driving Behavior by a Smartphone. In *2012 IEEE Intelligent Vehicles Symposium*, pages 234–239.

Farsi, M., Ratcliff, K., and Barbosa, M. (1999). An overview of Controller Area Network. *Computing Control Engineering Journal*, 10(3):113–120.

Fazeen, M., Gozick, B., Dantu, R., Bhukhiya, M., and González, M. C. (2012). Safe Driving Using Mobile Phones. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1462–1468.

Geurts, P. (2002). *Contributions to Decision Tree Induction: Bias/Variance Tradeoff and Time Series Classification*. PhD thesis, University of Liège Belgium.

Google (2017). Motion Sensors. `https://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-rotate`. Accessed: 02, March 2017.

Han, H., Yu, J., Zhu, H., Chen, Y., Yang, J., Zhu, Y., Xue, G., and Li, M. (2014). SenSpeed: Sensing Driving Conditions to Estimate Vehicle Speed in Urban Environments. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 727–735.

Hawkins, D. M. (2004). The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12.

Heideman, M., Johnson, D., and Burrus, C. (1984). Gauss and the History of the Fast Fourier Transform. *IEEE ASSP Magazine*, 1(4):14–21.

Johnson, D. A. and Trivedi, M. M. (2011). Driving Style Recognition Using a Smartphone as a Sensor Platform. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1609–1615.

Kaiser, J. and Reed, W. A. (1977). Data smoothing using low-pass digital filters. *Review of Scientific Instruments*, 48(11):1447–1457.

Khan, A. M., Lee, Y. K., Lee, S. Y., and Kim, T. S. (2010). Human Activity Recognition via an Accelerometer-Enabled-Smartphone Using Kernel Discriminant Analysis. In *2010 5th International Conference on Future Information Technology*, pages 1–6.

Lee, Y.-S. and Cho, S.-B. (2011). Activity Recognition Using Hierarchical Hidden Markov Models on a Smartphone with 3D Accelerometer. In *Hybrid Artificial Intelligent Systems*, pages 460–467. Springer, Berlin, Heidelberg.

Lie, A., Tingvall, C., Krafft, M., and Kullgren, A. (2006). The Effectiveness of Electronic Stability Control (ESC) in Reducing Real Life Crashes and Injuries. *Traffic Injury Prevention*, 7(1):38–43.

Liebemann, E. K., Meder, K., Schuh, J., and Nenninger, G. (2004). Safety and Performance Enhancement: The Bosch Electronic Stability Control (ESP). In *SAE Convergence*, pages 1–9.

Lu, H., Yang, J., Liu, Z., Lane, N. D., Choudhury, T., and Campbell, A. T. (2010). The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 71–84, New York, NY, USA. ACM.

Malekipirbazari, M. and Aksakalli, V. (2015). Risk Assessment in Social Lending via Random Forests. *Expert Systems with Applications*, 42(10):4621–4631.

Mangan, S., Wang, J., and Wu, Q. (2003). Longitudinal Road Gradient Estimation Using Vehicle CAN Bus Data. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2336–2341.

Mitrovic, D. (2005). Reliable Method for Driving Events Recognition. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):198–205.

Mukherjee, A. and Majhi, S. (2016). Characterisation of road bumps using smartphones. *European Transport Research Review*, 8(2):13:1–13:12.

Murphey, Y. L., Milton, R., and Kiliaris, L. (2009). Driver's Style Classification Using Jerk Analysis. In *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, pages 23–28.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT press.

Nice, K. (2000). How Anti-Lock Brakes Work. `http://auto.howstuffworks.com/auto-parts/brakes/brake-types/anti-lock-brake1.htm`. Accessed: 27, June 2017.

Paefgen, J., Kehr, F., Zhai, Y., and Michahelles, F. (2012). Driving Behavior Analysis with Smartphones: Insights from a Controlled Field Study. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, MUM '12, pages 36:1–36:8, New York, NY, USA. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ruta, M., Scioscia, F., Gramegna, F., and Di Sciascio, E. (2010). A Mobile Knowledge-based System for On-Board Diagnostics and Car Driving Assistance. In *UBICOMM 2010, The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 91–96.

Ryder, B., Gahr, B., and Dahlinger, A. (2016). An In-Vehicle Information System Providing Accident Hotspot Warnings. In *Twenty-Fourth European Conference on Information Systems*. AIS.

Safavian, S. R. and Landgrebe, D. (1991). A Survey of Decision Wee Classifier Methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674.

Sathyanarayana, A., Sadjadi, S. O., and Hansen, J. H. L. (2012). Leveraging Sensor Information from Portable Devices towards Automatic Driving Maneuver Recognition. In *15th International IEEE Conference on Intelligent Transportation Systems*, pages 660–665.

Sferco, R., Page, Y., Le Coz, J.-Y., and Fay, P. A. (2001). Potential Effectiveness of Electronic Stability Programs (ESP). *Proc. 17th Int. Enhanced Safety Vechicles Conf.*

Shannon, C. E. (1949). Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21.

Sharma, H., Naik, S., Jain, A., Raman, R. K., Reddy, R. K., and Shet, R. B. (2015). S-Road Assist: Road surface conditions and Driving Behavior analysis using Smartphones. In *International Conference on Connected Vehicles and Expo*, pages 291–296.

Shih, W. Y., Chen, L. Y., and Lan, K. C. (2012). Estimating Walking Distance with a Smart Phone. In *Fifth International Symposium on Parallel Architectures, Algorithms and Programming*, pages 166–171.

Shoemake, K. (1985). Animating Rotation with Quaternion Curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 245–254, New York, NY, USA. ACM.

Siegel, J. E., Bhattacharyya, R., Sarma, S., and Deshpande, A. (2015). Smartphone-Based Wheel Imbalance Detection. In *ASME 2015 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers.

Thompson, C., White, J., Dougherty, B., Albright, A., and Schmidt, D. C. (2010). Using Smartphones to Detect Car Accidents and Provide Situational Awareness to Emergency Responders. In *Mobile Wireless Middleware, Operating Systems, and Applications*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 29–42. Springer Berlin Heidelberg.

Vaitkus, V., Lengvenis, P., and Žylius, G. (2014). Driving Style Classification using Long-Term Accelerometer Information. In *19th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 641–644.

Veloso, F. and Fixson, S. (2001). Make–Buy Decisions in the Auto Industry. *Technological Forecasting and Social Change*, 67(2):239–257.

Vo, Q. V., Lee, G., and Choi, D. (2012). Fall Detection Based on Movement and Smart Phone Technology. In *IEEE RIVF International Conference on Computing Communication Technologies, Research, Innovation, and Vision for the Future*, pages 1–4.

Volkswagen (2017). With the aim of increasing safety in road traffic, Volkswagen will enable vehicles to communicate with each other as from 2019. `https://www.volkswagen-media-services.com/en/detailpage/-/detail/With-the-aim-of-increasing-safety-in-road-traffic-Volkswagen-will-enable-vehicles-to-view/5234247/7a5bbec13158edd433c6630f5ac445da`. Accessed: 29, June 2017.

Wang, Z. and Bovik, A. C. (2009). Mean Squared Error: Love It or Leave It? A new look at Signal Fidelity Measures. *IEEE Signal Processing Magazine*, 26(1):98–117.

White, J., Thompson, C., Turner, H., Dougherty, B., and Schmidt, D. C. (2011). WreckWatch: Automatic Traffic Accident Detection and Notification with Smartphones. *Mobile Networks and Applications*, 16(3):285.

Wing, M. G., Eklund, A., and Kellogg, L. D. (2005). Consumer-Grade Global Positioning System (GPS) Accuracy and Reliability. *Journal of Forestry*, 103(4):169–173.

Wu, M., Zhang, S., and Dong, Y. (2016). A Novel Model-Based Driving Behavior Recognition System Using Motion Sensors. *Sensors*, 16(10):1746.

Zaldivar, J., Calafate, C. T., Cano, J. C., and Manzoni, P. (2011). Providing Accident Detection in Vehicular Networks Through OBD-II Devices and Android-based Smartphones. In *2011 IEEE 36th Conference on Local Computer Networks (LCN)*, pages 813–819.

Zheng, J., Cao, J., He, Z., and Liu, X. (2014). iTrip: Traffic Signal Prediction using Smartphone based Community Sensing. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2944–2949.

Zheng, Y. and Hansen, J. H. L. (2016). Unsupervised Driving Performance Assessment using Free- Positioned Smartphones in Vehicles. In *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1598–1603.