Learning controller for prediction of lane change times:
A study of driving behaviour using naive Bayes and Artificial Neural Networks

Ermin Kodzaga
Abstract

Learning controller for prediction of lane change times

Ermin Kodzaga

Today’s trucks are becoming more and more safe due to the use of an Advanced Driver Assistance System (ADAS). This system is aimed to assist the driver in the driving process, and to increase the safety for both the driver and the environment around the vehicle. These systems require strict design criteria to enable sufficiently high precision and robustness. ADAS are developing intensely today, and these systems represent a way towards a completely autonomous vehicle community.

The main focus of this master thesis project is to investigate the possibility of predicting a driver’s typical lane change time before the truck reaches a highway. This was done by trying to identify the driving behaviour using sensor data from non-highway driving. Techniques from machine learning, such as naive Bayes and Artificial Neural Networks (ANN), with various combinations of sensor inputs were used during this process.

The results indicate that the assumption that different driving behaviours are representing different lane change times is true. Furthermore, predicting lane change times in whole seconds was as difficult as predicting lane change of three classes, fast, medium and slow. Predicting fast or slow lane change gave a better result. Only one set of validation data of totally five was predicted incorrectly. There was no big difference in the results between naive Bayes and the designed ANN. However, the results were not good enough for practical use, and more research is needed. Methods for increasing the performance and future work are also discussed.
Acknowledgements

First and foremost, I would like to thank Scania, and especially my supervisors Joseph Ah-King and Christian Larsson for giving me the opportunity to do my master thesis project and for all the support and guidance during the project. Also, many thanks to others colleagues at Scania that have helped advised me, and also my thesis colleagues at the department, Klas Lindsten, Gustav Ling and Fredrik Ljungberg. Last but not least, a special thank to my subject reader Marcus Björk and examiner Mikael Bergkvist.
# Contents

Acknowledgements iii

Abbreviations vi

1 Introduction 1
   1.1 Background 1
   1.2 The Problem 2
   1.3 Limitations 2

2 Theory 3
   2.1 Supervised versus Unsupervised Learning 3
   2.2 Classification methods 5
      2.2.1 Naive Bayes 6
      2.2.2 Artificial Neural Network 7
      2.2.3 Training neural networks 9
      2.2.4 Stopping criterion for neural networks 9

3 Implementation 10
   3.1 Acquiring sensor data 10
      3.1.1 Sensor data 12
   3.2 Preprocessing sensor data 13
      3.2.1 Filtering sensor data 14
      3.2.2 Defining lane change 14
      3.2.3 Defining lane change time 16
      3.2.4 Choosing lane change time interval 16
      3.2.5 Define a lane change class 17
   3.3 Window Design 17
      3.3.1 Window with one sample in features 18
      3.3.2 Window with features representing time series 18
   3.4 Classifier design 20
      3.4.1 Feature analysis 20

4 Results 22
   4.1 Classification - features with one sample 22
      4.1.1 Naive Bayes method 22
   4.2 Classification - features representing time series 23
4.2.1 Naive Bayes method ........................................ 23
4.2.2 ANN method .............................................. 25

5 Discussion ....................................................... 27
  5.1 Classification results ...................................... 27
  5.2 Improving the classification results ..................... 28
  5.3 Conclusions ................................................ 29

Bibliography ................................................... 30
Abbreviations

ADAS  Advanced Driver Assistance System
ANN   Artificial Neural Network
CAN   Controller Area Network
ECU   Electronic Control Unit
LCA   Lane Change Assist
LKA   Lane Keep Assist
Chapter 1

Introduction

Autonomous vehicles are developing intensively today. The way towards a completely autonomous vehicle community is an Advanced Driver Assistance System (ADAS). ADAS is a system that is aimed to assist the driver in the driving process and to increase the safety for both the driver, and the environment around the vehicle. These systems require strict criteria to ensure sufficiently high precision and robustness. Some ADAS systems are required by law today, like Advanced Emergency Brake. The law is applied in Europe on newly produced buses and trucks with two and three axles. Other systems that increase the driver’s comfort and safety are Lane Keep Assist (LKA), Lane Departure Warning, Adaptive Cruise Control and Electronic Stability Program. ADAS systems have also been proven to reduce fuel consumption, which plays an important part in profitability for many companies.

1.1 Background

Run off road crashes due to tiredness, and distraction, or crashes due to changing lanes, are becoming more common today. An LKA system warns and helps the driver to avoid such accidents, and also assists in steering the vehicle. The system uses sensor technology in order to identify the lanes on the road. It is based on controllers, like Proportional-Integral-Derivative or Linear-Quadratic controllers. Modern controllers, like learning controllers, are finding their way into these systems more and more. For example, learning controller can be controller that has learned about drivers’ driving behaviour and classified them into groups, such as group of aggressive or calm drivers. An adaptive controller adjusts itself when the input data is varying. By doing so, it learns how it should react and regulate. The interest in this technology is increasing again, after being dormant in the last 10-20 years. Modern learning controllers based on ideas from machine learning, have been developing very fast in recent years. There
are different strategies available for such controllers, for example Gaussian processes and Deep Neural Networks.

1.2 The Problem

A further development of LKA is a lane change assist (LCA) system. An LCA system can perform a lane change on the driver’s command, without the driver manoeuvring the vehicle. The command to initiate the lane change could, for example, be a blinker, however it is the driver’s responsibility to make sure that the target lane is empty. To make the lane change as comfortable as possible, and in a way that the driver had wanted, a learning controller can be used. This learning controller aims to classify driver’s lane change time by identifying the driving behaviour before reaching a highway, and using this information to make driver-familiar automatic lane changes. This can be achieved using techniques from machine learning.

The purpose of the thesis project is to investigate if it is possible to develop a learning controller for prediction of lane change times. Following question had to be answered:

- Is it possible to identify the driver’s driving behaviour and categorize it to a lane change time before the vehicle reaches a highway?

1.3 Limitations

For the purpose of this analysis, some limitations had to be set. The system is to be used in normal traffic situations, where nothing exceptional has occurred. This means that the controller does not have to learn how to manage an area with roadwork or traffic accidents. Today’s algorithms that detect the road lines through the camera can only detect two lines at the most. Since roadwork often consists of new road lines, in addition to the old ones, it would be very hard to have a controller that can handle these kind of situations.

The road before the highway must be long enough and contain enough events, such as curvatures. A curvature could perhaps be a roundabout, which would give better information than just regular turns, but that is not necessary.
Chapter 2

Theory

Supervised learning and unsupervised learning are two paradigms in the fields of machine learning. This section describes the theory of these two learning problems, as well as two different classification methodologies that have been used within the project.

2.1 Supervised versus Unsupervised Learning

Supervised learning is an approach of training a model with labeled data, i.e. features that have a known response.

Classification techniques use the training data to construct a decision rule that will assign labels to the features in a new set of data. Figure 2.1 presents a linear decision rule that divides data into two classes.

![Figure 2.1: Illustration of a linear decision rule that separates two sets of data, blue and red, into two classes. The illustration is in 2D.](image)

Figure 2.1: Illustration of a linear decision rule that separates two sets of data, blue and red, into two classes. The illustration is in 2D.
In classification techniques the responses are categorized. A categorical response is also known as qualitative response. For example, qualitative responses can be: gender, eye colors or truck brands. Classification techniques are used in many different fields of study, such as machine vision and biological applications. Common classification approaches are decision trees, naive Bayes, Artificial Neural Networks (ANN) and Support Vector Machine[1].

Regression techniques are using an infinite continuous set of values as the output. The aim of regression technique is to find a function that will fit the observed data as well as possible, in some sense. The simplest regression technique is linear regression. The approach of this technique is to model a relationship between dependent and independent data by fitting a linear equation to the observed data. Figure 2.2 visualizes the approach of linear regression[1]. Since the output of a regression technique is numerical, the response is also called a quantitative response. For example, quantitative response can be: outdoor temperature, a person’s weight or a person’s income. Methods that are used to handle regression problems are ANN and random forest.

Unsupervised learning is an approach of training a model to find structure of data that is unlabeled, i.e. that does not have a known response. Clustering is a technique that belongs to the group of unsupervised learning algorithms. The aim of a clustering method is to separate unlabeled data into groups, so called clusters. Common clustering techniques are Fuzzy clustering and Hierarchical clustering. Figure 2.3 presents a visualization of data separated into three clusters[1][2].

A data set that is both labeled and unlabeled leads to a semi-supervised learning problem. Solving these problems can be accomplished using both techniques from
supervised and unsupervised learning simultaneously[1].

Different machine learning algorithms are summarized and presented in the Table 2.1.

![Figure 2.3: Illustration of how clusters have been assigned to three different sets of unlabeled data presented as blue, red and yellow points. The illustration is in 2D.](image)

**Table 2.1: Algorithms in machine learning**

<table>
<thead>
<tr>
<th>Classification</th>
<th>Clustering</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support vector machine</td>
<td>Fuzzy clustering</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>Hierarchical clustering</td>
<td>Random forest</td>
</tr>
<tr>
<td>Decision trees</td>
<td>Linear regression</td>
<td></td>
</tr>
<tr>
<td>Artificial Neural Network</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.2 Classification methods

Two different classification techniques have been used in this project. Firstly, classification with naive Bayes because of its simplicity and ability to quickly classify the data. In order to see if the results from naive Bayes classification could possibly be improved, more complex classification method, ANN, was used. These two techniques are presented in the following subsections.

Classification was chosen because the data was labeled, i.e. lane change times for each data set was known. Other reason for using classification was because it can predict classes of seconds, represented with both integer and decimal numbers, or classes divided into groups, like fast and slow groups. Regression was not used since it can predict lane
change times in seconds, that are represented only with decimal number, i.e. not predict lane change times in whole seconds. To predict such detailed lane change times is not of interest.

2.2.1 Naive Bayes

Naive Bayes is a classification method that is widely used in different classification applications, due to its simplicity. It belongs to the classification group of probabilistic classifiers. A probabilistic classifier is a classifier that will not predict to which particular class a set of input features belongs to. Instead, it predicts the probability distribution of the given classes.

Naive Bayes model is based on Bayes’ theorem. In conditional probability, the product rule for any proposition a and b, is defined as:

$$ Pr(a \land b) = Pr(a|b)Pr(b) $$ (2.1)

The probabilities $Pr(a)$ and $Pr(b)$ are independent, and $Pr(a|b)$ is the probability of $a$, given $b$. This product rule can also be written as:

$$ Pr(a \land b) = Pr(b|a)Pr(a) $$ (2.2)

By equating (2.1) and (2.2), the probability of $b$, given $a$ is:

$$ Pr(b|a) = \frac{Pr(a|b)Pr(b)}{Pr(a)} $$ (2.3)

Equation (2.3) is the Bayes’ theorem[3].

Given input features, $(x_1, \ldots, x_n)$, of a particular instance with classes, $C_k$, where $n$ defines the number of feature’s and $k$ the number classes, Bayes theorem can be stated in the following way:

$$ Pr(C_k|x_1, \ldots, x_n) = \frac{Pr(C_k)Pr(x_1, \ldots, x_n|C_k)}{Pr(x_1, \ldots, x_n)} $$ (2.4)

Using product rule defined in (2.1), (2.4) can be rewritten in the following way:

$$ Pr(C_k|x_1, \ldots, x_n) = \frac{Pr(x_1, \ldots, x_n, C_k)}{Pr(x_1, \ldots, x_n)} $$ (2.5)

If we only look at the nominator in (2.5), and repeatedly apply (2.1), the nominator can be expanded to:
\[ Pr(x_1, \ldots, x_n, C_k) \]
\[ = Pr(x_1|x_2 \ldots, x_n, C_k) Pr(x_2, \ldots, x_n, C_k) \]
\[ \vdots \]
\[ = Pr(x_1|x_2 \ldots, x_n, C_k) Pr(x_2|x_3 \ldots, x_n, C_k) \ldots Pr(x_{n-1}|x_n, C_k) Pr(x_n|C_k) Pr(C_k) \]

Since all features depend on each other, a naive independence assumption has to be introduced, i.e. every feature is independent of every other feature. Under this assumption we have:

\[ Pr(x_i|x_{i+1}, \ldots, x_n, C_k) = Pr(x_i, C_k) \quad (2.6) \]

for any feature \( x_i \). Thus,

\[ Pr(x_1, \ldots, x_n, C_k) = Pr(C_k) \prod_{i=1}^{n} Pr(x_i|C_k) \quad (2.7) \]

By inserting (2.7) into (2.5), following expression is given:

\[ Pr(C_k|x_1, \ldots, x_n) = \frac{Pr(C_k) \prod_{i=1}^{n} Pr(x_i|C_k)}{Pr(x_1, \ldots, x_n)} \quad (2.8) \]

Since the denominator in (2.8), \( Pr(x_1, \ldots, x_n) \), will not be affected by the classes, this part will be seen only as a constant and therefore it can be neglected.

\[ Pr(C_k|x_1, \ldots, x_n) \propto Pr(C_k) \prod_{i=1}^{n} Pr(x_i|C_k) \quad (2.9) \]

The naive Bayes classifier is often combined with a decision rule. Maximum a posteriori (MAP) estimation is commonly used[4].

\[ \hat{C}_k = \arg \max_{k \in \{1, \ldots, K\}} Pr(C_k) \prod_{i=1}^{n} Pr(x_i|C_k) \quad (2.10) \]

The use of the naive Bayes classifier has a lot of advantages: 1) It works for both small and large data sets, 2) It has very fast computational time, since classifier can very easily and fast estimate the parameters in order to do a classification, 3) The classification time increases linearly with the size of training data, and 4) The classifier can for example manage noisy or missing attributes[3][4].

### 2.2.2 Artificial Neural Network

Artificial neural networks is a classification model that is based on the structure of a human brain. If the input signal, a stimuli, is strong enough, the neuron will fire giving
Figure 2.4: Mathematical model of a neuron. Steps I-IV illustrate the input links with a numeric weight on every link, the summation of the input links, the activation function and the output, respectively.

an output impulse. Generally, networks constitute of more than one neuron that are all connected to each other. Figure 2.4 presents the mathematical model of a neuron from an ANN. Neurons are also called units or nodes sometimes. Steps I-IV illustrate the input links, $a_k$, the summation of the input links, the activation function and the output, $b$, respectively. The neuron has a numeric weight, $w_k$, on every input link, $a_k$, which correspond to the strength of the connections. Step two in Figure 2.4 is the weighted sum of all inputs to the neuron. The output activation function $g$ then transforms the output from step II. It is usually either a hard threshold or a logistic function. If it is a hard threshold, the output is binary. In this case a neuron is called a perceptron. If it is a logistic function, see (2.11), the output is some value between 0 and 1. A logistic function can also be called sigmoid perceptron[3].

$$h(z) = \frac{1}{1 + e^{-z}}$$ (2.11)

There are two ways of forming a network, either with a feed-forward network or with a recurrent network. In a feed-forward network neurons are only connected in one direction, i.e. there are no loops. The neuron will receive an input only from another neuron and also deliver an output to another neuron. In a recurrent network neurons are connected in loops, i.e. the neuron will feed back the output to its own input[3].

In a network, whether it’s a feed-forward network or a recurrent network, neurons are arranged in an input layer, a number of hidden layers and an output layer. Figure 2.5 presents multilayer feed-forward network with an input layer with $n_x$ neurons, $j$ hidden layers and $k$ neurons in each hidden layer, and an output layer with $n_y$ neurons.
Figure 2.5: Presents a multilayer feed-forward ANN with an input layer with $n_x$ neurons, $j$ hidden layers and $k$ neurons in each hidden layer, and an output layer with $n_y$ neurons.

The rule of thumb states that it is better to have more hidden layers than a few. The number of hidden layers depends on the number of features. Typically, a network has between 5 – 100 hidden layers [2].

### 2.2.3 Training neural networks

Back-propagation is used when training an artificial neural network. When input data has been propagated from the input to the output layer through all the hidden layers, an error will be calculated at the output layer using an objective function[5]. The error is presented as the difference between the outcome from the network and the desired output. The error is then back-propagated by computing the gradients of the error with respect to the weights. This is done layer by layer, until the input layer is reached. After every calculation of the gradient, the weights between the layers have to be updated.

### 2.2.4 Stopping criterion for neural networks

Overfitting is a common problem in neural networks. It occurs when there are too many parameters in the model. To avoid overfitting and retrieve as good as possible model, cross-validation techniques are used. One approach is to use a set of test data on the network after each update. When the error between the network outcome and the correct output from the set of test data, is small enough, the training will be stopped. Value of the error limit is set by the user [2][6].
Chapter 3

Implementation

This chapter describes, in detail, the practical work and implementation method used during the project. The first section is an introduction of how sensor data was acquired and which features were chosen. Then, the analysis of retrieving features is described, and finally, the window and classification design.

The software program that was used during this project was MATLAB, developed by MathWorks. MATLAB contains many toolboxes, and in order to solve the classification problems in this master thesis project, Statistics and Machine Learning Toolbox, as well as Neural Network Toolbox were used. In the Statistics and Machine Learning Toolbox, naive Bayes classifier was used. Features and classes are given to the naive Bayes classifier, which creates a model. This model can then be verified by only giving features to the model. The model will predict a class for the given features. The ANN classifier in the Neural Network Toolbox works in the same way. The only difference is that in this case design parameters can also be selected, such as how many hidden layers a network should contain and how many neurons should be in each layer.

3.1 Acquiring sensor data

All sensors in a Scania truck are connected through controller area networks (CAN). CAN allows several different Electronic Control Units (ECU) in a truck to communicate with each other through the main control unit, COO. All control units are connected with six different CAN buses to the COO. These CAN-buses can either be blue, brown, green, orange, red or yellow, see Figure 3.1. Nowadays, the systems are connected to the any available CAN-bus colors. However, one still tries to group them based on the level of security. For example, in the red bus, the most safety-critical systems like engine and brake are connected.
Logged data collected during the field tests are used in this thesis project. The field tests were performed for approximately two months with the use of three different trucks. One truck had only one working shift and the two other trucks where being driven in two shifts, morning and afternoon. Each shift is approximately 8 hours long, including breaks. Every shift includes, at least, driving up to the highway, highway driving, and driving after the highway, i.e. driving from the highway to the truck garage in the city. Each shift gives rise to a data set of logged sensor information, which contains non-highway and highway driving. Also, an assumption had been made, that drivers do not change driving styles during their shifts.

The three trucks used in the field tests made 130 runs altogether. From these 130 runs, 5 runs were saved as validation data and 125 runs as training data. The validation data consist of only non-highway driving before highway, while training data consist of non-highway driving, both, before and after highway. Each validation run is representing a run that had a lane change time of 4s, 5s, 6s, 7s, or 8s. 3s were never used as a set of validation data, since the total set of runs contained only few runs with lane change time of 3s.

In order to distinguish between highway driving and non-highway driving, speed limits were set. All over 75km/h is seen as highway driving, and all under 60km/h is seen as non-highway driving. The gap between 75km/h and 60km/h is an uncertain zone, where it is hard to decide if it is highway or non-highway driving.

All sensor data from relevant situations were collected using CAN Crawler. CAN Crawler is a framework for analyzing and retrieving sensor data with self-written algorithm from a larger amount of logged data, without changing the existing data.
3.1.1 Sensor data

Road marks must be estimated in order to provide important information, such as the truck’s position in the lane. This is done with a third degree polynomial, see (3.1), which is used for defining the shape of the road marks. Scania has developed an algorithm that calculates the coefficients $a, b, k$ and $m$ in the third degree polynomial and with help of the camera, which is placed in the centre of a truck’s front, estimates a position in $y$ direction by knowing the $x$ position. The coordinate system is shown in Figure 3.2.

$$y = ax^3 + bx^2 + kx + m$$  \hspace{1cm} (3.1)

A road mark can only be estimated at a distance of approximately 50 m forward, due to the range of the camera.

![Cartesian coordinate system of a truck](image)

**Figure 3.2:** Cartesian coordinate system of a truck

There are various sensors available in a Scania truck, like sensors that describe the car dynamics, the driver behavior or the environment. In order to limit the project, only sensors that are believed to help in identifying driving behavior were selected. In Table 3.1, the selected sensors that were used in this project are presented.
There is a sensor in Scania trucks that can calculate the road curvature. This sensor uses road marks to make these calculations. Because situations like roundabouts and road intersections do not have any road marks, this sensor cannot be used in the scope of the project. Instead, an assumption had to be made that the driver is following the road in curvature when driving, i.e., the driver does not drive off the road when the curve appears. If this assumption is fulfilled, road curvature can be calculated using two sensors, yaw rate sensor and vehicle speed sensor, according to

\[
\text{Road curvature} = \frac{\text{Yaw rate}}{\text{Vehicle speed}}
\] (3.2)

There is a lateral acceleration sensor available in the truck. The signal from this sensor is noisier than the yaw rate sensor, due to vibrations in the truck. By using the same sensors as in the road curvature calculations, lateral acceleration can be calculated in the following way:

\[
\text{Lateral acceleration} = \text{Yaw rate} \times \text{Vehicle speed}
\] (3.3)

### 3.2 Preprocessing sensor data

Some of the selected signals were used to obtain data from different situations, such as lane changes. All the signals were also preprocessed to retrieve smoother data. This process will be described in the following subsections.
3.2.1 Filtering sensor data

All the signals, used in a truck during the tests, contained noise. The noise is thermal noise or vibrations that the sensors are exposed to. The sensors that are most exposed to thermal noise and vibrations, are acceleration and velocity sensors, both of which can measure in the longitudinal and lateral directions. Yaw rate sensor is also always noisy, see first subplot in Figure 3.3. A way of reducing the noise in these signals is by filtering the them through a low-pass filter. Two Butterworth [7][8][9] low-pass filters were designed and tested in this project. One with a cutoff frequency, $f_c$, of 8 Hz and the other one with a cutoff frequency of 4 Hz. The main reason behind designing two different filters with different cutoff frequencies was to examine what affect would it have on the results of the classification. The two selected cutoff frequencies fulfill the Nyquist sampling theorem, $f_s \geq 2f_c$, since the sampling frequency, $f_s$, of the signals is 20Hz. It should be noted that, in addition to reducing the noise, filtering the signals can cause other important information to be filtered away.

![Figure 3.3: Presents the yaw rate sensor from approximately 2 minutes. In the first subplot, the yaw rate sensor is presented without being filtered. In the subplot in the middle, the yaw rate sensor is presented after being filtered with a cutoff frequency of 8 Hz. The last subplot, the yaw rate sensor is presented after being filtered with a cutoff frequency of 4 Hz.](image)

3.2.2 Defining lane change

According to Trafikverket, the Swedish Transport Administration, there are two categories of highway standards in Sweden. There is a low road standard, with a total
width of 18.5 m, or a normal road standard with a width of 21.5 m. This includes traffic barriers and two roadways in each directions. A low road standard has a lane width of 3.25 m per lane, and a normal road standard a lane width of 3.5 m per lane[10].

A Scania truck uses the Forward Looking Camera in order to detect the road marks on lanes and to calculate the road width, the distance to the left and right road mark, as well as the angle to respective road marks. The angle depends only on the truck’s placement in the lane, given that the road marks are parallel. This camera is placed in the centre of the truck’s front, see Figure 3.4.

![Diagram of road marks and Scania truck](image)

**Figure 3.4**: Schematic view of the road marks. Distance to the road mark, $d_{right}$ and $d_{left}$, the road width, $w$, and angle to the right road mark, $\alpha_{right}$, are illustrated with respect to the centre of the truck.

A Scania truck’s width is approximately 2.6 m. A lane change is identified when,

$$2.6 \, m < d_{right} < 5.2 \, m$$

or

$$2.6 \, m < d_{left} < 5.2 \, m$$

depending on whether it is a lane change to the left or right lane. If the distance to the left or right road mark, as illustrated in Figure 3.4, is 2.6 m with respect to the center of the truck, i.e. one truck width, the truck has most likely started to make a lane change. This way, small movements in a lane can not be mistaken for a lane change. In order to
avoid a double lane change, a lane change that intersects two roadways, the maximum distance to the right and left road mark is set to 5.2 m, which is two truck widths.

3.2.3 Defining lane change time

A lane change time is calculated by using the angle to the left or right road mark. When a lane change has been identified, the implemented algorithm looks back in time until the angle to the left or right road mark is approximately zero. Afterwards, the algorithm finds the subsequent time when the angle to the road mark had become approximately zero again. By subtracting these two times, a lane change time is defined. If the value of lane change time is decimal number, it will be rounded to an integer.

3.2.4 Choosing lane change time interval

Figure 3.5 presents a histogram of lane change times of all lane changes that were identified from the recorded sensor data. It occurs very often that the shape of the road mark is incorrectly estimated, for example when a slip road occurs. A common error is that the system believes, for a short moment, that the distance to the left or right road mark is greater than it actually is. Often, these short moments are in the range 0–0.5 s, hence these false lane changes can be identified. In order to avoid these incorrect estimates and only use true lane changes, a maximum and minimum lane change time is set to 3 s and 10.5 s, respectively.

\[ 3 < t_{\text{lane change}} < 10.5 \]  

Both the lower and the upper limit is set by intuition, believing that a lane change can not be performed under 3 s and that the driver does not want a system assisting him in performing a lane change with a time over 10.5 s. Lane changes that are performed over 10.5 s are probably customized by the surrounding traffic situations and not performed in a normal way.
3.2.5 Define a lane change class

Since the runs often consisted of a series of lane changes with different times, one lane change time had to be defined for each run. This lane change time, that represents the run, can be calculated in three ways, by taking the median, mean or most frequent value of the series of lane change times.

Assume that, during the run, there have been four lane changes, with following lane change times:

\[ 4s, 4s, 6s, 10s \]

This run can not be represented with four different lane change times. Only one lane change time needs to be defined. The median value of the set of lane changes would be 5s, the mean 6s and the most frequent value 4s.

3.3 Window Design

When performing the classification with naive Bayes and ANN, two designs of windows with data were built and presented to the models. These windows will be explained in this section.

The main reason behind designing windows was because it would require too much space in the ECUs if the classification was performed in real time with a whole run. That big storage space was not available. Therefore, windows were created with data
representing short periods of times, which does not require as much storage space as a whole run would do.

### 3.3.1 Window with one sample in features

Window with only one time sample in a feature, $F$, is presented in Figure 3.6. The window also contains a class, $C$, which stands for defined lane change time calculated from data. The feature, $F$, can either be one signal from a sensor, or a combination of signals from various sensors.

![Figure 3.6: Illustrates a window with one sample, $N_1$, in a feature, $F$, and the class, $C$. The feature can be a signal from one sensor or a combination of signals from different sensors. The class stands for defined lane change time, calculated from data.](image)

When training the classifier, this type of windows are submitted into the classification methods. This is done in each run with all time samples in a set of data, i.e. 20 windows/second. When the classifier has been trained, windows with the set of validation data are constructed in the same way, but without any classes. When these windows are given to the naive Bayes and ANN classifiers, the classifier will predict lane change times for each window. Of course, it can predict different lane change times, since the training data contained different runs representing different lane change times.

### 3.3.2 Window with features representing time series

The window in Figure 3.7 contains four time samples in feature, $F$, and a class, $C$, that stands for the defined lane change time calculated from the data. Each feature represents a time series of data, and the feature can either contain one signal or a combination of signals. A window may contain several features.
The set of training data is packed this way, but it never uses the same samples in multiple windows. In the same way as in the window with only one sample, the windows are submitted into the classifications methods and the classifier is trained. Then, the set of validation data are presented to the classifier, but without any classes in the windows. The classifier will predict a lane change time for each window.

The original sampling frequency, $f_s$, of all signals was 20Hz. Three different time series windows were tested during the project. A window with 10s, 20s and 30s, respectively. Number of samples in each feature are:

\[
N(t_{\text{window}} = 10s) = 200 \\
N(t_{\text{window}} = 20s) = 400 \\
N(t_{\text{window}} = 30s) = 600
\]

Since the number of samples 200, 400 or 600 in each feature would be too high and would require a much longer training time, down-sampling had to be carried out. The down-sampling was performed with a factor of 4. Down-sampling is a method that decreases the sampling rate of a set of values, by keeping every $n^{th}$ value. For example, if a set of values,

\[1, 2, 3, 4, 5, 6, 7, 8, 9\]

were down-sampled with factor 4, following output would be obtained:

\[1, 5, 9\]

This down-sampling will not affect the result so much, since the original sampling frequency was 20Hz, which is a high sampling frequency in this context.
3.4 Classifier design

The signals that were used as input features to naive Bayes and ANN were all from the category *Vehicle dynamics* in Table 3.1, and the road curvature from the category *Environmental data*. All the input features were collected data from before highway driving behaviour. The output response of the methods is the estimated lane change class.

To quantify the performance of the classification, following measure was used:

\[
\text{Fraction of accurately predicted lane changes [\%]} = \frac{\text{Number of widows leading to correct classification}}{\text{Total number of windows}} \tag{3.5}
\]

3.4.1 Feature analysis

Various combinations of the selected signals were tested as input features to the different methods. Some data analysis were also made, in order to help choosing input features and see if there is any difference in the structure between the signals with different lane change times. By knowing the structure of the data, it becomes easier to plug it into the classification methods. If the structure of the data is more complex, then more data needs to be retrieved or other signals need to be used in order to extract relevant information.

The analysis are based on the calculations of features variances. The variances of two different features are plotted against each other in order to see if any structure in data can be distinguished between features representing different lane change times. This analysis was only performed on features representing time series, since the features representing time series consists of a larger amount of samples. The variance of a features can be only calculated if there is more than one sample. If a feature contains only one sample of a signal, the variance could not be calculated and this analysis could not be performed.

Various of combination of features where analysed, such as vehicle velocity, pedal signals, weight of the vehicle. It was very difficult to find any structure from the analyses of these features. The only structure that could possibly be discern from was the feature that contains data from lateral acceleration divided with road curvature, and the feature that contains longitudinal acceleration. These features represents time series of 20s. Figure 3.8 presents three subplots with variance of these two features, plotted one against each other, representing different lane change times defined with the most frequent value method. Each plot presents only data of two lane change times. The first subplot presents the variance of the data representing lane change times of 3s and 4s, the second subplot the data representing 5s and 6s, and the last one 7s and 8s.
Figure 3.8: Presents the variance plots of $a_x/\kappa$ vs. $a_y$. In the first subplot, the red dots are representing lane change time of 3s, and the blue dots of 4s. In the subplot in the middle, the red dots are representing lane change time of 5s and the blue dots of 6s. In the last subplot, red dots are representing lane change time of 7s and the blue dots of 8s.

As shown in Figure 3.8, it is hard to see any clear structure. The lane change times of 5s, 6s and 7s almost have the same structure of data. It is impossible to discern any difference between 3s and 4s, as it is represented as tall and narrow lump of data. The data representing 8s is a small lump in the left corner. The structure states that there might be possibilities to divide the data into three classes, fast, medium and slow, but it is not for sure because the structure is not so clear. The fast class representing lane change times of 3s and 4s, the medium class 5s, 6s and 7s and the slow class 8s. It is also noteworthy that there is much more data representing lane changes of 5s, 6s and 7s than that representing 3s, 4s and 8s. If there had been enough data of all lane change categories, then these plots could have looked more similar.

The other data, such as vehicle velocity and pedal signals did not show any correlation between them and different lane changes. One premise was that the weight of the truck could have a big impact on the lane change time, since all three trucks had different weights. Unfortunately, this analysis showed that the weight did not affect the lane changes. Neither did the brake pedal position, acceleration pedal position or the vehicle speed.
Chapter 4

Results

First, the results from classification with window with one sample in each feature will be presented, then the results from the windows with the feature that represents a time series and finally some analysis of the data in time series window.

4.1 Classification - features with one sample

4.1.1 Naive Bayes method

Various combinations of signals, along with three different methods of defining a lane change time were used in developing a classifier using a window with features containing one time sample. The classifier with the highest performance used seven different features in each window. Each feature represented one signal, namely: road curvature, lateral acceleration, longitudinal acceleration, longitudinal velocity, acceleration pedal signal, brake pedal signal, and weight of the vehicle. Filtering of the signals was tested when developing the classifier. The best results were achieved without any filtering. A reason for this can lie in the fact that important informations from the signals could have been filtered away, i.e. the information that would help to identify a driving behavior from non-highway driving.

The results of a windows with one time sample were the same, regardless of method chosen for defining a lane change time. Time that represents how a lane change is usually performed by the driver is the lane change time that occurred most often during one run. Therefore, the results from the classifier using most frequent value method have been chosen to be presented. By using this, approach lane changes of 9s and 10s never occurred.

Validation data was given in terms of before highway driving, representing a lane change times of 4s, 5s, 6s, 7s, and 8s. Validation data representing 3s was never given to the model, since the total set of runs contained only few runs with lane change time of 3s. The model predicts lane changes of these same times, see Table 4.1.
4.2 Classification - features representing time series

Only the results from windows of 20s will be presented. Windows with 10s and 30s were also tested, but the performance with the highest accuracy was achieved with a window of 20s. Also, the most frequent value method was used to define a lane change time. This choice was made with the same reasoning as in the classification using window with one time sample in each feature.

Filtering of the signals worsened the results in the same way as in the previous section. The best results were achieved without any filtering.

Five different sets of validation data were given to the models. Three different classification models were developed, the models will from these validation data sets predict either lane changes of six different classes (3s, 4s, 5s, 6s, 7s and 8s), lane changes of two classes (fast and slow) or of three classes (fast, slow and medium).

4.2.1 Naive Bayes method

Classification using the naive Bayes classifier was performed with various combinations of signals. The best results were achieved with only two input features, where one feature was data from the lateral acceleration signal divided with data from the road curvature signal, and the other was data from the longitudinal acceleration signal. Table 4.2 presents the validation results of this developed classifier for predicting lane change time in seconds.
The results show that the classifier, that predicts lane changes in seconds representing a lane changes of 8s, with the percentage of 70.8%. All the other validation sets, representing lane changes of 4s, 5s, 6s and 7s, were incorrectly predicted.

The validation results obtained by training the classifier with two different classes, fast and slow class, are presented in Table 4.3.

**Table 4.3: Validation distribution of predicting slow and fast lane changes**

<table>
<thead>
<tr>
<th>True</th>
<th>Predicted</th>
<th>Distribution of predicted lane change times [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fast</td>
<td>Slow</td>
</tr>
<tr>
<td>Fast (4s)</td>
<td>62.4</td>
<td>37.6</td>
</tr>
<tr>
<td>Fast (5s)</td>
<td>45.6</td>
<td>54.4</td>
</tr>
<tr>
<td>Slow (6s)</td>
<td>41.3</td>
<td>58.7</td>
</tr>
<tr>
<td>Slow (7s)</td>
<td>38.1</td>
<td>61.9</td>
</tr>
<tr>
<td>Slow (8s)</td>
<td>23.2</td>
<td>76.8</td>
</tr>
</tbody>
</table>

The only wrong classification was achieved with 5s. Both, 5s and 6s, represent boundaries between a fast and a slow lane change, which makes it not so strange that it is hard to distinguish between these two driving behaviours and categorize them into a fast lane change or a slow lane change.

Classification with three classes was also performed, by using fast, medium and slow lane change classes. A fast lane change class represented the lane changes of 3s and 4s, a medium class represented lane changes of 5s and 6s and a slow class represented lane changes of 7s and 8s. The validation results of this classification are presented in 4.4.
Table 4.4: Validation distribution of predicting slow, medium and fast lane changes

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Slow</th>
<th>Medium</th>
<th>Fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast (4s)</td>
<td>16.3</td>
<td>43.2</td>
<td>40.5</td>
</tr>
<tr>
<td>Medium (5s)</td>
<td>38.6</td>
<td>27.3</td>
<td>34.1</td>
</tr>
<tr>
<td>Medium (6s)</td>
<td>39.1</td>
<td>27.1</td>
<td>33.8</td>
</tr>
<tr>
<td>Slow (7s)</td>
<td>31.0</td>
<td>42.4</td>
<td>26.6</td>
</tr>
<tr>
<td>Slow (8s)</td>
<td>43.2</td>
<td>41.1</td>
<td>15.7</td>
</tr>
</tbody>
</table>

Only one prediction of a class was correctly done by the classifier, the validation data representing the slow class of 8s. The margin to the medium class was not big, approximately 2 percentage points, which is close to an incorrect classification.

In Table 4.5, classification using three classes is presented again. The only difference from the previous classification, presented in Table 4.4, is that the lane change time of 7s is moved over to the medium class. The validation results in Table 4.5, show that the model predicts the slow and fast class correctly, while the medium class is predicted incorrectly.

Table 4.5: Validation distribution of predicting slow, medium and fast lane changes

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Slow</th>
<th>Medium</th>
<th>Fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast (4s)</td>
<td>35.1</td>
<td>21.7</td>
<td>43.2</td>
</tr>
<tr>
<td>Medium (5s)</td>
<td>47.7</td>
<td>17.8</td>
<td>34.5</td>
</tr>
<tr>
<td>Medium (6s)</td>
<td>57.5</td>
<td>8.7</td>
<td>33.8</td>
</tr>
<tr>
<td>Medium (7s)</td>
<td>58.5</td>
<td>15.7</td>
<td>25.8</td>
</tr>
<tr>
<td>Slow (8s)</td>
<td>72.9</td>
<td>11.9</td>
<td>15.2</td>
</tr>
</tbody>
</table>

4.2.2 ANN method

Classification using ANN was performed with the same number of input features as in naive Bayes, i.e. with two input features using three signals. The network was trained with only two classes, fast class representing lane change times of 3s, 4s and 5s and slow
class representing lane change times of 6s, 7s and 8s. As in the naive Bayes classification with two classes, five sets of validation data were given to the network: two sets of data that represented fast lane changes and three sets of data that represented slow lane changes. The network predicted the lane changes of two categories, fast and slow.

Table 4.6 presents the validation results of the trained network. Various number of hidden layers and neurons were tested, but the best achieved results were with the network consisting of 3 hidden layers and 140 neurons in each layer.

**Table 4.6: Validation distribution of predicting slow and fast lane changes with ANN**

<table>
<thead>
<tr>
<th>True</th>
<th>Predicted</th>
<th>Distribution of predicted lane changes [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fast</td>
<td>Slow</td>
</tr>
<tr>
<td>Fast (4s)</td>
<td>70.2</td>
<td>29.8</td>
</tr>
<tr>
<td>Fast (5s)</td>
<td>50.3</td>
<td>49.7</td>
</tr>
<tr>
<td>Slow (6s)</td>
<td>74.9</td>
<td>25.1</td>
</tr>
<tr>
<td>Slow (7s)</td>
<td>42.8</td>
<td>57.2</td>
</tr>
<tr>
<td>Slow (8s)</td>
<td>41.5</td>
<td>58.5</td>
</tr>
</tbody>
</table>

ANN classification method for predicting fast and slow lane changes, predicted only the slow class representing 6s incorrectly. All others validation data that was given to the model was predicted correctly, both those data sets representing slow and fast classes.
Chapter 5

Discussion

This chapter is focused on discussion of the results, as well as some ideas on how they can be improved and, finally, some concluding remarks.

5.1 Classification results

Classification using a window with features of one sample in time did not give good results at all. No matter what validation data was given to the model, it always predicted the same lane change times. Classification using a window with time series features worked better, but not when predicting lane changes in whole seconds. The best results were achieved when predicting two classes corresponding to fast and slow lane changes. Using the naive Bayes and ANN classification techniques gave similar results. Both methods correctly predicted four, out of five, validation sets of data, but they had problems with predicting classes that lie between the fast and slow lane changes, i.e. around 5-6s. ANN predicted incorrectly the validation set of data representing the slow lane change of 6s, while naive Bayes predicted incorrectly the validation set of data representing the fast lane change of 5s. It is tricky to decide what a slow and fast lane change time is.

For the window representing a time series, the positive outcome of naive Bayes classification with the use of three classes was that it correctly predicted the fast and slow lane changes. However, medium lane change was always incorrectly predicted. ANN was also tested, but since the amount of data in the fast and slow classes was small, unlike in the medium classes, the results with this technique were even worse. This classifier predicted a medium class, regardless of validation data given to the network. This technique does not cope well with the cases where there is too big difference in data quantity between the classes, or when a class contains a small amount of data. On the other hand, naive Bayes technique seems to be able to handle these issues better.
5.2 Improving the classification results

By using both ANN and naive Bayes techniques during the tests, it has been shown that the choice of classification method does not have a big impact on the results in classification of two classes. The problem lies within the selected features, based on which it is hard to distinguish between a fast and slow class.

With the introduction of a third class, medium class, the classification was only performed with naive Bayes. In order to improve those results, another classification technique could possibly be tested that can handle various amounts of training data between the three classes, and hopefully better distinguish between fast, medium and slow lane changes than naive Bayes.

In order to improve the results, signals from other sensor could be used. Signals that could help identify the non-highway driving behavior and categorize it to a lane change time could be the information where inside the lane the truck has been driven, i.e. is the truck driven near the road marks or kept in the middle of the lane. Information regarding whether or not the driver is using the direction indicator in roundabouts and road intersections could be of interest. In sum, it can be concluded that it is important to find data that gives more information about driving behaviour, than those used in this project.

Another attempt to improve the results could be to only train models on non-highway runs that occur before the highway. There is a difference in driving behaviour of a well-rested driver and a tired driver. Since the data used in training the models consisted of non-highway driving before highway and after highway, and since the after-highway driving occurred at the end of a 6-7 hours working shift, there might have been different driving behaviours, due to driver fatigue.

In this project, windows were built without using window overlap. This may have divided a sequence with important information into two windows. If windows had overlapped, i.e. if each window also contained some sample points from the previous window, this problem could possibly have been avoided. If overlapping windows had been built, results could have been better.

Instead of estimating a lane change time for each window without taking the estimated lane change time of the previous window into account, recursive estimation can be performed. Recursive estimation uses the lane change time of the previous window, and together with the new information from the current window it updates the lane change time.

The amount of training data may also have had an impact on the results. If more training data could be retrieved, a better model could be learned. In particular, data from lane changes that occur less often, i.e. 3s, 4s and 8s, would need to be retrieved.
Intuitively, the weather conditions could also have an impact on the driving behaviour. When the weather conditions are difficult, i.e. when it rains or snows, a driver can have different driving behaviour compared to when the weather is good. Since the weather conditions were not known in the training data, this could be taken into consideration. If the weather conditions had been known, data consisting of different weather conditions during the same run could have been filtered out as only one driving behaviour was to be classified. So, by knowing the weather conditions, the results could possibly be improved.

5.3 Conclusions

The purpose of this study was to investigate if it is possible to develop a model that can predict lane change times before the vehicle reaches the highway. It was shown that there are some differences between driving behaviours representing fast and slow lane change times, and that by using machine learning techniques these differences may be found. Furthermore, in this analysis it has been shown that the information in the features used were not rich enough. In order to improve the classification performance, other features that contain more valuable information about driving behaviour, must be chosen.

One reason why it was hard to distinguish between driving behaviours representing different lane change classes, may lie within the fact that the assumption, stated in Section 3.1, was not fulfilled, i.e. the runs contained several different driving behaviours.
Bibliography


