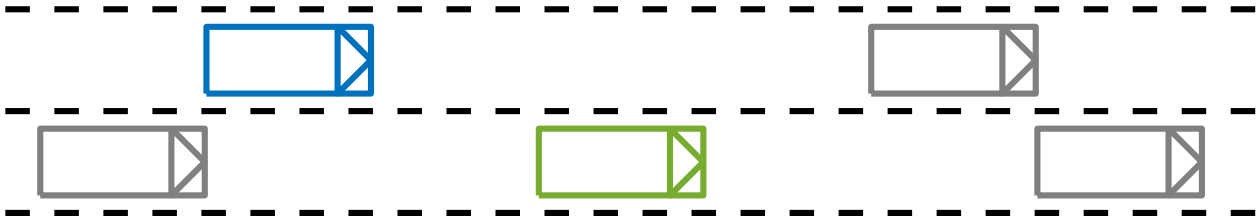


Modelling Object Movement Around an Ego Vehicle

Master's thesis in Applied Physics

IAN MACISAAC
JOHAN HULTBERG

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2019

MASTER'S THESIS IN APPLIED PHYSICS

Modelling Object Movement Around an Ego Vehicle

IAN MACISAAC
JOHAN HULTBERG

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2019

Modelling Object Movement Around an Ego Vehicle
IAN MACISAAC
JOHAN HULTBERG

Master's thesis 2019:100
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
The specific highway scenario that has been used in this thesis. The ego vehicle is shown in green, the target in blue and all other vehicles of interest in grey. The parameter to be modeled is the movement of the target vehicle relative the ego.

Chalmers Reproservice
Göteborg, Sweden 2019

Modelling Object Movement Around an Ego Vehicle
Master's thesis in Applied Physics
IAN MACISAAC
JOHAN HULTBERG
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

One problem for automated vehicles is that the traffic environment surrounding a vehicle is diverse and populated by a large set of interacting agents in the form of drivers. With a model for vehicle movement in traffic, developers will be able design autonomous vehicles with better path planning functionalities.

In this thesis models for vehicle movement around an ego vehicle are developed in a data-driven manner with different machine learning techniques. Analysis is done to find how accuracy is related to the prediction horizon, and to determine which features are most important. It is clear that more features are not always better as removing unnecessary features provides better results.

When comparing the models, baselines based on equations of motion with constant velocity or acceleration have been used. All methods provide better predictions compared to the baselines, and can make predictions for longer horizons. For longitudinal position prediction, results are promising. In latitudinal direction the results are less impressive, especially lane changes are difficult to predict, due to the low amount of lane changes in the training data.

That leads to analyzing in what other situations the prediction accuracy is limited by the data set, rather than by the model itself. For example how the accuracy is correlated with the speed of the ego vehicle. It is clear that the models performs best in situations that is well represented in the training data. To make a model that handles rare situations, a lot of data with those situations is needed.

Keywords: Autonomous Vehicles, Machine Learning, Data-driven, Feed Forward Neural Network, Linear Genetic Programming, Principal Components Analysis, Fishers Linear Discriminant, Regression, Path Prediction

Acknowledgements

Thanks go to all the people at Zenuity who shared their knowledge and time with us, in particular our supervisors Tobias Karlsson and Emre Sancar. Thanks for bearing with all our discussions and ideas, and helping us to get forward.

Also a big thanks to our examiner Krister Wolff who guided us through the academic processes during this thesis. Your office was always open for us, and you showed real interest in what we were doing.

Nomenclature

AD mode	**A**utonomous **D**riving mode
Ego Lag	The vehicle behind the ego vehicle
Ego Lead	The vehicle in front of the ego vehicle
Ego Vehicle	The vehicle from which sensor data is collected.
EQM	**EQ**uations of **M**otion
Feature	Here used as parameters used as input for the prediction models
FFNN	**F**eed **F**orward **N**eural **N**etwork
FLDA	**F**isher's **L**inear **D**iscriminant **A**nalysis
LGP	**L**inear **G**enetic **P**rogramming
LIME	**L**ocal **I**nterpretable **M**odel-Agnostic **E**xplanations
LR	**L**inear **R**egression
PCA	**P**rincipal **C**omponents **A**nalysis
PR	**P**olynomial **R**egression
STD	**ST**andard **D**eviation
Target Lead	The vehicle in front of the target vehicle
Target Vehicle	The vehicle of interest, whose position is predicted

Contents

1 Introduction and Background

Technological developments have allowed for the automation of an ever-growing number of every day tasks. Mowing grass and vacuuming are now commonly done by robots [1] and for the last decades dishwashers have been washing our dishes for us. With the increasing availability of computational resources and developments in machine learning techniques, even complex tasks such as driving are in line for automation. Automakers are promising autonomous vehicles within the next 4 years [2, 3, 4, 5]. Even so, there are still many obstacles which will need to be overcome before you will be waiting next to an autonomous vehicle in traffic [6].

One problem for autonomous vehicles is that the traffic environment surrounding a vehicle is diverse and populated by a large set of interacting agents. These agents, in the form of drivers, each have a unique way of interpreting and reacting to the traffic environment around them. Elements such as driver experience, mood, reaction time and intention, among others, are important factors influencing their participation in the traffic situation. This represents a difficulty for autonomous vehicle development in that these driver specific elements are not explicitly visible to vehicle sensors. Because of this it has been difficult for autonomous vehicle designers to incorporate such elements into decision making algorithms.

When considering a behavior of interest, movement in traffic for example, the question then becomes how well can this behavior be modelled using the sensor data that is currently available? How accurately can that model predict future positions? Which features from the sensor data are the most important? Which features are missing?

1.1 Purpose

In this thesis we develop and evaluate different models for vehicle movement around an ego vehicle, analyze their accuracy for different prediction horizons, and determine which features are most important. Model creation is made in a data-driven manner via different machine learning techniques.

We also investigate how the specific characteristics of the of the data-set may influence model performance. This is evaluated in an effort to provide guidance on building a data-set that best represents the problem to be solved.

With a model for vehicle movement in traffic, developers will be able design autonomous vehicles with better path planning functionalities. Such model can increase both safety and comfort for vehicle occupants, as well as resulting in more energy efficient ways of interacting in traffic.

1.2 Limitations

The scope of the project is limited to modeling and predicting the behavior of a vehicle slightly behind and to the left of the ego vehicle, *referred to as the target vehicle*, with a prediction window of 5s. Analysis is further limited to situations in which the ego vehicles adaptive cruise control with steer assist (AD mode) is activated and the ego vehicle velocity is over $5\,\mathrm{m/s}$.

The data provided to each model is limited to data on the vehicle in front of the target, the ego vehicle as well as vehicles to the front and rear of the ego vehicle. No explicit road information is included. More detailed information about the input parameters can be found in section 3.1.2.

Furthermore, the choice of modelling methods is limited to Linear Regression (LR), Polynomial Regression (PR), Linear Genetic Programming (LGP), Principal Component Analysis (PCA) with Fisher Linear Discriminant Analysis (FLDA) and Feed Forward Neural Networks (FFNN).

1.3 Motivation

Being able to model traffic participants and make accurate predictions about their behavior is an essential element in designing autonomous vehicle systems. Whether it be for path planning, collision avoidance, or even something more specific like cut-in detection, an accurate prediction of a vehicles future position is the first step.

The traffic situation presented here for analysis was chosen as it represents a situation that is interesting when planning lane changes, but is also general enough to be extended for other goals. To this end, further limiting the traffic situations by defining a minimum ego velocity and requiring the ego vehicle AD mode to be

on was done to maintain a mix of kinematics and driver behavior, compared to a higher percentage of driver behavior at lower speeds, and to isolate the interaction between an autonomous vehicle and the vehicles around it. These limitations also help to limit the complexity of the initial problem. The methods proposed here are general enough to extended to model other traffic scenarios by extracting training data from those.

The modeling methods proposed are methods within machine learning that are used in contexts such as facial recognition and function approximation. Linear and polynomial regression were chosen as they represent standard methods within statistical modeling [7]. Principle component analysis and linear genetic programming were chosen as we had experience with them from other domains and were interested in seeing how they would perform modeling traffic situations. The feed forward neural network was chosen for its capability in modeling highly complex systems and the fact that neural networks often outperform traditional methods in many applications [8, 9, 10]. As a baseline to judge the performance of the different modeling methods, the kinematic equations of motion with constant acceleration or velocity was used as in [11].

1.4 Related Work

A number of studies have been done which are similar to this thesis in that they deal with the movement of vehicles in traffic. In most cases, these studies can be separated into two categories depending on the observers viewpoint, what we will call stationary respective participatory. The stationary case involves an observer which is stationary and removed from the situation which it is observing. Studies in this category have been done to predict driver behavior at traffic bottlenecks [12], predict red light running at crossings [13] and predict lane changes at on and of ramps [14, 15].

Studies in the participatory category on the other hand observe the environment around an ego vehicle from the perspective of the vehicle itself. This is what has been done in this thesis and this is what is of most interest for in-vehicle decision making systems. It is common for studies in this category to focus on predicting maneuvers, such as cut-ins and lane changes, rather than positions. In [16], ego lane cut-in prediction was done by using case based methods that match a cut-in against one that has been stored by the algorithm. Lane change predictions have been done with models based on Support Vector Machines [17, 18], fuzzy neural networks [19], and a combination of a Naïve Bayes Algorithm, Hidden Markov Model and Gaussian Filtering [20]. This last study also includes an analysis of the prediction horizon and feature importance and they have been able to precisely detect lane changes of other trafc participants up to 2.2s before the lane assignment changes. They have also found that the lateral velocity relative to the lane, the relative velocity of the preceding car and the lateral offset to the lane center are the most discriminant features for lane change recognition.

In some cases, studies combine maneuver and position prediction. One such study focuses on trajectory prediction by first anticipating lane change maneuvers and then predicting vehicle position using a Gaussian Mixture Regression [21]. Vehicle position is measured in a curvilinear coordinate system which follows the curvature of the road. The study focuses on predicting the position of vehicles in both the longitudinal, or road, direction and latitudinal direction of this coordinate system. Different models for the longitudinal and lateral position prediction are used depending on which maneuver is expected.

While maneuver prediction can be valuable, the problem is that there are few maneuvers that are well defined. Is a vehicle moving halfway into the next lane and then back again a lane change? Probably not, but it can still be of interest for behavior algorithms that use position predictions to make decisions.

A study by [11] uses Gaussian Mixture Regression to predict the longitudinal position of the vehicle in front of the ego vehicle for prediction horizons of between 0.1 and 5.0 seconds. To make the prediction, the velocity and acceleration of both the vehicle to be modelled and its lead vehicle, along with the distance between them, are included as features.

1.5 Contributions

This thesis presents a data-driven approach to predict vehicle movement in traffic from the perspective of an ego vehicle. A prediction of this type has many applications within the development of autonomous vehicle systems, such as facilitating cut-in and lane change detection. Specifically, this thesis contributes with the following four conclusions. Modeling methods exist which are better able to extract meaning from the data than equations of motion, model complexity does not predict model performance, more features to not imply better model performance and data driven modeling can be limited by the dataset.

2 Theory

The thesis compares 5 different modeling methods, namely Linear Regression (LR), Polynomial Regression (PR), Linear Genetic Programming (LGP), Principal Component Analysis (PCA) with Fisher Linear Discriminant Analysis (FLDA) and Feed Forward Neural Networks (FFNN). The first two methods are based on least-squares regression, which is considered as a standard method [22] and will not be explained further. The remaining methods are explained briefly in this section to provide a reader unfamiliar with these methods a better understanding.

In the end of this chapter a tool called LIME is introduced. This tool helps analyzing feature influence in FFNNs.

2.1 Genetic Algorithms and Linear Genetic Programming

Linear genetic programming (LGP) is a programming technique by which a linear set of instructions is evolved using a genetic algorithm (GA). GA is an optimization algorithm inspired by biology that evolves a solution with mechanisms such as reproduction, mutation and selection [23, 24]. This section will begin with describing a basic GA, before going into the details of an LGP where the main differences are highlighted.

A GA can be used to find optima in a multivariate function $f(x_1, ..., x_n)$ where all allowed values $\mathbf{x}$ is called the search space. To be able to apply the GA to a problem like this the variables x_i gets encoded in a string, called a chromosome. Historically this variables has been encoded as binary digits, but there exists other encoding schemes, and the digits building up the chromosome are called genes.

When the GA is initialized, a set of N chromosomes are generated with random numbers. This set is called the first generation. The set is then decoded to obtain their corresponding values, and later evaluated. During the evaluation each individual i is assigned a fitness value F_i. This value will be used for selecting individuals for reproduction in order to maximize the fitness.

To form the next generation of chromosomes, inspired by biological sexual reproduction, individuals are selected in pairs and new individuals are formed by combining their genes. The selection should favor individuals with higher fitness, but in a non-deterministic way; individuals with lower fitness can also contain valuable genes.

The reproduction starts selecting two individuals and combine their genes with crossover, which results int two new chromosomes with the beginning from one of the parents chromosomes and the end from the other one, see figure 2.1. The point where the chromosomes is cut is randomly chosen.

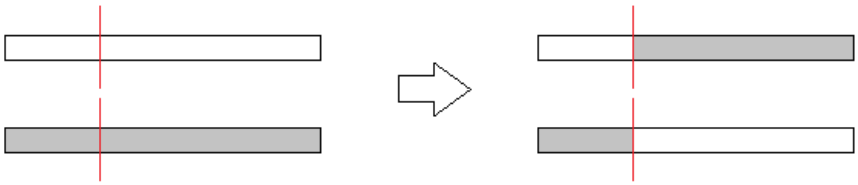


Figure 2.1: *Single point, length conserving serving crossover.*

An important concept for evolution, that can introduce new genomes into the population, is mutation. After the crossover all genes in the two new chromosomes are, at a low probability, changed. The probability is called mutation rate, and is usually set higher than mutations occurs in nature.

Selection, crossover and mutation is repeated until N new chromosomes are generated. This is the next generation that replaces the first one. Then the process starts over and new generations can be evolved until an acceptable solution is found.

The big difference when moving to LGPs is the introduction of registers and instructions. Instead of evolving chromosomes that is solutions to a multivariate functions, chromosomes in LGP represent a sequence of instructions that modifies the registers. The instructions can for example be mathematical operations and the registers constants and variables. The whole chromosome then represents an equation. Except for small changes, as different genes now can have different ranges of valid values during initialization and mutation, most concepts from GA are similar.

Registers and Instructions

Registers and instruction sets are two concepts that LGP relies on. Every LGP algorithm has two registers, a variable register and a constant register. As the names imply, the constant register is not allowed to be modified after creation, whereas the variable register is. The desired inputs to the equation are contained within the variable register together with some empty spaces to be used for storage and output by the algorithm. The constant register contains only a few constants, which can be used by the algorithm to constrict any new constant it might need. The instruction set contains all operators available to the algorithm.

Chromosome

Individuals within the population are represented as chromosomes. A chromosome is constructed of genes where each gene represent an instruction. There are many ways to encode instructions as genes, where the specific implementation often depends on the problem. Common to all encodings however is the requirement that the operands, operator and destination registers are identified.

A common encoding is to represent each gene in the chromosome as a series of four integers. The first integer represents the destination location within the variable register. This is where the result of the instruction sequence is stored. The second integer represents the operator to be used and the third and fourth integers represent which registers the operator will act upon.

Selection

In selecting which individuals of a population will be allowed to crossover, or reproduce, it is not a good idea to only select the best performing individuals. For complex functions it is often the case that even poorly performing individuals can contain valuable genetic information. Selection is therefore done in a more stochastic manner.

Stochastic selection is comprised of two main sub-types: roulette wheel selection and tournament selection. In roulette wheel selection the probability for one individual to be selected is equal to the individuals fitness divided by the sum of all individuals fitness, shown in the following equation.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

An advantage of roulette wheel selection is that there is only one parameter, the selection probability, which needs to be set.

In tournament selection, two additional parameters have to be set: tournament size n and tournament probability p. n individuals are randomly selected and ranked in order of their fitness. The two with lowest fitness start to "fight". The one of these two with highest fitness wins with a probability p. The winner gets to fight against the individual with third lowest fitness and so on until there is one individual left.

Crossover

When two parents have been selected they can generate children by crossover. This is when the two original chromosomes are cut in two, or more, pieces and then put together with each other, see figure 2.2.

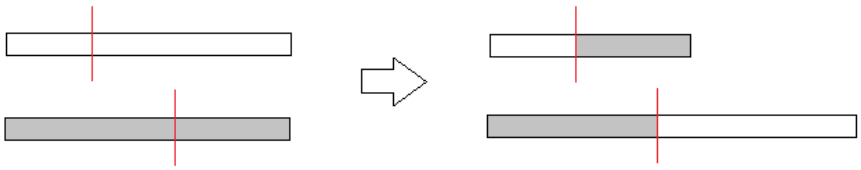


Figure 2.2: *Single point, non-length conserving serving crossover.*

Crossover points are chosen randomly, but for LGP they are placed between instructions for compatibility. If the two chromosomes are cut at the same point the crossover is length conserving, but if the points are generated individually the chromosomes can change length and generate more complex sequences of instructions.

The latter one is often used in LGP, if there is no reason to believe that the initial length of the chromosomes is optimal.

Crossover is an effective way to explore parts of the search space that are represented in the population, but it can not add new information to the chromosomes.

Mutation

As in nature mutations are needed to drive evolution forward. By randomly changing a number in a chromosome, the result can be completely different. In most cases, the mutation will not add any useful information to the individual and can often even decrease an individuals fitness. This is offset by the potential positive contributions mutation can make to an individuals genome. By improving the performance, the individual will have greater chance to reproduce and spread the mutation.

In nature, many generations are required for a mutation to evolve and spread. In genetic algorithms the mutation probability is usually set much higher to speed up the process.

Length Penalty

When using non length conserving crossover the chromosomes has a tendency to grow larger and larger. More instructions can give a more complex solution, but it is common that not all instructions are useful as many of them will not affect the output. These unused genes can, to a point, be healthy for the simulation as they can store valuable information that might come to use in later generations. But if the chromosomes only continue to grow the simulation will become very computationally heavy. One way to overcome this is to include a chromosome length penalty in the fitness parameter of an individual.

There are two types of penalties, a hard limit where all chromosomes longer than that get zero fitness or a soft limit where the fitness gets reduced proportional to the chromosomes length.

Elitism

One problem introduced by crossover and mutation is that the best individual of a generation may be destroyed via these processes and not make it to the next generation. To avoid this problem, one ore more copies of the best individual can be transferred directly to the next generation. This is called elitism.

Elite Population

The LGP algorithm initializes with random chromosomes. These evolve to fairly good solutions in just a few generations, often finding different local optima quickly. Getting out of the local optima and find better solutions is harder, sometimes requiring the simulation to be restarted.

Different runs result in different solutions as this is a stochastic method. Some local optima represent better starting points than others when trying to find better solutions. By exploring multiple local optima, population diversity can be encouraged and the population becomes better prepared for finding better solutions.

One way to do this is to introduce an elite population. If the desired population size is n, n simulations are initialized and run for N generations. The elite population is assembled by taking the best individual from each of the n simulations. Then a new simulation is started with the elite population as starting point.

Batch training

If the set of training data is large the training will become very computationally expensive. The number of operations per generation will be in the order of instructions per chromosome times population size times number of input output pairs.

To ease this, the training data is divided into batches, or random subsets, inspired by neural networks. By evaluating a generation only on a small part of the data fewer operations are required. When all batches have been evaluated, the order of the data is shuffled and new batches are created to avoid getting stuck in local optima.

If an individual has a better fitness for the batch than the individual inserted by elitism, this single chromosome is evaluated with the whole training set to determine its ranking within the population.

2.2 Principal Components and Fisher's Linear Discriminant Analysis

It is possible to store all unique situations in a data-set in a multidimensional feature space. By assuming that identical situations result in identical outcomes, new data can be evaluated against the outcome of the stored situation with the most similarity. Similarity here is represented as the inverse of the euclidean distance between the new situation and the stored situation. Problems arise when dealing with multiple features as the dimensionality of the feature space increases. A large feature space requires a significant quantity of training data to fill and causes the evaluation of new data, i.e. the calculation of the distance to all other points, to be very computationally intensive.

There are multiple methods to reduce the dimensionality of the feature space which are often used together. In the following sections two of them are presented, namely Principal Components Analysis and Fisher's Linear Discriminant Analysis.

Principal Component Analysis

If every input variable is considered as a dimension in a space, PCA tries do reduce the dimensions without losing too much information by extracting the most important feature combinations and expressing them as *principal components*. These principal components represent a set of orthogonal variables [25, 26] that are the basis for a new feature space. Principal components are found as the n most prominent eigenvectors of the co-variance matrix of the training set, where prominence is defined as the amount of variance in the input data that an eigenvector is responsible for.

When predicting new data the new inputs is transformed into the same lower dimensional space. The training data with the shortest euclidean distance to this new input is chosen as the most similar case, and is assumed to have a similar output.

Fisher's Linear Discriminant Analysis

While PCA is an unsupervised method, FLDA is supervised by making a transformation that depends on the output [27]. The goal with FLDA is to minimize the scattering within a class in the new space and maximize the distance between classes [28]. For data that is not classified, the scattering is minimized for points with an difference in output less than ϵ_1, and maximized for points with outputs that differs more than ϵ_2.

2.3 Feed Forward Neural Networks

A feed forward neural network is a biologically inspired construction consisting of layers of *neurons* or *nodes*[23]. These networks are called feed forward because information moves in only one direction; from the input towards the output of the network. This contrasts with some other neural network architectures in which recurrences are introduced, allowing information to travel in both directions [29]. Figure 2.3 shows the architecture of a FFNN with 3 layers. This type of network is commonly referred to as a multi-layer perceptron [30].

The basic building block of a neural network is the node. A node is a mathematical construction which sums the product of each input with a specific weight, together with a constant bias term . This sum is then passed through an "activation function" which bounds the output of each node and introduces non-linearities into the network [23], shown in the following equation.

$$f(x, w) = \sigma\left(\sum_i x_i w_i + b\right) \tag{2.1}$$

Here, σ represents the activation, x_i the input, w_i the weight and b the bias.

In a FFNN, these nodes are arranged into layers, called hidden layers as they are isolated between the input and output layers of the network [29]. Every node in each layer is connected to every node in the preceding and succeeding layers. Because of this, such networks are often referred to as being densely connected.

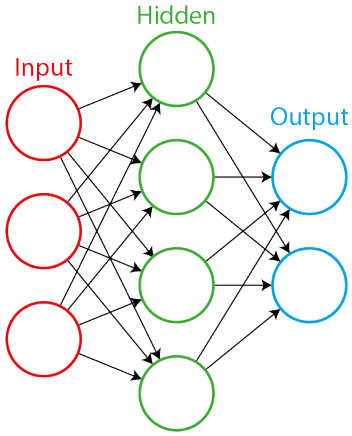


Figure 2.3: *A simple feed forward neural network with three layers.*

Learning

FFNN's learn by iteratively adjusting layer weights to minimize some arbitrary loss function. The loss function uses the error, calculated as the difference between the networks output and the expected output, to quantify the model's performance over the entire training data-set. Weight adjustment then takes place by distributing the result of this loss function back through each layer. In this way, each node is made aware of its contribution to the total loss and the gradient of the loss function can be calculated. A nodes weights are then adjusted using the gradient to minimize the loss function. The most common loss functions for regression tasks include mean absolute error and mean squared error.

Overfitting

A FFNN is capable of modeling extremely complex relationships. In fact, the universal approximation theory states that neural network is capable of approximating any function [29]. While this represents one of the strengths of using a FFNN over other methods, it can also lead to the network modelling the noise and specific detail of the training data. This produces a model which performs very well when evaluated on the training data-set, but which is unable generalize and as such performs quite poorly on new, unseen data. See figure 2.4.

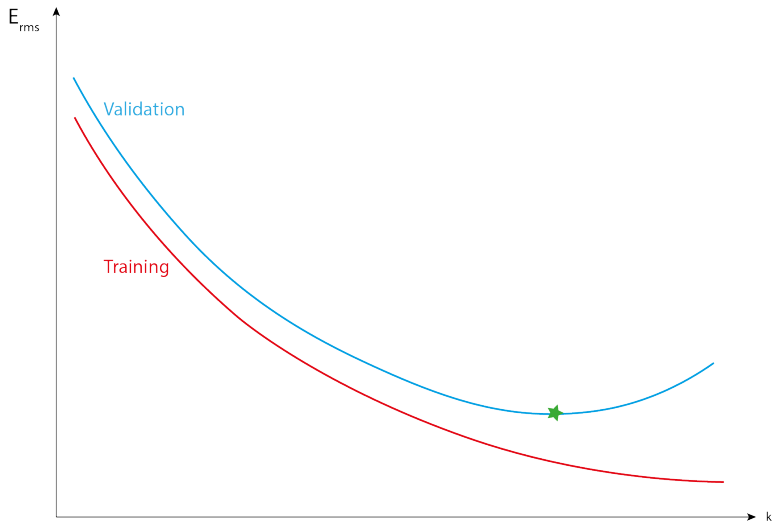


Figure 2.4: *Error over training epochs for training and validation data sets. At one point the error can start to increase for the validation set while the error continues to decrease for the training set.*

There are a few different methods for avoiding overfitting, the most common being the application of drop out. Drop out is when a certain percentage of nodes in the hidden layer are randomly removed from the

training process. This forces each remaining node to be more general in its operation, keeping it from becoming overspecialized for some data-set specific parameter.

Network Design

Designing a FFNN is an iterative process and there are no accepted standards on how a network should be configured for any specific problem. There are however a few rules of thumb that can be used to find a reasonable starting point. Often times a FFNN is configured to have a single hidden layer with a number of neurons somewhere between the number of inputs and the number of outputs [31]. This has been modified in relation to the learning ability and storage capacity of a network to be

$$N_h = \sqrt{(m+2)N} + 2\sqrt{\frac{N}{m+2}}$$

where N_o is the number of outputs and N_s the number of samples in the training data-set [32].

An additional rule of thumb, proposed by [30] places an upper bound on the number of neurons that will not result in overfitting. This can be written as

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

where N_i is the number of inputs and α a scaling factor, often between 2 and 10.

2.4 LIME

One drawback associated with more complex modelling techniques, such as FFNNs, is that it is difficult to see how the models are using different input features. To gain insight into the inner workings of such complex regression models, a technique called Local Interpretable Model-Agnostic Explanations (LIME) has been developed. With this technique, perturbations are introduced to the input and the model response to these perturbations is fit locally to a sparse linear model [33].

3 Method

The work done to model vehicles was divided into three main categories: data preparation, model construction and analysis. Data preparation included processing, to filter noisy signals and ensure track continuity, and feature extraction to build the data set to be used as input to the models.

The data set was divided into a training set and a validation set, with around 70 respective 30 percent of the data points. Models based on five different methods: LGP, PCA-FLDA, LR, PR and FFNN, were trained with the training set, and validated with the validation set together with physical models based on the equations of motion. The relative feature importance of the different models was then evaluated.

3.1 Data Preparation

Before extracting features for the data sets the log files went through replay, path construction and feature smoothing. In a real-time application replay and smoothing may not be possible, but here it was used to lower the noise levels and facilitate proof of concept.

3.1.1 Preprocessing

The ego vehicle measurements of its surroundings are locally very accurate, but that accuracy decreases with distance. To account for this and improve road information that is far ahead the drives were replayed to correct the road information data with "future" information. A transformation of the coordinate system was also done in the replay. This transformation changes from a Cartesian coordinate system with ego at $x = y = 0$ and x in the direction of ego, to a curvilinear road coordinate system with ego at $x = 0$ and $y =$lateral offset in lane, and x in the road direction. In this way longitudinal position for an object always means distance away at the road, and longitudinal speed is the speed in the roads direction.

It happens that tracks are not continuous, for example when the sensors loose track of an object for a few frames. To overcome this, all tracks were cut into pieces at frames where the longitudinal or lateral position changes more than $v_{max} * dt + \epsilon$. Here, v_{max} is the max speed in respective direction, dt is the time per frame and ϵ is the magnitude of the noise. Segments shorter than 50 frames are deleted.

The end of each segment was then compared to the beginning of all segments starting within 0-400 frames of the segment end point. The first segment was extrapolated to the second segments start frame and the second segment was extrapolated back to the first segments end frame. In these two points a fitness value was obtained as $f = error^{-1}$ where the error was the sum of the difference in speed and position in the two points in both lateral and longitudinal direction. A hard limit for possible connections was drawn for each feature to limit not-physical connections. This limit was based on a maximum feasible acceleration change and an error margin.

All fitness values were saved in a fitness matrix of size n*n, where n is the number of segments. The value at f_{ij} was the fitness of a connection from the end of segment i to the start of segment j. All impossible connections were marked with NaN. Paths were then built up starting with the best fitness. When a connection was made from i to j, row i and colon j was set to NaN so no more connections were allowed between these points.

When the paths were constructed they were saved together with the object data of interest. Where gaps exist in the paths, the data was interpolated. To remove noise, the data vectors were then smoothed with a moving average.

3.1.2 Feature Extraction

Features were extracted from five vehicles in the scene visualized in figure 3.1; the ego vehicle shown as green, the target vehicle shown as blue, the vehicle behind ego, and the two vehicles in front of ego respective target. The latter three are shown as grey.

The features extracted from each of these vehicles are listed in table 3.1, together with four extra features. For the ego vehicle though no heading or curvature was archived. Also, for ego the vehicle width, length and longitudinal position are constant. In total 39 non-constant features were extracted.

For each set of features the future position of the target vehicle was gathered as output, for prediction horizons from 0.5 s to 5 s in steps of 0.5 seconds.

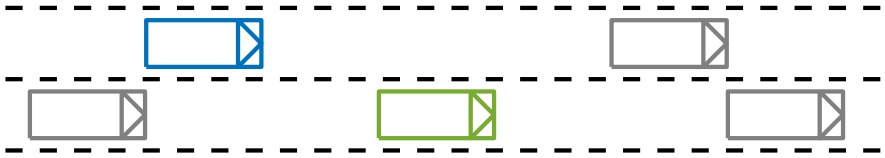


Figure 3.1: *The ego vehicle is shown in green, the target in blue and all other vehicles in grey. The movement of the target vehicle in relation to the ego is the parameter to be modeled.*

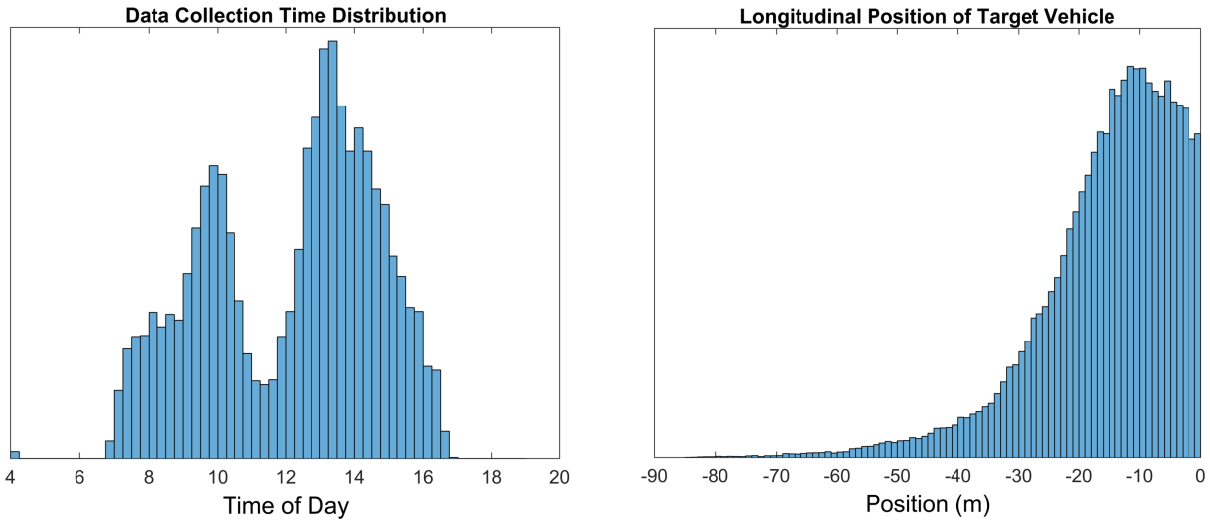
The original data has a sample frequency of 40 Hz. If taking every data point the result would be a huge amount of training data, and because of the short time intervals points close in time would be very similar. Instead the training and validation data was sampled at 4 Hz, in this way the same amount of data can represent a larger amount of driving.

Table 3.1: Features extracted from each vehicle of interest together with four extra features

Vehicle Specific Features	**Other Features**
Longitudinal Position	Ego Left Blinker
Latitudinal Position	Ego Right Blinker
Velocity	Speed Limit
Acceleration	Lane Width
Heading	
Curvature	
Vehicle Width	
Vehicle Length	

3.1.3 Statistics

The data used for analysis was provided by Zenuity AB and consisted of driving records for three test vehicles driven in the greater Gothenburg area from October 2017 to March 2018. During this period the three test vehicles logged 137 hours of driving data. The majority of those hours were logged between 8 and 17. Figure 3.2a shows how the logs are distributed within the day. There are two peaks, one around 10 and one around 14, which implies that the majority of the log files represent driving in off peak traffic.

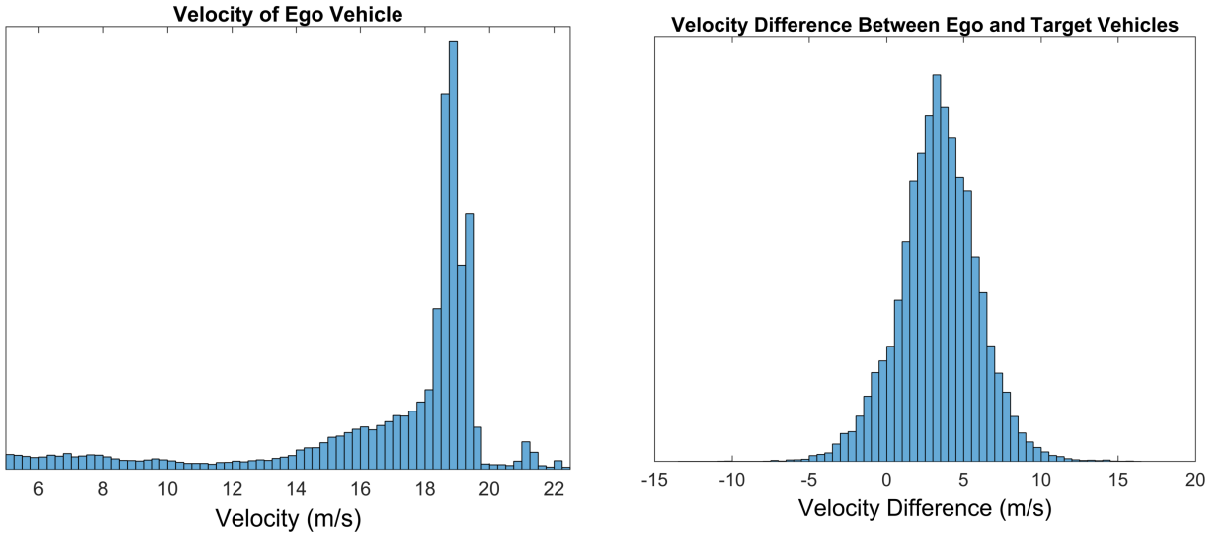


(a) (b)

Figure 3.2: *(a) shows the distribution of what time of the day the data is collected. Working hours is over represented in the data set. (b) shows the distribution of the longitudinal position of the target vehicle.*

The longitudinal position for the target vehicle is distributed as in figure 3.2b with a peek at -10 m. Cars with a positive position are not of interest according to the limitations, which is the reason for the cutoff at 0 m.

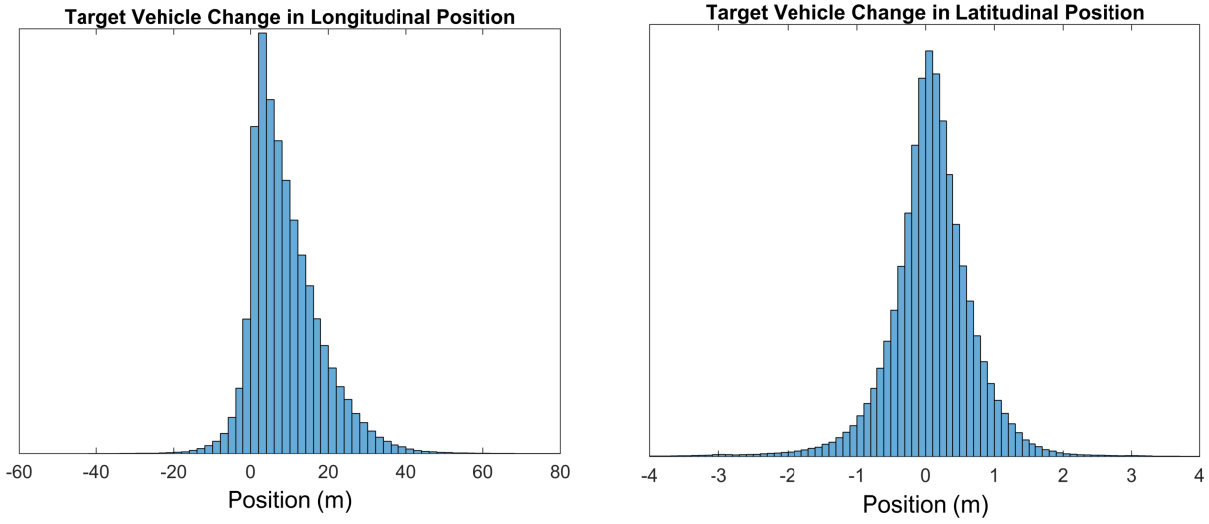
Figure 3.3a shows how the ego vehicle velocity is distributed in the data set. Observe the peak at 19 m/s, velocities between 18 and 20 m/s are over represented in the data. In figure 3.3b the distribution off the target speed relative ego is shown. The target vehicle usually has a slightly higher speed than ego.



(a) (b)

Figure 3.3: *(a) shows the distribution of ego velocity in the data set. (b) shows the distribution of the difference between the velocity of the target vehicle and the ego vehicle. Even if the ego vehicle often drives as fast as the speed limit allows it generally goes a bit slower than the surrounding traffic*

How the change of target longitudinal and latitudinal position are distributed in the data set is shown in figure 3.4a respectively 3.4b. All prediction horizons are represented in this data set which explains the narrow peaks. The longitudinal distribution is fairly wide, but positive values is over represented. The latitudinal distribution on the other hand is narrow. Most vehicles in the data keep their position in the lane, and the number of targets that end up with their center position in another lane after 5 s is around 2%.



(a) (b)

Figure 3.4: *Distribution of the change in target vehicle position for all prediction horizons relative the ego vehicle. (a) shows the longitudinal distribution, and (b) the latitudinal.*

3.2 Models

3.2.1 Linear and Polynomial Regression

A linear regression is done by describing the data as

$$X = a_0 + \sum_{i=1}^{N} a_i x_i, \tag{3.1}$$

where the constants a are determined by a least squares method. The constants describe how the output is related to the input features, and by standardizing the data before the fitting the constants can be used to rank the features by influence.

To include non-linearities, the training data was also fit to a polynomial of the third degree as

$$X = a_0 + \sum_{i=1}^{N} \left(a_i x_i + \sum_{j=i}^{N} \left(a_{ij} x_i x_j + \sum_{k=j}^{N} \left(a_{ijk} x_i x_j x_k \right) \right) \right), \tag{3.2}$$

where the constants a are determined as before.

3.2.2 Linear Genetic Programming

As mentioned in the background section, Linear Genetic Programming generates linear sequences of instructions called chromosomes, here with the purpose of building an equation. An operand set register was created consisting of all the input features, 3 empty spaces and the constants 1, 3 and 10. Positions in this register were then modified according to the instructions in the chromosomes by the operators addition, subtraction, division, multiplication, power, exp, cosine and sinus. The final equation was taken as at the first index in the operand register after all instructions had been performed. Despite the name of the method the resulting equation does not have to be linear, instead it can be rather a linear sequence of non-linear instructions.

The training process of the LGP can be very time consuming, so the model was created only for predictions of 5 seconds and subsequently modified to include other prediction horizons by dividing the output by 5 and multiplying it by the desired prediction horizon.

3.2.3 Principal Components and Fisher's Linear Discriminant Analysis

Dimensional reduction can be used to narrow down the search space for other methods as FFNN, but here it was used as a prediction method.

PCA reduces the dimensions to keep as much of the variance as possible, but does not take the output into account. FLDA reduces the dimensions even more with a transformation that minimize the within class scatter and maximize the in between class scatter. Here the output was continuous and not divided into classes. Instead the scatter was minimized between all data points with $|\Delta\text{output}| < \epsilon_{close}$ and maximized for data points with $|\Delta\text{output}| > \epsilon_{far}$. Here, ϵ_{close} and ϵ_{far} were both set to 0.01 meters.

The validation could then be transformed into the same space as the training data. By then calculating the euclidean distance to all training points it was then possible to rank situations by similarity.

3.2.4 Neural Network

Regression was performed here using a FFNN to predict the longitudinal and latitudinal position of the target vehicle. For each position, a different network was trained and evaluated as to limit the complexity of the model. Because the best performing architecture of FFNN is highly dependant on the subject to be modelled, the network architecture was evolved iteratively through trial and error.

The network used for both the longitudinal and latitudinal predictions consisted of a single hidden layer of 126 nodes with an activation function called Parametric Rectified Linear Unit (PReLU). Drop out regularization was applied at a rate of 0.1 to control overfitting. The input data to the network was standardized by mean centering a feature and normalizing it by its standard deviation.

For every improvement in validation loss, an error histogram was saved as well as a copy of the model and weights at that point. Once the training was complete a plot comparing the training and validation loss was saved along with a copy of the final model.

The network was initialized and trained 20 times and the models with the best validation performance was selected as the final models for analysis.

3.3 Evaluation

The evaluation was divided into three parts. First a comparison of the models was done for different prediction horizons from 0.5 to 5 seconds, in 0.5 second intervals. This was followed by an analysis of the feature importance for the different models and an analysis of the model performance related in relation to the data-set.

3.3.1 Comparing Methods

Two metrics were chosen for comparison: mean error and standard deviation. The mean error is to control the centering of the error distribution. A good predictor evaluated on data containing stochastic noise, should be able to produce an error with a distribution mean close to zero. Standard deviation is used to measure the width of the distribution.

The 90% confidence interval of the prediction was used in a limited capacity to evaluate a few specific use cases of the FFNN. This was done to better illustrate the results in the context of a highway situation.

Baseline

Constant velocity or constant acceleration are assumptions commonly used as baseline measures by publications on similar topics [11]. Here, these two assumptions will be used as baselines for the comparison of the different models. By supposing that velocities will be constant predictions can be made with

$$\Delta x = v\Delta t, \tag{3.3}$$

and if assuming constant acceleration the prediction can be made with

$$\Delta x = v\Delta t + \frac{1}{2}a\Delta t^2. \tag{3.4}$$

3.3.2 Feature Ranking

The relative importance of the different input features for different models were analyzed in different ways depending on the model. For all features, however, were normalized to unit variance to avoid skewing the feature ranking in favor of features with larger variances.

After normalization the feature were ranked for the LR model by looking at their contribution to the linear fit. This is simply the absolute value of the coefficient vector multiplied element-wise with the feature vector. In a similar way the features were ranked with PCA-FLDA by looking at the the numbers in the transformation matrix for the first principal components.

With the LGP it was difficult to perform a ranking, but some conclusions can be drawn from looking at the generated function.

From FFNN models, a hint about the feature importance can be seen in the weights between the input layer and the first hidden layer, but due to the complexity of the model no certain conclusions can be drawn from this. Instead a method called LIME was used to analyze how the output is correlated with the input features.

After ranking the features, the models were retrained with fewer and fewer features, removing the least with each iteration. When doing this with PR the ranking from LR is used. For the PR this is impossible to do with all features as the number of unknown constants is in the order of number of features to the power of 3, and quickly needs more data points what is available.

Limitation by the Data Set

To analyze how the models are performing in different situations, and analyze how the performance is limited by the data set, the models were evaluated on different subsets of the data, namely subsets organized by ego speed and relative target speed. This was then plotted against the distribution for the corresponding data subset.

4 Results

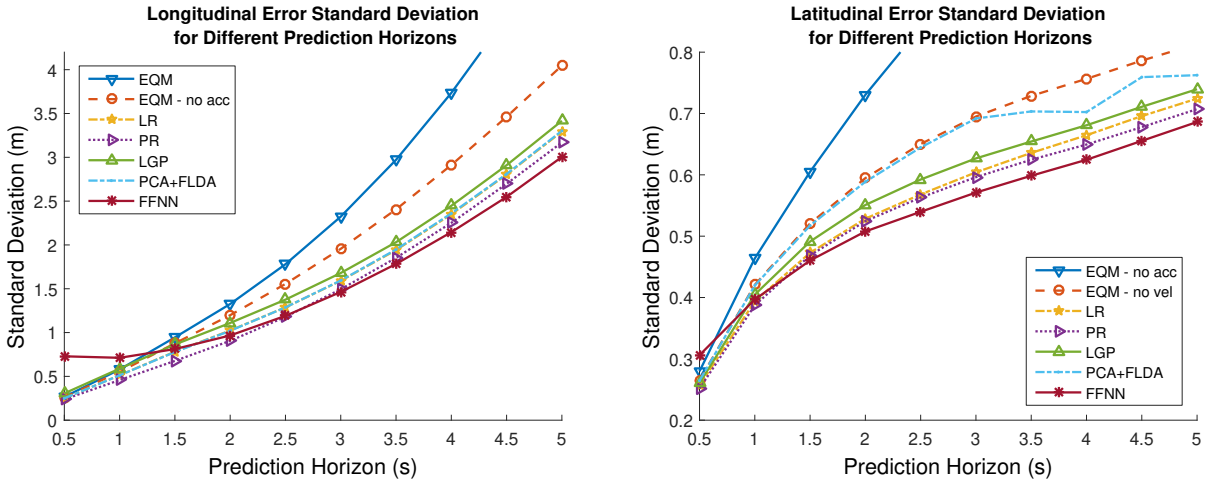
This chapter is a presentation of the results from the modeling. First there is a comparison of the models performance and how that depends on the prediction horizon. The chapter also includes an analyze of the amount of influence of the features in the models, followed by an analyze of how the training data affects the performance. Finally the most accurate model is evaluated in four test case scenarios to better contextualize the results.

4.1 Model Performance

The longitudinal and latitudinal prediction performance of the different models can be seen in figure 4.1 as the STD of the prediction error compared to the prediction horizon. The figure shows the prediction results for prediction horizons between 0.5 and 5 seconds, in 0.5 second intervals, for each position component.

In figure 4.1a the performance of the models in predicting the longitudinal position component of the target vehicle is shown. For prediction from approximately 2.5 seconds and onwards the FFNN model has the lowest STD. Below that prediction horizon the polynomial regression model predicts with a lower STD, with all other models predicting with a lower STD for horizons below approximately 1.25 seconds. Predictions with the highest STD belong to the equation of motion model. Note how removing the acceleration term results in a significant reduction in the STD for longer prediction horizons. Note also how the STD curves for the linear regression and PCA+FLDA overlap each other almost exactly for all prediction horizons.

Figure 4.1b shows the performance of the models in predicting the latitudinal position of the target vehicle. Here, as with the longitudinal case, the FFNN and polynomial regression models predict with the lowest STD. Compared to prediction of the longitudinal position, the FFNN model has the lowest prediction error down to prediction horizons of approximately 1 second. Continuing this comparison, one can see that the linear regression and PCA+FLDA models do not overlap, providing different results from each other. The equation of motion model produces a prediction with the highest STD for almost all prediction horizons, improving most when both the velocity and acceleration terms are removed, i.e. steady state.



(a) (b)

Figure 4.1: *The prediction performance of the different models for prediction horizons between 0.5 and 5 seconds. Performance is measured as the standard deviation of the prediction error. (a) shows the longitudinal position prediction performance. (b) shows the latitudinal prediction performance.*

4.2 Feature Ranking

For the LR, PCA-FLDA and FFNN models, the input features are ranked in order of their relative influence on the output. These models receive all features as compared to the LGP model which only accepts a subset of the features, determined through its evolutionary process. Figures 4.2 through 4.6 show for each of the first three

models the relative importance of the 15 most influential features. Common to these plots, the longitudinal and latitudinal positions of the vehicles have been referred to as the X and Y positions, respectively. Equation (4.1) and (4.2) represents the LGP model and shows which features it has deemed important. The relative feature importance for the LGP can not be quantified in the same way and is not included here.

Using LIME to evaluate the FFNN model resulted in the feature rankings presented in figures 4.2 and 4.3. In figure 4.2a the feature rankings for the longitudinal FFNN model evaluated with all prediction horizons is shown. Observe how the 4 most important features in the list, namely the target velocity, target X position, ego velocity and prediction horizon (dt) can be combined to result in the EQM model with constant velocity. Note also how quickly the feature importance decreases when moving down the list. Figure 4.2b shows the same feature ranking for the FFNN model evaluated with a prediction horizon of 5 seconds. This list is not similar the previous case aside from the disappearance of the dt feature. Note how the importance of the target X position is amplified with the presence of the dt feature.

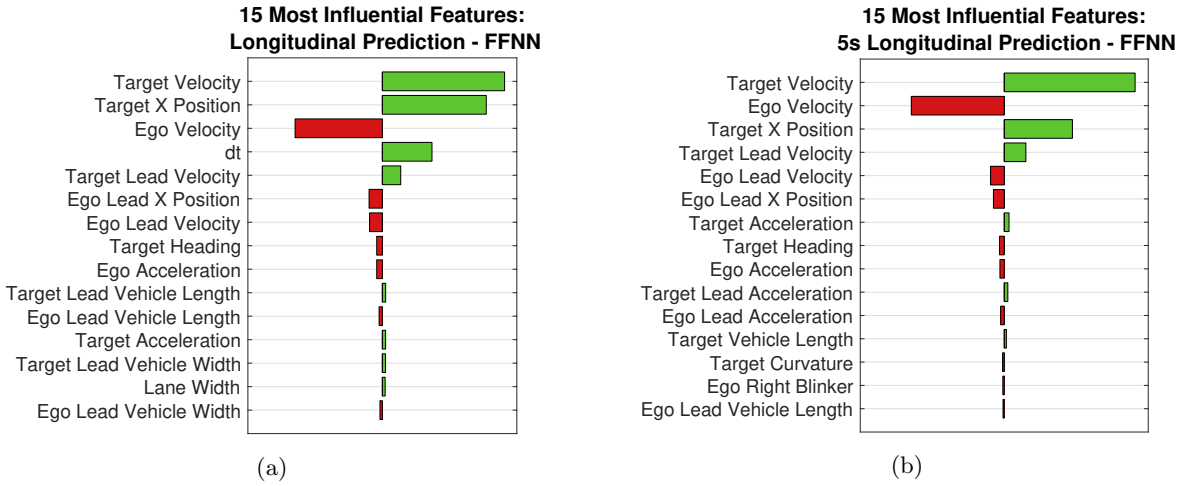


Figure 4.2: *Input features ranked by relative importance as calculated by LIME for the longitudinal FFNN model. Here, dt is the prediction horizon and Xposition is the longitudinal position. (a) shows the feature ranking for all prediction horizons. (b) shows the feature ranking for a prediction horizon of 5 seconds.*

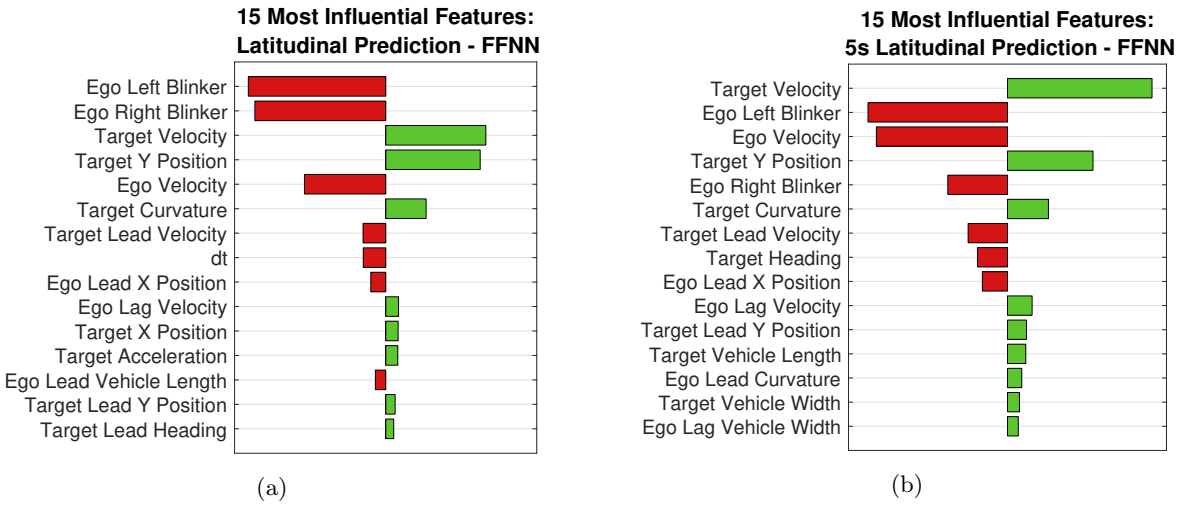


Figure 4.3: *Input features ranked by relative importance as calculated by LIME for the latitudinal FFNN model. Here, dt is the prediction horizon and Yposition is the latitudinal position. (a) shows the feature ranking for all prediction horizons. (b) Shows the feature ranking for a prediction horizon of 5 seconds.*

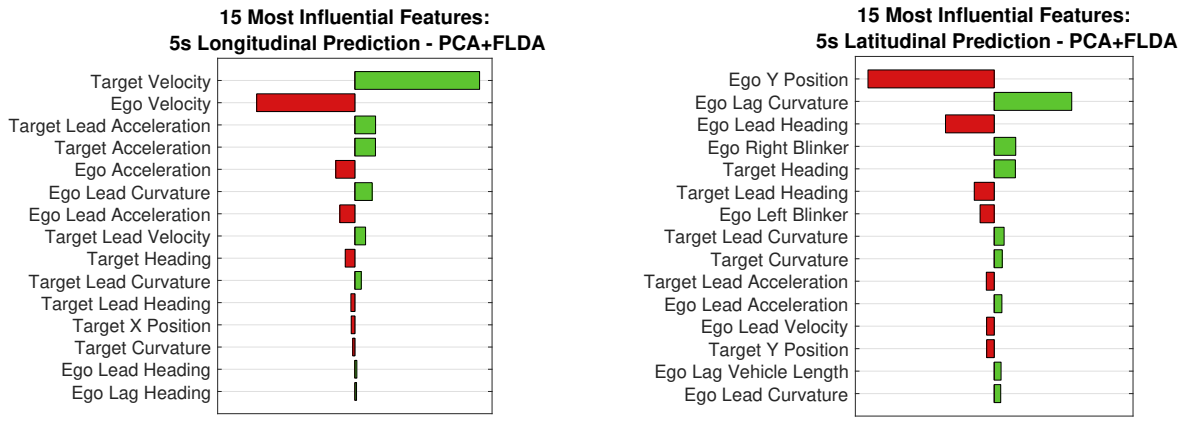
The feature ranking of the latitudinal FFNN can be seen in figure 4.3. Note in 4.3a how the top two features consist of information about the ego blinker status. Also note how low the dt term comes on the list, behind

such features as target curvature and target lead velocity.

The feature list for 5 second prediction, figure 4.3b, is now a fair bit different than that shown in 4.3a for all prediction horizons. In both cases it is interesting to note how the feature importance is not as quick to decrease as one moves down the list as for the longitudinal feature rankings in figure 4.2

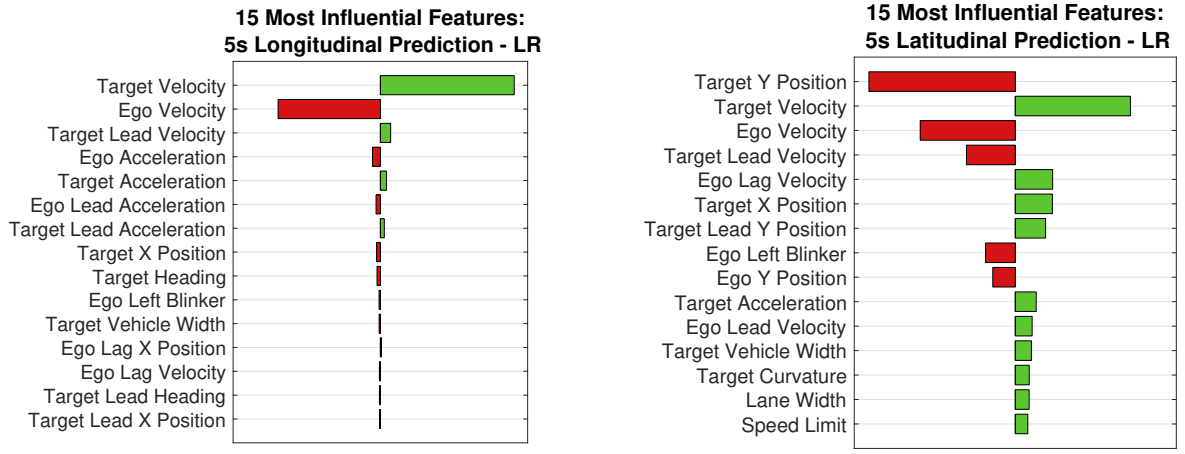
To get a ranking of the features using PCA-FLDA the weights in the first principal component is used. This component turned out to be the only one that gave useful information about the result. The weights describes how the features vary with the output, and due to the normalization of the features the magnitude of the weights can be seen as an importance value for the features. The ranking of the 15 most important features can be seen in figure 4.4, for longitudinal respective latitudinal prediction. Note in figure 4.4a the outstanding influence from the top two features, that together with feature 4 and 5 well represents EQM. Interesting is that information about the two lead vehicles as acceleration and curvature seems to be important.

In figure 4.4b it can be seen that the initial latitudinal position of ego has a big role. Followed by curvature and heading from almost all cars, even the car behind ego, together with ego blinker information.



(a) (b)

Figure 4.4: *Input features ranked by relative importance obtained as the weights in the first principal component. Xposition is the longitudinal position and Yposition is the latitudinal position. (a) shows the feature ranking for position prediction in the longitudinal direction. (b) shows instead the feature ranking for position prediction in the latitudinal direction.*



(a) (b)

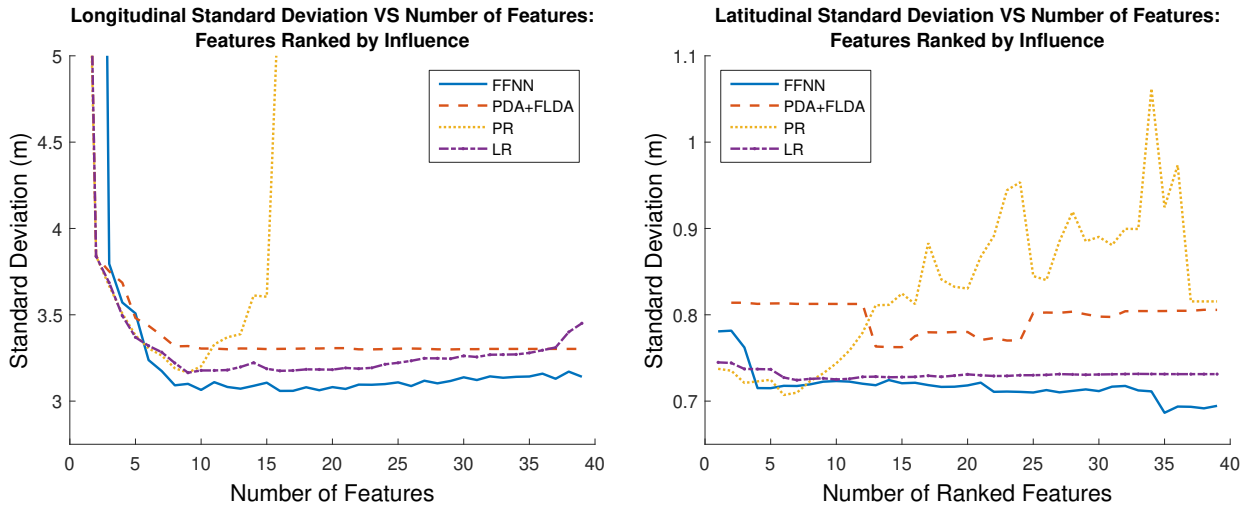
Figure 4.5: *Input features ranked by relative importance obtained as the coefficients in the linear least square regression of the normalized features. Xposition is the longitudinal position and Yposition is the latitudinal position. (a) shows the feature ranking for position prediction in the longitudinal direction. (b) shows instead the feature ranking for position prediction in the latitudinal direction.*

Similar to the PCA-FLDA a ranking is obtained from linear regression by first normalize the features and then take the magnitude of the coefficients. The result can be seen in figure 4.5.

Note in 4.5a the outstanding influence from the top two features, that together with feature 4 and 5 well represents EQM. Interesting is that acceleration information about the two lead vehicles, together with the velocity of one of them, seem to be useful for the model. Observe also the the left blinker of ego seem to have influence on the future longitudinal position of the target vehicle.

For the latitudinal prediction in 4.5b the initial target position has a big role, followed by the velocity of all cars, even the car behind ego, together with ego blinker information. Interesting is that both lane with and speed limit shows up in the top 15 features.

What happens with the accuracy if only the most important features is used? To answer that question the models are retained with one removed feature at the time, always the least important remaining one according to the rankings above. The result can be seen in figure 4.6 for longitudinal and latitudinal prediction respectively. Note that the models has individual ranking, and can have different subsets of features. Also note that no ranking has been made for PR, instead the ranking order from LR is used.



(a) (b)

Figure 4.6: *How the error distribution is related to the number of features used in the models. Here the features has been ranked after importance, individually for the methods, and then the models has been retrained on a subset of the most important ones. The figure shows how the STD is related to the number of features included in the subset. In (a) it is done for longitudinal position prediction and in (b) latitudinal. The ranking is done for individually for longitudinal and latitudinal.*

When it comes to the LGP, no ranking of importance of the features is made. However, the generated equations can by themselves show feature importance to some extent. The function generated for longitudinal prediction at a horizon of 5 seconds, and used as the LGP model, is as follows:

$$\begin{aligned}\Delta x =& 2tLeadAcc - 3tLeadHead + 3targetAcc - 15targetCurv \\ &+ \cos\left(\frac{egoLeftBlinker}{EgoLagHead}\right) - 6tLeadHead - 12targetHead) \\ &- 2egoLeadAcceleration + 2targetVel \\ &- 3\left(\frac{targetVel}{10}\right)^{\cos(egoLeftBlinker)} - 3\frac{targetVel}{egoLeadLong},\end{aligned} \tag{4.1}$$

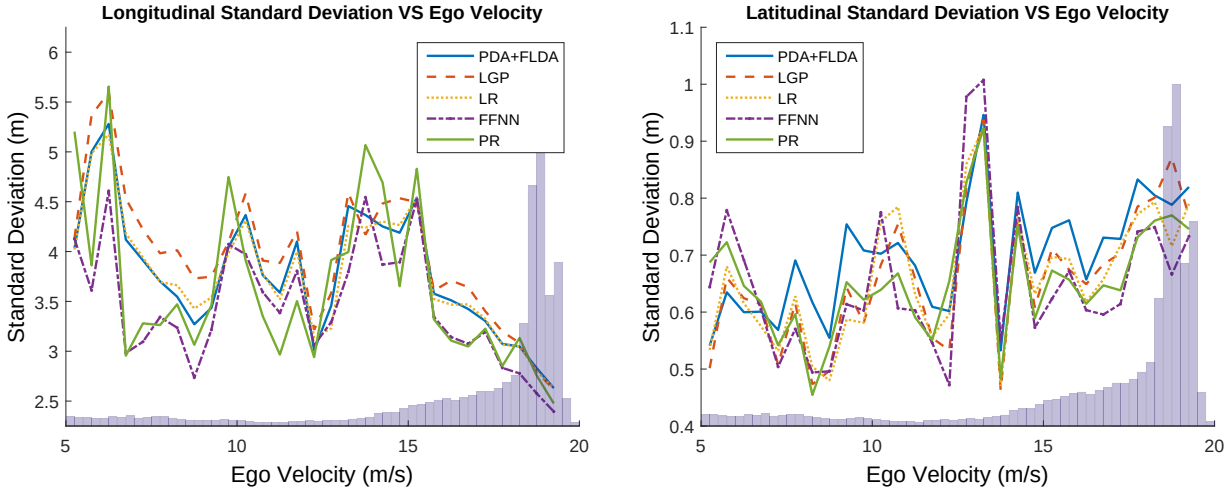
and the corresponding equation generated for the latitudinal model is:

$$\begin{aligned}\Delta y =& 0.2 + \frac{tLeadLat}{10} - tLeadHead - 2^{targetVel} \\ & \exp\left(-targetLat - \left(2.5 - \frac{tLeadLat}{10}\right)^{targetVel}\right) \\ & - targetLat \exp(tLeadHead - \frac{tLeadLat}{10} - 0.2).\end{aligned} \tag{4.2}$$

4.3 Data Influence on Model Performance

Model performance is also influenced by the specific characteristics of the training and validation data sets. The prediction STD for each model, with the exception of the EQM models, are shown as a function of the ego vehicle velocity, figure 4.7, and the velocity difference between the target and ego vehicles, figure 4.8. Overlayed onto these plots are the distributions of the ego velocity respective the target - ego velocity difference as represented in the data. Emphasized here is not how any individual method performs per se, but rather the trends that can be observed in relation to the distributions. In all cases note how the areas with the most erratic behavior correspond to areas where the data is sparse.

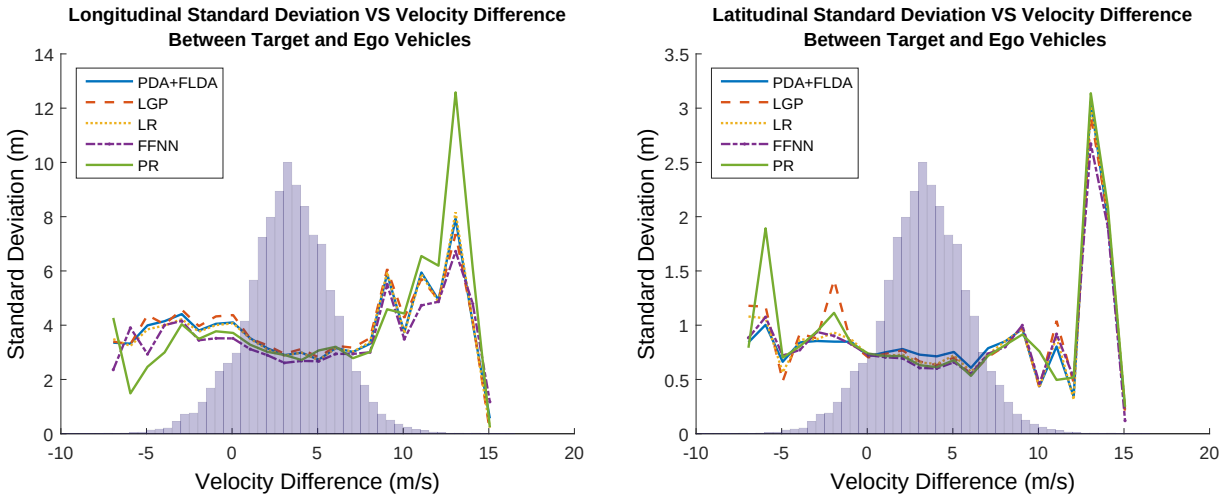
In figure 4.7, the longitudinal and latitudinal STD is plotted against the ego vehicle velocity. Figure 4.7a shows the longitudinal STD for all models fluctuating violently until ego velocities of about 15 m/s at which point the STD begins to decrease. Note how this corresponds to the distribution of ego velocities in the data set. Figure 4.7b shows a similar behavior in the latitudinal dimension, with fluctuations in STD also stabilize at an ego velocity of 15 m/s, Here however the STD does not decrease but rather increases with the distribution of ego velocities in the data set.



(a) (b)

Figure 4.7: *(a) shows the longitudinal STD for all models except the EQM models, plot against the ego velocity. Behind this plot, the ego velocity distribution of the data set is visible. (b) shows the latitudinal STD in a similar manner as in (a).*

A similar behavior is also visible in figure 4.8. Here, the longitudinal and latitudinal STD for all methods, excluding the EQM models, are plotted against the difference between the target and ego velocities. In both cases, fluctuations in model STD calm and consolidate for velocity differences between approximately 0 m/s and 8 m/s. Figure 4.8a shows that the longitudinal STD for all models seems to be less volatile for velocities on the negative edge of the distribution versus the positive edge. Compare this to figure 4.8b which shows the latitudinal STD as almost equally volatile regardless of which edge of the distribution a model is evaluated on.



(a) (b)

Figure 4.8: *(a) shows the longitudinal STD as a function of the velocity difference between the target and ego vehicles. (b) shows the latitudinal STD distribution as a function of the same. Behind both (a) and (b), the velocity difference distribution of the data set is visible.*

4.4 Test Cases with the FFNN

To better contextualize these results, the 5 second predictions by the FFNN model are shown in in four situations in figures 4.9 to 4.12 below. In each of the four figures, the ego vehicle is shown in green, the target in blue and the targets actual future position in dashed blue. The prediction is shown as a heat map derived from the error distribution, encompassed by a dashed red line representing the edge of the 90% confidence interval. In other words, the targets actual future position will lie within the bounds of the dashed oval with 90% confidence, where the darker red of the heat map corresponds to a higher probability. It is worth reminding the reader that the term *position* as it is being used here corresponds to a point in the longitudinal and latitudinal center of the target vehicle. Figure 4.9 describes a situation in which the target vehicle continues in its own lane and moves past the ego vehicle. Note how the close the prediction is to the actual position, with the center of the target well inside the confidence interval.

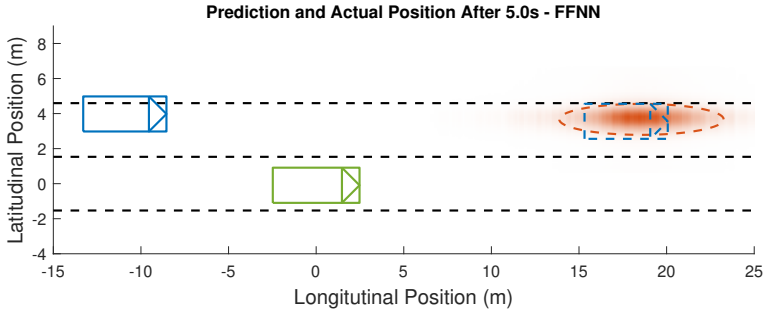


Figure 4.9: *Example of a situation where target moves past the ego vehicle and continues in its own lane. The ego vehicle is shown in green, the target is blue and the true future position in dashed blue. The ellipse bounding the heat map represents the 90% confidence interval and the heat map represents the probability distribution for the prediction.*

A different situation, one in which the target vehicle stays in its lane but does not move past the ego vehicle is presented in figure 4.10. Similar to the previous example, the actual target position falls comfortably within the 90% confidence interval.

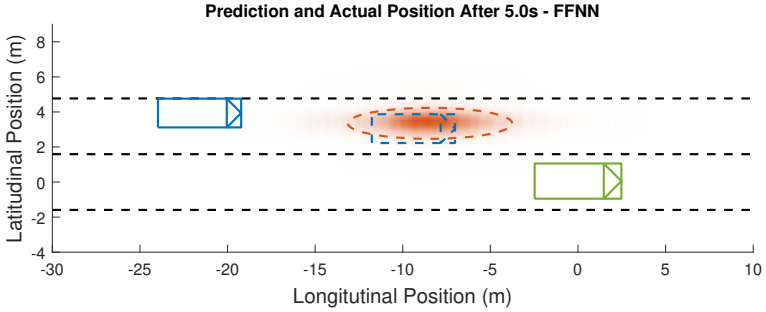


Figure 4.10: *Example of a situation where target stays behind the ego vehicle and does not change lanes. The ego vehicle is shown in green, the target is blue and the true future position in dashed blue. The ellipse bounding the heat map represents the 90% confidence interval and the heat map represents the probability distribution for the prediction.*

Prediction accuracy in situations with a lane change are shown in figures 4.11 and 4.12. Figure 4.11 illustrates a situation in which the target vehicle moves past the ego vehicle and changes into the ego vehicles lane. The FFNN model was able to predict the longitudinal position but completely missed the lane change. Figure 4.12 illustrates a similar situation in which the target vehicle changes into the ego vehicles lane, but maintains position behind the ego vehicle. Here the FFNN manages to predict both the lane change and the longitudinal position.

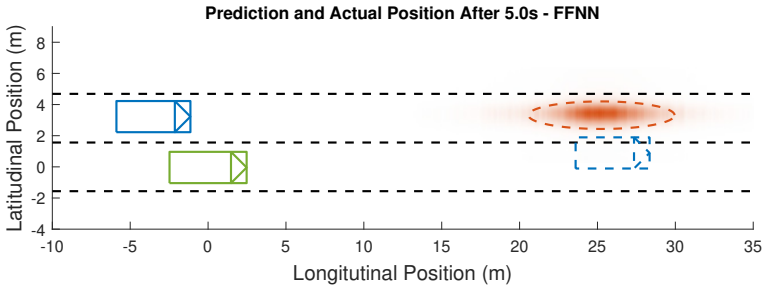


Figure 4.11: *Example of a situation where target moves past the ego vehicle before changing into the ego lane. The ego vehicle is shown in green, the target is blue and the true future position in dashed blue. The ellipse bounding the heat map represents the 90% confidence interval and the heat map represents the probability distribution for the prediction.*

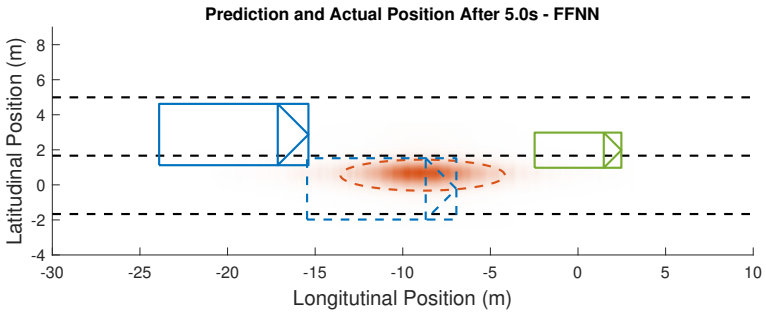


Figure 4.12: *Example of a situation where target stays behind the ego vehicle and moves into the ego lane. The ego vehicle is shown in green, the target is blue and the true future position in dashed blue. The ellipse bounding the heat map represents the 90% confidence interval and the heat map represents the probability distribution for the prediction.*

5 Discussion

The discussion in this section focuses on the results connection to the three main goals presented in the purpose, namely model performance for longitudinal and latitudinal position prediction, ranking of features by the different models and data-set influence on model performance.

5.1 Model Performance

Longitudinal

There are a number of interesting points for discussion raised by figure 4.1a. It is clear from the figure that for predictions over 2 seconds, the EQM models have a prediction error much greater than that of the other models. For the same prediction window, it can also be seen that difference in prediction error for the 5 remaining models is small and slow to diverge. For shorter prediction horizons, the FFNN deviates from the group with a much larger prediction error than the other methods.

Predicting vehicle movement is a mixture of kinematics and human behavior. The EQM models account only for the kinematic side of the problem which explains the larger prediction error for longer horizons as compared to the other models. Looking deeper, it is interesting how the EQM method performs better when removing the acceleration term. A reason for this may be that the EQM model in its original form assumes constant acceleration, which is an assumption not valid for longer time scales as people do not tend to keep acceleration constant for very long.

For the 5 remaining models in the figure, the differences in model complexity are not mirrored by the difference in prediction error. The differences between the methods are in fact small which, depending on the application, suggests that a simpler model may be more than adequate. This behavior also suggests that predictions of this accuracy may be the best one can hope for with the current feature set, regardless of the modeling method employed.

The deviation of the FFNN from the grouping mentioned above for shorter predictions, below 2 or 2.5 seconds, could be seen as another indication that model complexity does not always pay dividends. Keep in mind, however, that this model has been trained for all prediction horizons using the mean squared prediction error as a loss function. The prediction error is, in general, smaller for shorter prediction horizons. This, combined with the tendency of the mean squared error loss function to emphasize larger errors over small causes the training of the network to be more effected by the larger errors of longer predictions. As such, this behavior does not represent a limitation of the model but rather a limitation of the method used to train the model.

Latitudinal

Figure 4.1b has some similarities to its longitudinal counterpart, but also some clear differences. The similarities are found in the performance of the EQM models in relationship to the other methods. Different here, however, is the fact that the STD develops with a logarithmic, rather than exponential, tendency. Furthermore, the performance of the PCA+FLDA method exhibits strange behavior for prediction horizons above 3 seconds.

EQM models shown here differ from their longitudinal counterparts in that here they assume constant velocity respective constant position. In both cases, however, longitudinal and latitudinal, the reduction in EQM model complexity by one step improves the models prediction performance. The longitudinal explanation for this can be extended to the latitudinal case by assuming that constant lateral movement is not valid for time scales of this length. Given the constraints placed on vehicle movement by the rules and geometry of the road, there is a limitation for how much a vehicle can move in the latitudinal direction. Normal movements involve either smaller corrections or lane changes which typically take place in time scales shorter than 5 seconds.

Excluding non-physical predictions, the limitations placed on vehicle movement by the road also bound the size of the prediction error. In other words, it is impossible for an outlier to have moved a distance greater than the width of the road. This should, in theory, give a horizontal asymptotic behavior were the error approaches an upper limit when the prediction horizon tends towards longer prediction horizons. As seen in the figure though, STD curves increase almost linearly for prediction horizons over 2.5 seconds.

The strange behavior of the PCA+FLDA model for the latitudinal case has not been able to be explained. It is thought that this behavior might have to do with the method being more suitable for classification, rather than regression. The behavior may also be related to the Fisher epsilon parameter or the reduction of the

feature space down to a single dimension. This implies that the model was unable to map the features to a space with a linearly separable output.

5.2 Feature Ranking

An almost infinite number of possible features exist which could be included in the model. While it is not realistic to include everything, it is also the case that not all features will be of relevance. Irrelevant features with no meaning to contribute function as a source of noise, as well as create a larger search space and increase model complexity. This, in turn, makes the model more difficult to train and provides the model with more opportunities to overfit, clearly seen in figure 4.6. Analysis of feature influence can also provide insight into the inner workings of model, helping to establish trust in the models predictions. Proper feature selection is therefore essential.

The results in section 4.2 show that, in general, the longitudinal models provide similar rankings while the latitudinal models tend to rank features differently.

Longitudinal

For longitudinal prediction, it is common to all models that ego and target vehicle velocities are important, where target velocity tends to affect the outcome positively and ego velocity the outcome negatively. The same thing is the case with target and ego accelerations, even if they have a lower influence. This behavior is kinematically intuitive and inspires trust in the models.

Also common to all models is the appearance of the velocity or acceleration features from the ego lead and target lead vehicles. These feature hold information about the traffic situation ahead of the target vehicle which is likely to influence its behavior. For example, if the target lead vehicle starts to decelerate, it is likely that the target vehicle will also start to decelerate with a delay corresponding to the human reaction time.

Interestingly, the target heading feature also shows up for all methods. The question is, what does the heading of the target vehicle, defined as the angle between the vehicle and the direction of its lane, have to do with its longitudinal position? One thought is that the heading feature encompasses elements of target vehicle intent. If the model can differentiate between, for example, intent to perform a cut-in or intent to take an off-ramp, it would make sense that it can use this information to help with the position prediction. This is because each of these behaviors comes with its own set of constraints, explored in the discussion surrounding position prediction in the previous section.

Once clear difference between the feature rankings, on the other hand, is the importance of the target x position for the FFNN model, see figure 4.2. The reason behind this is that the FFNN model was trained to predict a position while the other models were trained to predict a change in position. The end result is essentially the same, except that the target x position is included in the output feature.

The FFNN model also ranks the width and length features of the lead vehicles, both ego lead and target lead, higher than the other models. It is conceivable that the network is able to use these features to distinguish between different vehicle types with different typical driving characteristics. In other words, the network may be using these features to distinguish between cars and trucks.

Latitudinal

The feature rankings for the latitudinal models are difficult to interpret as there is no consistency between the feature rankings. For example, LR favors velocities and positions of surrounding vehicles, figure 4.5b, while PCA-FLDA favors curvatures and headings, figure 4.4b. This inconsistency in the feature rankings is likely due to the poor fit of the different models on the latitudinal data-set.

Looking at the feature ranking from FFNN, figure 4.3a, it is clear that the ego blinker, both left and right, has a sizable correlation with the models latitudinal prediction. It could be that the ego vehicle is influencing the targets behavior via use of its blinkers. The influence of the left blinker is intuitive in situations where target makes room for a lane change by also changing lane. The ego use of the right blinker may partly be explained as a result of the specific distribution of blinker events possibly dominating in the data-set. For example, it could be that the ego vehicle mostly uses the blinker in merging situations where other vehicles tend to change lanes too.

5.3 Data Influence on Model Performance

In figure 4.9 and 4.10 reasonably accurate predictions were made while in figure 4.11 only the longitudinal prediction is correct. These figures are fairly representative for the whole validation set; the models have difficulty in predicting lane changes. This can be explained, in part, by the fact that this lane change situation represents only about 2% of the samples in the data-set. In figure 3.4b it is clear that in most of the data the target vehicle does not change lateral position more than one meter, and in just a few cases the change is larger than two meters. Compare this to the mean lane width of 3.3 m. To better model lane changes more data of lane changes is needed.

Other aspects of the data-set also have a pronounced affect on model performance. In figure 4.7a, model performance is correlated to the ego velocity and compared to the distribution of these parameters in the data set. It is clear that the model performs best when trained and evaluated on situations that are well represented by the data. The same is the case for the velocity difference between the target vehicle and ego vehicle. In figure 4.8a and 4.8b it is clear that the predictions are more stable where the data is well represented. Some methods perform very well on some points outside the main distribution but the question is if that is more luck than anything else.

5.4 Considerations for Model Selection

There are pros and cons for all models. Neither PCA-FLDA or LGP has reached their full potential with this problem. PCA-FLDA can gain from classifying the output data as different outcomes as cut in, passing etc. It is a method that has been widely used in areas as face recognition and works well when the input has a well defined structure and the output is divided into well defined categories.

LGP is an interesting method that generates equations, but has difficulties navigating the large search space of a many features problem like this. The training of the LGP is the most time consuming of the tested methods and because of its stochastic nature each training results in a different outcome. This makes it difficult to evaluate different setups. Interestingly, the training process initializes with random solutions and after relatively few generations produces the equations of motion before developing into something more complex. The method is flexible in the sense that the user decides what kind of instructions or operations the algorithm will use. But at the same time it can be inflexible, training the LGP for all prediction horizons crippled the algorithm with a vast search space and enormous number of samples. This caused the LGP training to become so slow as to make it effectively unusable.

FFNN is the best performing model over all. It is also the most flexible with the inputs. The model can handle a larger number of features better than the other models and can be trained for varying prediction horizons. The training is pretty computationally expensive, but does not take as much time as it does for the LGP. A potential downside of this model is that FFNNs are seen as black boxes. This is because it is difficult to understand what happens inside them. For applications where safety is a high priority, this could represent a hurdle as there is no way to interpret what the model will do in all situations.

LGP, LR and PR create models which are more transparent than the FFNN, at least as long as the equations and polynomials are kept short. But when the LGP produces in a long and complex equation, is it still transparent? Is there a point of complexity where an equation becomes as much of a black box as an FFNN? And how transparent is a polynomial really when it has over a thousand coefficients?

Both the linear and polynomial regression performed better than expected. Some reservations should be made that the used data is from highway driving. If more of the data-set represented situations with a higher percentage of human behavior versus kinematic influence, such as traffic jams or city driving, would these methods still perform as well? Or would the FFNN gain a bigger advantage?

Contrary to what one might think reading the results, the EQM models are not worthless. Depending on the requirements, they might, in fact, be the best choice as they do not need any training while offer a high degree of transparency and low complexity. For shorter prediction horizons they are even able to predict at least as well, and in some cases better, than the other more complicated modeling methods. It is this balance between performance, complexity and transparency that much be matched to the requirements of the specific model use case.

6 Conclusions

Based on the investigations made in this thesis, there are the following four main conclusions. Firstly, Modeling methods exist which offer better prediction performance than the equations of motion. This is not surprising as the traffic situation is dynamic and therefore assuming steady state becomes an increasingly poor approximation over time, see figure 4.1. As one can see in chapter 4.2 many of the more complex methods include the equations of motion but modify them with other features to create a model which is more sensitive to the dynamics of the traffic situation.

The second main conclusion is that model complexity does not predict model performance. This can be shown by the fact that, as discussed in chapter 5.1, the differences in model performance are quite small while the differences in model complexity are quite large. A model based on linear regression is much simpler than a model based on a FFNN but as one can see from figure 4.1, the complexity increase does not result in a performance increase of the same magnitude.

Thirdly, more input features do not imply better model performance. Chapter 5.2 discusses how irrelevant features can contribute to noise and model complexity which in turn can result in lower model performance. This can be seen clearly in figure 4.6.It is also clear that some modeling methods are better at handling noise than others, implying that the optimal number of input features may depend on the method being used.

Lastly, data driven modeling can be limited by the data set. As discussed in chapter 5.3, a model can only reliably predict situations that it recognizes from the dataset it is trained on. This can be seen in figures 4.7 and 4.8 which show that the prediction error is lower in regions with high representation in the training dataset.

7 Future Work

We believe that it is possible to get a little better results by working on the validation method and use cross validation, at least for FFNN. It would be interesting to explore how this problem is handled by other types of network architectures as recurrent neural networks, after all it is time series data we work with.

It would also be interesting to include and analyze more features as ego steering angle, or road information as curvature or tilting.

Some areas may have use of longer prediction horizons than the five seconds we have used. How the accuracy depends on the horizon can be investigated further, maybe to find an absolute limit.

Next big step though is to extend a model to more traffic constellations, and evaluate with data that is less processed and possible to do live in the car.

References

[1] *Teknikbarometern 2016 - whitepaper - Robotgräsklippare*. Tech. rep. HUI Research, June 2016. URL: http://www.hui.se/BinaryLoader.axd?OwnerID=474ff6a0-530d-465c-bc92-4b288a666c94&OwnerType=0&PropertyName=EmbeddedFile_986cd959-482f-47cb-9995-c3b35369a041&FileName=Teknikbarometern_Whitepaper_201606.pdf&Attachment=True.

[2] G. Prodhan. *BMW says self-driving car to be Level 5 capable by 2021*. http://www.autonews.com/article/20170316/MOBILITY/170319877/bmw-says-self-driving-car-to-be-level-5-capable-by-2021. (Accessed on 06/12/2018). Mar. 2017.

[3] M. McFarland. *BMW promises fully driverless cars by 2021*. http://money.cnn.com/2016/07/01/technology/bmw-intel-mobileye/. (Accessed on 06/12/2018). July 2016.

[4] T. Halleck. *Google Inc. Says Self-Driving Car Will Be Ready By 2020*. http://www.ibtimes.com/google-inc-says-self-driving-car-will-be-ready-2020-1784150. (Accessed on 06/12/2018). Jan. 2015.

[5] Ford. *Ford Targets Fully Autonomous Vehicle for Ride Sharing in 2021; Invests in New Tech Companies, Doubles Silicon Valley Team — Ford Media Center*. https://media.ford.com/content/fordmedia/fna/us/en/news/2016/08/16/ford-targets-fully-autonomous-vehicle-for-ride-sharing-in-2021.html. (Accessed on 06/12/2018). Aug. 2016.

[6] M. Martínez-Díaz and F. Soriguera. Autonomous vehicles: theoretical and practical challenges. *Transportation Research Procedia* **33** (2018). XIII Conference on Transport Engineering, CIT2018, pp. 275–282. ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2018.10.103. URL: http://www.sciencedirect.com/science/article/pii/S2352146518302606.

[7] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to Linear Regression Analysis*. Hoboken, NJ: Wiley, 2012. ISBN: 978-0-470-54281-1.

[8] H. Kavuncuoglu et al. Prediction of the antimicrobial activity of walnut (Juglans regia L.) kernel aqueous extracts using artificial neural network and multiple linear regression. *Journal of Microbiological Methods* **148** (2018), pp. 78–86. ISSN: 0167-7012. DOI: https://doi.org/10.1016/j.mimet.2018.04.003. URL: http://www.sciencedirect.com/science/article/pii/S0167701218300897.

[9] S. Azadi and A. Karimi-Jashni. Verifying the performance of artificial neural network and multiple linear regression in predicting the mean seasonal municipal solid waste generation rate: A case study of Fars province, Iran. *Waste Management* **48** (2016), pp. 14–23. ISSN: 0956-053X. DOI: https://doi.org/10.1016/j.wasman.2015.09.034. URL: http://www.sciencedirect.com/science/article/pii/S0956053X15301434.

[10] Y. Wang et al. Artificial neural networks for infectious diarrhea prediction using meteorological factors in Shanghai (China). *Applied Soft Computing* **35** (2015), pp. 280–290. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2015.05.047. URL: http://www.sciencedirect.com/science/article/pii/S1568494615003968.

[11] J. Schlechtriemen et al. "A probabilistic long term prediction approach for highway scenarios". *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Oct. 2014. DOI: 10.1109/itsc.2014.6957776. URL: https://doi.org/10.1109/itsc.2014.6957776.

[12] X. Shen and Z. C. He. "Analysis of Vehicular Behavior at Bottlenecks Considering Lateral Separation". *Proceedings of the Second International Conference on Intelligent Transportation*. Springer Singapore, Nov. 2016, pp. 169–185. DOI: 10.1007/978-981-10-2398-9_16. URL: https://doi.org/10.1007/978-981-10-2398-9_16.

[13] Y. Ren et al. Influential factors of red-light running at signalized intersection and prediction using a rare events logistic regression model. *Accident Analysis & Prevention* **95** (Oct. 2016), pp. 266–273. DOI: 10.1016/j.aap.2016.07.017. URL: https://doi.org/10.1016/j.aap.2016.07.017.

[14] Y. Hou, P. Edara, and C. Sun. Modeling Mandatory Lane Changing Using Bayes Classifier and Decision Trees. *IEEE Transactions on Intelligent Transportation Systems* **15**.2 (Apr. 2014), pp. 647–655. DOI: 10.1109/tits.2013.2285337. URL: https://doi.org/10.1109/tits.2013.2285337.

[15] E.-g. Wang et al. Modeling the Various Merging Behaviors at Expressway On-Ramp Bottlenecks Using Support Vector Machine Models. *Transportation Research Procedia* **25** (2017), pp. 1327–1341. DOI: 10.1016/j.trpro.2017.05.157. URL: https://doi.org/10.1016/j.trpro.2017.05.157.

[16] R. Graf et al. "A learning concept for behavior prediction in traffic situations". *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, June 2013. DOI: 10.1109/ivs.2013.6629544. URL: https://doi.org/10.1109/ivs.2013.6629544.

[17] H. Woo et al. Lane-Change Detection Based on Vehicle-Trajectory Prediction. *IEEE Robotics and Automation Letters* **2**.2 (Apr. 2017), pp. 1109–1116. DOI: 10.1109/lra.2017.2660543. URL: https://doi.org/10.1109/lra.2017.2660543.

[18] P. Kumar et al. "Learning-based approach for online lane change intention prediction". *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, June 2013. DOI: 10.1109/ivs.2013.6629564. URL: https://doi.org/10.1109/ivs.2013.6629564.

[19] J. Tang et al. Lane-changes prediction based on adaptive fuzzy neural network. *Expert Systems with Applications* **91** (Jan. 2018), pp. 452–463. DOI: 10.1016/j.eswa.2017.09.025. URL: https://doi.org/10.1016/j.eswa.2017.09.025.

[20] J. Schlechtriemen et al. "A lane change detection approach using feature ranking with maximized predictive power". *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, June 2014. DOI: 10.1109/ivs.2014.6856491. URL: https://doi.org/10.1109/ivs.2014.6856491.

[21] J. Schlechtriemen, K. P. Wabersich, and K.-D. Kuhnert. "Wiggling through complex traffic: Planning trajectories constrained by predictions". *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, June 2016. DOI: 10.1109/ivs.2016.7535557. URL: https://doi.org/10.1109/ivs.2016.7535557.

[22] M. Natrella. NIST/SEMATECH e-handbook of statistical methods (2010).

[23] M. Wahde. *Biologically inspired optimization methods: an introduction*. WIT press, 2008.

[24] M. F. Brameier and W. Banzhaf. *Linear genetic programming*. Springer Science & Business Media, 2007.

[25] I. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[26] A. Hervé and W. L. J. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics* **2**.4 (2010), pp. 433–459. DOI: 10.1002/wics.101. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.101. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101.

[27] M. Welling. Fisher linear discriminant analysis. *Department of Computer Science, University of Toronto* **3**.1 (2005).

[28] C. Li and B. Wang. *Fisher Linear Discriminant Analysis*. 2017.

[29] M. A. Nielsen. *Neural networks and deep learning*. Determination Press, 2015.

[30] H. B. Demuth et al. *Neural network design*. Martin Hagan, 2014.

[31] J. Heaton. *Introduction to neural networks with Java*. Heaton Research, Inc., 2008.

[32] G.-B. Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks* **14**.2 (2003), pp. 274–281.

[33] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should i trust you?: Explaining the predictions of any classifier". *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 1135–1144.