Learning Behavior Models for Interpreting and Predicting Traffic Situations

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Tobias Gindele

aus Stuttgart

Tag der mündlichen Prüfung:	30. Juli 2014
Erster Gutachter:	Prof. Dr.-Ing. Rüdiger Dillmann
Zweiter Gutachter:	Prof. Dr.-Ing. Alois Knoll

Zusammenfassung

In der vorliegenden Arbeit wird die Vorhersage von Verkehrssituationen unter Verwendung von Bayes'schen Schätzmethoden und maschinellen Lernverfahren untersucht. Die Fähigkeit, Situationen und das Verhalten von Verkehrsteilnehmern richtig einzuschätzen und mögliche Situationsverläufe zu antizipieren, stellt eine essentielle Voraussetzung für eine Vielzahl von Anwendungen in der Verkehrsdomäne dar. Entscheidend ist diese Fähigkeit für die automatische Handlungsplanung, Regelung und Risikoeinschätzung und damit grundlegend für selbstfahrende Fahrzeuge und fortschrittliche Fahrerassistenzsysteme.

Die Herausforderungen dabei sind vielfältig. Die Verhaltensweisen von Verkehrsteilnehmern sind stark gekoppelt und sehr situationsabhängig. Die Ursachen, die zu bestimmten Verhaltensweisen führen, sind dabei nur unzureichend bekannt und schwierig modellierbar. Hinzu kommt, dass die Umwelt nur partiell und fehlerbehaftet wahrnehmbar ist. Wichtige Aspekte, wie die Gedanken von Verkehrsteilnehmern und damit ihre Ziele und Pläne, lassen sich nicht direkt messen. Ebenso sind die Aktionen von Verkehrsteilnehmern nur unsicher vorhersagbar.

Zur Lösung dieser Herausforderungen wird in dieser Arbeit ein Ansatz präsentiert, der die Entwicklung von Verkehrssituationen als stochastischen Prozess formuliert. Dabei werden die Entscheidungsprozesse von Verkehrsteilnehmern nachgebildet, um ihre wechselseitige Beeinflussung zu modellieren. Dadurch werden genaue langfristige Vorhersagen möglich. Den Kern bilden Verhaltensmodelle, welche die Zusammenhänge zwischen Situationen und Zielen, Plänen und Aktionen von Verkehrsteilnehmern beschreiben. Da diese Zusammenhänge nur schwierig manuell modellierbar sind, wird ein datengetriebener Ansatz verfolgt. Hierzu wird in dieser Arbeit ein Lernverfahren vorgestellt, das es unter Verwendung von domänenspezifischem Wissen ermöglicht, die Verhaltensmodelle aus Verkehrsbeobachtungen zu lernen.

Zur Steigerung der Generalisierungsfähigkeit dieses Lernansatzes werden neuartige, allgemeine Lernverfahren für Entscheidungsbäume präsentiert.

Der präsentierte Ansatz zeichnet sich dadurch aus, dass durch die Konzeption des Schätzprozesses, zusammen mit der datengetriebenen Modellierung, Vorhersagen in unterschiedlichsten Verkehrsszenarien ohne spezielle Anpassungen ermöglicht werden. Der gesamte Ansatz wird in einer Vielzahl von Verkehrsszenarien, wie beispielsweise an Kreuzungen und Kreisverkehren, mit mehreren Verkehrsteilnehmern evaluiert.

Abstract

In this thesis, we study Bayesian state estimation and machine learning methods for predicting traffic situations. The cognitive ability to assess situations and behaviors of traffic participants, and to anticipate possible developments is an essential requirement for several applications in the traffic domain, especially for self-driving cars and advanced driver assistance systems.

There are several aspects that make the prediction of traffic situations difficult. The causes of specific behaviors of traffic participants are only insufficiently known and difficult to model. The behaviors are highly coupled and strongly depend on the situational context. This is aggravated by the fact that the environment is only partially observable. Important features such as the thoughts of traffic participants including their goals and plans are not directly measurable and can only be estimated. Due to the partial observability and the stochastic nature of the environment the actions of traffic participants and their outcomes can only be predicted with uncertainty.

In this work, we present an approach that models the development of traffic situations as a stochastic process. Thereby, decision making of traffic participants is simulated to model their mutual influences. This way, we achieve accurate predictions over longer time periods. The core of this approach are behavior models which describe relations between situations, goals, plans, and actions of traffic participants. Since these relations are difficult to model manually, we use a data-driven approach. We present a method for learning behavior models from unlabeled traffic observations. As a contribution to machine learning in general, we develop novel learning methods for decision trees with improved generalization and efficiency properties.

The presented approach distinguishes itself through accurate predictions in a large variety of situation without special adaptations. We demonstrate this ability for a multitude of traffic scenarios including intersections and roundabouts with several traffic participants.

Contents

1 Introduction

Fully autonomous cars are on the verge of coming into existence and becoming a part of our daily lives. The dream of vehicles transporting us safely and without the constant need of our attention is old and appears in many fictional stories. With the possibility of providing a cheap and efficient solution to personal transportation, they have the power to transform our society.

Since the first pioneering work of (Dickmanns and Zapp, 1988) in the 1980s, significant advances have been made on all technology frontiers relevant to autonomous driving. The research in the field of autonomous driving and advanced driver assistance systems (ADAS) has already led to assistance systems with series maturity, e.g., lane departure warning, traffic sign recognition and automatic parking systems. The first prototypes of autonomous cars are being developed and tested by research organizations and major companies. However, some fundamental open question have yet to be resolved until robotic cars will populate our roads and supersede human drivers in terms of safe and foresightful driving. The research areas concerned range from perception and situation understanding to decision making and control.

An autonomous car faces the challenge to continuously act in a highly dynamic environment, which can only be perceived partially. Observations are not only noisy and affected by sensor range limitations and occlusions, some relevant aspects cannot be measured at all such as the intentions of other road users. Besides the capabilities to perceive its environment and conduct actions, the system must be able to fuse new information with its present knowledge in an ongoing process and derive decisions on that basis. An important prerequisite to anticipating situation developments is the capability to assess other drivers and their intentions. These cannot be directly measured and have to be inferred from observations. Since the behavior of traffic participants is highly coupled, realistic predictions can only be made with an understanding of how traffic participants influence each other. Human drivers acquire the ability to interpret traffic situations and to predict the likely developments through a prolonged learning process. Through their experiences, they develop a deeper understanding of the coherences that underlie the behavior of traffic participants.

The goal of this thesis is to contribute to the understanding and the solution of the state estimation and prediction problem encountered by autonomous systems navigating in traffic. By modeling a learning process, we enable a system to learn from observed motion patterns of traffic participants and putting them into relation. The system acquires the ability to predict situation developments in new and similar traffic constellations by generalizing from the observed traffic.

We present a hierarchical Bayesian model that resembles the decision making of traffic participants and their interactions in order to make accurate predictions about future situation developments. By taking their perspectives, it is possible to draw conclusions about their goals, plans and actions and reason about their mutual interactions. We take a probabilistic approach to handle the uncertainties inherent in this domain in a mathematically sound way and to be able to provide uncertainty estimates for the predictions.

The main challenge of this approach lies in the conception of the complex predictive policy models representing the decision making of the traffic participants. We solve this problem with machine learning methods that identify the behavior patterns in traffic observations in order to derive the models. We show that this data-driven approach has several advantages over approaches that mainly rely on manual modeling, mostly in terms of scalability and generalization.

In the course of this work, we derive generalizations and extensions to the decision tree learning methodology as a contribution to the general field of machine learning. The learning methods show an improved performance compared to state-of-the-art decision tree induction methods. Their incremental nature makes them applicable for big data problems and online settings even outside the robotics domain.

1.1 Problem Statement

Given a series of incoming noisy measurements of traffic participants' states, one is interested in estimating the current state of the environment and especially in anticipating future developments. The measured data consists of observable properties, such as position, orientation and velocity of traffic participants. Properties that are not directly observable, such as goals and plans of traffic participants, have to be inferred from their behavior and situational context over time.

To ensure realistic predictions, the required models are supposed to be learned from traffic observations and background knowledge without requiring manual labeling. The inherent uncertainties have to be considered in the whole process.

This poses an interesting and challenging problem due to a highly dynamic and only partially observable environment and the complex and mostly unknown system dynamics. Additional difficulties are induced by the manifold of situations that can arise, which makes generalization an important factor. In particular, the proposed method tries to answer the following questions:

- What is the current state of a traffic situation and how will it probably look like in a few seconds?
- What are the goals and plans of the traffic participants?
- How do traffic participants react in specific situations and how do they influence each other?
- How can models be learned from traffic observations, which allow answering of the aforementioned questions?
- How can background knowledge be used to leverage generalization?

We try to answer these questions based on measurements from traffic situations that can be acquired with state-of-the-art sensors, e.g., stereo cameras or lidar sensors. It is also assumed to have map information of sufficient accuracy.

1.2 Thesis Statement

The thesis statement is:

"The probabilities of possible future developments of traffic situations can be estimated based on learned models that resemble the decision making of traffic participants and consider their interactions."

We support the statement by presenting a novel learning based approach to prediction making in traffic environments and by evaluating its performance in comparison to other approaches in a variety of traffic scenarios.

1.3 Concept Overview

Throughout this thesis, we develop a Bayesian model that interprets the evolution of traffic situations as a stochastic process. Embedded in application systems, such as a motion planner or a risk assessment assistant, the model can provide predictions together with probability estimates as well as estimates of the current situation. Through observations of the environment, the inference process is fed with an ongoing stream of measurements. It fuses the incoming information with the systems prior knowledge to yield posterior distributions over the possible situations together with updated anticipations. In the course of the updating procedure, some hypotheses are ruled out and others become more likely depending on the supplied evidence. The presented approach makes no assumptions about where the measurements of the environment originate from. This renders it possible to use it in a self-driving car with on-board sensors as well as in a traffic surveillance system using stationary cameras. It is also possible to utilize a mix of sensing systems using on-board sensors in combination with communication to receive information about the environment from other vehicles and infrastructure.

Figure 1.1 shows an overview of the concept with an increasing level of detail. On the highest level one sees the cyclic interaction between the real world and possible application systems, which make use of the Bayesian model. Based on measurements of the environment and background knowledge like map data, the model provides state estimations and predictions. The model itself is represented by a dynamic Bayesian network and is sketched on the lowest level.

The Bayesian model builds upon the assumption that the observable behavior of traffic participants is the result of a rational thinking process, i.e., the actions they choose in a specific situation are determined by long-term goals they are trying to reach. The idea is that if we knew the goals of other traffic participants, we would be able make accurate predictions on how the whole situation will probably develop. Since the goals are not directly measurable, they have to be estimated from what can be observed. By taking the perspective of other traffic participants, we can reason about their intentions. The probable causes of their actions can be concluded by matching their observed behavior to the behavior one would expect when being in their position.

The Bayesian model enables this kind of reasoning by resembling the dependencies between the goals of traffic participants, the routes they plan to reach these goals and the situation-specific actions they choose. The models that predict their decisions on the different levels of abstraction are called *policy models*. We explicitly model the situation-dependency of the decision making processes with the purpose of doing justice to the mutual influence traffic participants have on the decisions of others through their behavior.

We develop a learning procedure that can capture the sophisticated patterns in behavior, which is additionally complicated by the fact that the dependencies of interest cannot be observed directly. A key role in our approach is the combination of machine learning methods with domain-specific knowledge and statistical modeling in order to derive predictive models that can generalize to a wide variety of situations and the potential to scale to the full complexity of traffic.

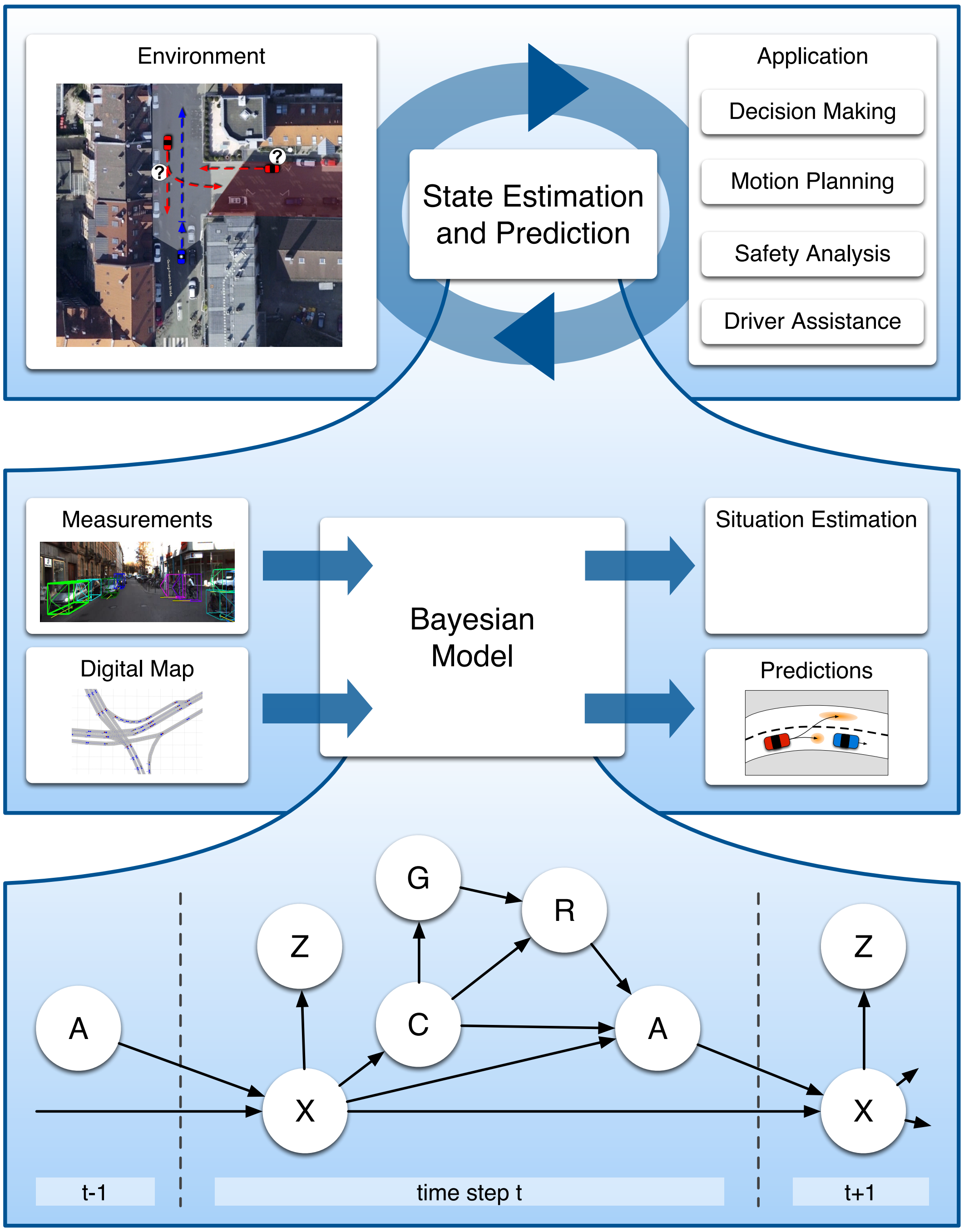


Figure 1.1: Conceptual overview.

1.4 Contributions

With this work, we contribute to robotics and machine learning research by developing a solution to the problem of state estimation and prediction in the context of traffic prediction together with novel machine learning methods. By taking a data-driven approach, we advance the understanding of machine learning methods in real-world settings.

We propose a novel hierarchical Bayesian model that resembles the decision making of traffic participants on multiple levels of abstraction. Compared to previous approaches, goals, plans and actions of traffic participants are simultaneously estimated in a unified probabilistic framework under consideration of context-dependent mutual influences. In contrast to existing approaches, map knowledge is used as a source of information for the prediction and learning process and not to constrain the representation by restricting traffic participants to be representable only on lanes.

In contrast to the state-of-the-art, the policy models of the Bayesian approach are derived by a combination of machine learning methods and domain-specific knowledge to capture the complex non-linear relationships between situations and decisions and to maximize generalization. We present efficient learning algorithms based on Monte Carlo expectation maximization to learn under partial observability. Input-dependent noise is considered in the learning procedures to satisfy the heteroscedasticity present in the data. Compared to the state-of-the-art, the models are learned in a non-parametric way with no need for manual labeling. This makes the approach scalable and the incorporation of new experiences easy.

As a general contribution to the field of machine learning, we derive a new class of decision trees called generalized decision trees, which generalize common tree models such as classic decision trees, hierarchical mixtures of experts and fuzzy decision trees. We present a gradient-based learning algorithm for this new class together with a new learning algorithm for classic decision trees based on continuation methods. These new learning methods achieve better generalization in evaluation than state-of-the-art decision tree induction methods.

The main contributions can be summarized as follows:

1. A novel hierarchical Bayesian model for estimating and predicting traffic situations under consideration of context-dependent mutual influences (Chapter 3).

2. Scalable learning algorithms for non-parametric learning of heteroscedastic policy models under partial observability (Chapter 4).

3. Improved learning methods for decision trees and generalized decision trees (Chapter 5).

1.5 Applications

We highlight four central areas of applications where anticipation of traffic situations and learning methods are of fundamental importance.

Autonomous driving The ultimate goal of autonomous driving is the complete replacement of the human driver by a machine that can perfectly navigate in every possible traffic situation. The potential benefits are compelling: less accidents (Petridou and Moustaki, 2000; McKenna, 2010), improved energy efficiency and comfort, and a much better exploitation of existing infrastructure (Fagnant and Kockelman, 2013) to name just a few. By removing the need for a human driver, people can use the travel time for their own purposes, e.g., for enjoyment, sleeping or working. Additionally, it will make personal transportation accessible to people which are not allowed or unable to drive a car, such as kids, the elderly or disabled people. While researchers and companies have made substantial progress in the field of autonomous driving over the last years and already demonstrated basic autonomous driving capabilities in some scenarios (FZI Research Center for Information Technology at Karlsruhe Institute of Technology, 2013; AutoNOMOS Labs, 2011; Nothdurft et al., 2011; Guizzo, 2013; Viz-Lab, 2013), a self driving car that drives as versatile and foresightful as a human and can cope with all the unexpected situations is still an open challenge (Knight, 2013; Shepardson, 2013). What seems to be certain is that once self-driving cars become reality and affordable, they will deeply transform our society and bring up new forms of transportation (Fagnant and Kockelman, 2013).

Advanced driver assistance systems While fully autonomously driving cars might be available in the future at some point, there are a lot of intermediate steps at which a car system can make a drive more pleasant, less stressful and safer by providing assistance in certain situations. While assistance systems like adaptive cruise control (Winner et al., 2009), lane departure warning (Batavia, 1999) and emergency braking assistants (Yi et al., 2002) are already available to the consumer today, the next generation of advanced driver assistance systems will heavily rely on functionalities that enable the assistant system to interpret situations and make accurate predictions.

Traffic surveillance systems Another field of application that can benefit from the findings in this work is traffic surveillance (Hu et al., 2004). With perception data from cameras mounted at intersections or highways, a surveillance system can automatically detect abnormal behavior of drivers, traffic violations or warn traffic participants via Car2X (Festag et al., 2008), if predictions indicate a critical situation about to emerge. It is also possible to reconstruct the most likely course of events after a traffic accident and to ascertain the causes that led to it.

Data mining The decision tree learning algorithms developed in this thesis are general regression methods and are not limited to the traffic domain. Their ability to handle large data sets and to learn from data streams incrementally makes them applicable in the area of data mining (Ikonomovska, 2012; Gama et al., 2003).

1.6 Document Outline

This thesis is structured as follows: Chapter 2 gives an overview of related work and discusses the pros and cons of the different approaches. Conclusions are drawn that motivate the presented approach. In Chapter 3, the developed Bayesian model and all its components are described in detail together with probabilistic inference techniques to make predictions. Subsequently, we present learning methods in Chapter 4, to derive the predictive models used to resemble the decision making of traffic participants from observations. In Chapter 5, we describe several improved learning methods for decision trees that were developed in the course of this thesis. Finally, Chapter 6 describes and assesses the experiments that have been carried out to evaluate the presented approach. We close with a summary and conclusion of the thesis in Chapter 7, where we also analyze the limitations and suggest directions for future research.

2 Related Work

In the last decade, the field of research of state estimation and prediction of traffic situations is receiving more and more attention from the research community due to the increasing interest in autonomous driving and advanced driver assistance systems. This has led to a number of novel approaches and developments. The main categories in which the approaches differ are their representation of traffic situations, the information sources they consider, the type of reasoning that is used and the methods for deriving models. In this chapter, we discuss the related work and draw conclusions for the requirements of a system that is able to realistically predict the developments of traffic situations.

2.1 Multi-Target Tracking

Methods for estimating the state of multiple dynamic objects from noisy measurements are subsumed by the area of multi-target tracking. The area has a long history in research and numerous methods have been proposed (Blackman and Popoli, 1999; Bar-Shalom, 2000). Classical approaches track the states of objects independently by instantiating a single object tracker, e.g., a Kalman filter, for each known object.

The main difficulty of multi-target tracking is the *data association* problem. It is concerned with the association of measurements and the individual objects. It is complicated by the fact that the number of measurements does not necessarily match the number of known objects. Techniques for handling these cases are summarized as *track management*. They include the decision whether a measurement is identified to stem from a new object or one that is already known. Track management methods also have to decide whether multiple measurements are assumed to be caused by a single object or when measurements are treated as outliers. Existing solutions for performing data association can be found in (Daum, 1996; Gauvrit et al., 1997; Cox, 1993; Chen et al., 2013).

The main limitation of an approach that independently tracks objects such as the classic approaches in multi-target tracking is that the complex interactions that are predominant in traffic situations cannot be considered. Since the reactions of road users on the actions of other road users determine the evolution of situations, it is crucial for an accurate prediction to take the mutual influences into account.

2.2 Dynamic Occupancy Grids

Traffic situations can be represented on different levels of abstraction. A research field that uses low-level representations for tracking and predicting the state of the environment are dynamic occupancy grids. Rather than describing traffic situations on an object level, occupancy grids use a grid representation which distinguishes only between cells that are occupied or not (Elfes, 1989). The cell states are estimated as independent random variables for complexity reasons. Static occupancy grids have been successfully uses in the area of simultaneous localization and mapping (SLAM) (Thrun et al., 2002) but are not able to cope with dynamic environments such as traffic.

Several extension have been proposed to enable dealing with moving objects and changing environments. A probabilistic formulation which also estimates the velocity of occupied cells called Bayesian occupancy filter (BOF) is presented in (Tay et al., 2008) and (Coue et al., 2006). In (Mekhnacha et al., 2008), the BOF framework is extended to establish a connection to the object level by clustering similar moving cells. The work in (Gindele et al., 2009) present several improvements to the BOF framework called BOFUM by using a physically more accurate transition model that also considers uncertainty in the velocity and incorporates map knowledge. In (Brechtel et al., 2010), an efficient approximate inference method for filtering in the BOFUM framework was presented. The work of (Brechtel et al., 2009) presents an interacting multiple-model extension to the BOFUM framework.

Extensions to incorporate information from various types of sensors such as stereo cameras and lidar sensors are presented by (Vatavu et al., 2011; Moras et al., 2011; Adarve et al., 2012)

The authors of (Kucner et al., 2013) present an extension to the BOF framework by learning position-specific cell transition models from observations. It is shown that this can improve the tracking accuracy. However, no ways are presented that allow generalizing the learned models to other road configurations.

Bayesian Occupancy Grids for dynamic environments are versatile tools for estimating the state of the environment in terms of free and occupied space. Their prediction accuracy is limited by the fact that they miss the notion of coherent objects and also by the independency assumption of cells. Thus, it is difficult to incorporate higher-level information such as traffic rules or object specific motion models. Some work addresses this issues by investigating hybrid approaches that combine the occupancy grid representation with an object-oriented representation (Laugier et al., 2011; Mekhnacha et al., 2008).

2.3 Vehicle Tracking

For tracking the motion of a single vehicle, numerous approaches have been proposed. Simple models predict the motion solely based on the kinematic and dynamic properties of a vehicle (Mitschke and Wallentowitz, 1972). A popular example is the single track model (Campion and Chung, 2008). Such models are sufficient for most tracking applications but are not eligible for accurate long-term predictions. This is because they do not consider context information and are solely based on the motion histories of a moving vehicle. Even in the simple scenario where a car follows a curvy road, the future trajectories cannot be anticipated correctly since nothing in the current motion state indicates the possible turn ahead.

More advanced approaches incorporate information about the road infrastructure into the motion models. This information either stems from digital map data or from a perception system that directly estimates the lanes from sensor data.

(Alin et al., 2012) present a Bayesian filter approach for tracking single vehicles. They utilize map data to extract probable attractor points a driver is heading for. Possible driving trajectories are derived by connecting the vehicle pose with these attractor points through spline functions. Experiments show that improved tracking performance is reached over a Bayesian filter with a kinematic model. In an extended version of the approach, a hidden Markov model (HMM) is added to estimate behaviors such as turning left or right over time (Alin et al., 2013).

Map data is also employed in the work of (Petrich et al., 2013) to improve the tracking quality of individual vehicles. They present a multi-hypothesis extended Kalman filter to track the dynamic state of a vehicle. By matching vehicle positions to lanes and using the center line of lanes as virtual evidence, the predictions derived with a bicycle model are corrected to follow the course of the lanes. Map data can significantly improve the prediction accuracy for single cars since drivers mainly drive along roads. However, if the map data constrains the representable motion patterns too much, it can cause inconsistencies. For example, if drivers deviate from driving along roads or if the map data is not accurate.

Another direction to improve the prediction of future driving trajectories is to consider only realistic trajectories instead of all possible trajectories. The authors (Hermes et al., 2009) present such an approach. They extract a set of driving trajectories from recorded driving data. For predicting future trajectories, the current trajectory of a car is matched against the trajectory set according to a similarity measure.

Another approach for deriving realistic driving trajectories is presented in (Yao et al., 2012). In this work, lane change maneuvers driven by humans are recorded and clustered into sets.

Data-driven approaches to trajectory prediction can significantly improve the accuracy of predicting human driving behavior. Nevertheless, the discussed approaches determine the predicted trajectories solely on the driving of a single vehicle and therefore do not consider context information.

2.4 Goal, Plan, and Behavior Recognition

The target of goal, plan and behavior recognition is to estimate the goals, plans and behaviors of an acting individual (agent) from a series of observations (Sukthankar et al., 2014). These are often combined in a hierarchical manner due to their strong interrelationships. Knowing one these aspects provides information about the others. Methods for solving these tasks were also investigated in other application domains such as human activity recognition and person tracking (Oliver et al., 2002; Choi and Savarese, 2012; Pellegrini et al., 2009).

2.4.1 Goal Recognition

Goal recognition in the traffic domain is concerned with estimating the goal(s) of a traffic participant. A goal is often defined in terms of a target location that a traffic participant is intending to reach.

The authors (Dagli et al., 2003; Dagli and Reichardt, 2002) present a hierarchical approach for behavior prediction based on the motivations of drivers. They conclude that in order to predict a drivers behavior, it is necessary to infer his situation-specific motivations and goals. A symbolic planner is used to generate possible plans for each traffic participant. These plans are are matched to the observed motion patterns. The approach is tested in a simulated highway scenario with the aim of improving adaptive cruise control (ACC) functionality. Due to complexity reasons, only interactions between nearest neighbors are considered in this approach. This limits the possible scenarios that can be assessed correctly.

For some common classes of situations such as driving at intersections or highway driving, specific solutions have been proposed that are tailored for the application domain. Predicting the behavior of road users at intersections is of special importance for ADAS since they exhibit a high risk for accidents due to the strong interactions between traffic participants.

In (Zhang and Rössler, 2009), an approach is presented which estimates the behavior of individual traffic participants at intersections and combines these predictions for risk assessment. The approach is structured hierarchically. In the first stage, the probabilities of possible paths through the intersection are derived from the lane topology of the intersection. In the second stage, the vehicle dynamics are estimated with Bayesian filtering using the path information. Possible conflicts between traffic participants are identified on the topological level. The final risk assessment is concluded by analyzing the potential conflicts via fuzzy rules.

The topological information of an intersection is also utilized in the work of (Lefèvre et al., 2011) in order to predict the lane on which a driver intends to exit the intersection. From the topological information a Bayesian network is derived that models the different pathways through the intersection. The estimates are updated by incorporating new measurements of the position and the turn signal state of a traffic participant. The concrete vehicle dynamics and the influence of other traffic participants are not considered in this approach.

The authors (von Eichhorn et al., 2013) take a different approach to predict the most likely maneuver of a single driver at an intersection. They model the drivers decision between the possible exits of an intersection as an optimal control problem with an unknown terminal state. By solving the constrained optimization problem numerically and comparing the expected costs of the possible choices, the most likely maneuver hypothesis is derived.

The discussed work in this paragraph focused on goal estimation. Reasoning about the goals of drivers can improve the anticipation of situation developments over longer periods of time. Some of the discussed approaches are conceptually constrained to intersection situations which limits their application in other traffic situations.

2.4.2 Plan Recognition

The goal of plan recognition is to estimate the plans of an acting individual from observations. While early work in this field only identified plans that are consistent with the observations (Kautz and Allen, 1986), newer approaches also estimate the probabilities of plans (Charniak and Goldman, 1993; Bauer, 1994) and update the estimates when new measurements become available (Pynadath and Wellman, 1995; Huber et al., 1994; Goldman et al., 1999). General hierarchical methods for solving plan recognition problems are abstract hidden Markov models (AHMMs) (Bui et al., 2002), hierarchical abstract machines (Parr and Russell, 1998) and probabilistic state-dependent grammars (Pynadath and Wellman, 2000). In the context of traffic scenarios, plans are often considered as possible routes to target locations. Knowing the possible plans and their probabilities provides valuable information about the possible realizations through behavior primitives or future trajectories.

In (Pynadath and Wellman, 1995), a general Bayesian framework for plan recognition is presented. The investigated application is traffic monitoring in highway scenarios. Context information is considered in the estimation process. However, the approach is not intended for online estimation.

In (Liebner et al., 2013), a method for estimating the distribution over possible routes of a driver for ADAS applications is presented. The set of possible routes is extracted from a digital map. The posterior probabilities of the individual routes are derived with a naive Bayes classifier. It combines attributes such as the velocity profile, the indicator signal state and the gaze direction of the driver.

The authors (Patterson et al., 2003) present a high-level approach for tracking the motion of a traveller in an urban environment. The model for predicting the most likely routes as well as the transportation mode is learned from GPS data using expectation maximization (EM). The tracking is realized with particle filters. The approach is not directly transferable to predict vehicle dynamics due to the high abstraction level of the representation.

Plan recognition can improve the anticipation of situation developments significantly. However, it is difficult to consider all the possible interactions between traffic participants in the plan estimation process due to the arising complexity.

2.4.3 Behavior Recognition

Behavior recognition is concerned with estimating behavior primitives from noisy observations. A behavior primitive can be, for instance, a lane change maneuver, an emergency braking or a lane following. Main directions are discriminative and generative approaches. Discriminative approaches aim to classify the most likely behavior while generative approaches model distributions over the set of possible behaviors (Doshi and Trivedi, 2011). The approaches differ in the type of classifier or graphical model they apply and the categorization of behavior primitives.

In (Aoude et al., 2011), two behavior recognition methods for application in intersection scenarios are presented and compared. One is based on a support vector machine (SVM) with Bayesian filtering and the other is based on hidden Markov models (HMMs).

The authors of (Morris et al., 2011) present an approach for detecting lane change maneuvers of a driver. They use a relevance vector machine to detect if a driver intends to initiate a lane change maneuver.

A multilayer perceptron-based approach to behavior recognition and prediction is presented in (Ortiz et al., 2011a). The training examples are derived from recorded vehicle data and are automatically labeled according to heuristics. The approach is evaluated in signaled intersection scenarios with the goal of predicting the driving and stopping behavior of a driver.

A generative approach to behavior recognition in highway scenarios is presented in (Kasper et al., 2011). Traffic situations are modeled with object-oriented Bayesian networks. The classification of lane change maneuvers is based on vehicle-lane and vehicle-vehicle relations.

Another generative approach based on conditional random fields for estimating driving behaviors at intersections is presented by (Tran and Firl, 2012).

A method for detecting lane change maneuvers in highway scenarios is investigated in (Tsogas et al., 2008). Dempster-Shafer theory is used to identify the maneuver type of a driving vehicle based on a relational description of the environment.

A hybrid approach that combines the output of a SVM with Bayesian filtering is presented in (Kumar et al., 2013). Based on ego-vehicle data and a lane tracker, the probabilities for lane change maneuvers are estimated.

Other methods have been applied to the task of behavior recognition such as probabilistic finite-state machines and fuzzy logic (Hulnhagen et al., 2010) and extreme learning machines (Demcenko et al., 2008).

Behavior recognition can be a useful component for predicting future situation developments. However, most discussed approaches do not make the connection between behavior primitives and distributions over the possible resulting driving trajectories. Grounding the symbolic level of behaviors on the continuous level of vehicle dynamics is important for predicting the quantitative properties of vehicles such as their position or heading. This is essential, for example, for the task of motion planning.

2.4.4 Situation Recognition

Situation recognition is concerned with identifying the type of situation a traffic participant is facing. Situation classes take the constellations of multiple traffic participants into account. Knowing the type of a situation allows making predictions of the traffic participants' behaviors. The abstraction level of the distinguished situations in the recognition process and their meaning depends on the type of application and varies in the literature. For instance, an ADAS may distinguish between critical and uncritical situations in a highway scenario. From a critical situation, it can predict that a driver will most likely brake or initiate an evasive maneuver in order to prevent an accident.

In (Meyer-Delius et al., 2008), a relational HMM is used to recognize different classes of traffic situations based on a semantic representation. The estimations of a Bayesian filter for tracking the vehicle dynamics serve as input to update the distribution of situation hypothesis on the semantic level. Experiments of simulated overtaking maneuvers are conducted where the situation classes *passing*, *aborted passing* and *follow* are recognized. Extensions to this approach are developed in (Meyer-Delius et al., 2009).

A generative approach for predicting vehicle motion that considers contextual information is presented in (Agamennoni et al., 2012). Based on feature functions that evaluate situational aspects, the types of situations are classified. Depending on the type of situation, the parameters of the motion models are conditioned. They present inference techniques for the probabilistic approach together with parameter estimation techniques for optimizing the recognition of context classes. The approach is evaluated for the task of tracking mining vehicles.

To recognize the class of situation a road user is facing, (Bonnin et al., 2012) propose using a decision tree. The hierarchy of situations and their discrimination functions are manually constructed. Each situation class is associated with a specific classifier that is used to predict the behavior of a traffic participant. Results of experiments at the entry of a highway are presented where the situation classes *Entrance Enter* and *Entrance Giveway* are distinguished.

The authors of (Ortiz et al., 2011b) present a prediction system on the level of behavior primitives. The situation types are classified with a manually constructed rule set. For each type of situation, a multilayer perceptron is trained on the basis of a feature-based situation representation in order to predict the next behavior primitive.

In the work of (Käfer et al., 2010), intersection situations are classified. A polynomial classifier is used to identify the types of situations with two cars. Depending on the situation class and the motion history of the cars, possible driving trajectories are predicted based on a set of recorded trajectories.

An approach for situation threat assessment is presented by (Eidehall and Petersson, 2008). They use a probabilistic driver model and lane-based tracking to predict probable future driving trajectories with Monte Carlo sampling. Threat measures are calculated on the set of probable future situations in order to identify critical situations.

A special case of situation recognition is investigated in (Batz et al., 2009). They classify situations types for groups of cooperative vehicles that communicate with an application to threat assessment.

Recognizing specific types of situations can improve the prediction process. The main challenge involved is defining a consistent set of situations and drawing the connections to probable resulting behaviors of traffic participants. While the results look promising for specific scenarios like intersections, it is difficult to find an exhaustive set of situations that covers all possibilities of traffic situations.

2.5 Urban Challenge

In 2007, the Defense Advanced Research Projects Agency (DARPA) initiated a competition named Urban Challenge to demonstrate and benchmark the state-of-the-art in autonomous driving (Buehler et al., 2009). In contrast to its predecessors, the Grand Challenge 2004 and 2005 (Buehler et al., 2007), the focus was on driving in an urban scenario rather than driving in unstructured environments like deserts. The competing self-driving cars had to drive a 96 km course fully autonomously. The main goals for the participating systems were to comply with traffic rules and to complete the course as fast as possible without incidents.

Most of the finalist teams used a state machine approach for interpreting situations and deriving behavior decisions (Urmson et al., 2008; Montemerlo et al., 2008; Bohren et al., 2008; Bacha et al., 2008; Miller et al., 2008; Kammel et al., 2008). An exception to this approach was presented by (Rauskolb et al., 2008), which used a voting based approach called DAMN for deriving driving decisions (Rosenblatt, 1997).

The Urban Challenge showed that fully autonomous driving is possible in simplified scenarios. The complexity of the encountered situations was low compared to inner city traffic situations with lots of interacting road users and other moving objects. Another aspect that simplified the driving task was the availability of accurate digital maps of the courses. Due to the manageable complexity of situations, manually modeled state machines proved as a successful method in the Urban Challenge. However, scaling these approaches to the complexity of real traffic is not trivial and may require different methods.

2.6 Requirements

Concluding from the analysis of the state-of-the-art, we derive the following requirements for a method that is able to predict the development of traffic situations with high accuracy from a series of noisy measurements:

1. Consideration of uncertainty
2. Consideration of mutual influences
3. Consideration of multiple abstraction levels
4. Integration of information
5. Application of data-driven models

Consideration of uncertainty Since the environment is only partially observable through noisy measurements, the current state cannot be determined with full certainty. Additionally, the development of a situation is uncertain. As a consequence, the uncertainties have to be considered in all aspects of the state estimation and prediction process.

Consideration of mutual influences A crucial aspect for understanding and anticipating driving behavior is the consideration of mutual influences between road users. Traffic participants continuously interact in traffic, which effects the behavior of each traffic participant. In congestions, for example, driving is mainly determined by the actions of the surrounding cars. Other examples are lane change maneuvers, merging or driving through intersections.

Consideration of multiple abstraction levels In order to predict a traffic situation accurately, it is beneficial to combine a continuous representation of basic properties such as positions of road users with a more abstract, symbolic representation, for instance, the right-of-way relationship between traffic participants or their intended driving routes. Purely symbolic representations of traffic situations suffer from discretization errors while purely continuous representations are not well suited to express symbolic relationships.

Integration of information To interpret situations right, it is important to take all relevant information into account. Combining different information sources, such as map data, road user measurements, measurements of the road infrastructure and traffic regulations allows drawing the right conclusions. For example, to determine who has right of way, it is necessary to know the road network configuration, the state of traffic participants and the applying traffic rules.

Not only is it important to take multiple information sources into consideration but also to integrate the information over time, i.e., to fuse new measurements with the existing knowledge about the environment in an ongoing process. Since the environment is only partially observable, the integration can reduce the uncertainty about the state of the environment and, therefore, allows making more accurate predictions.

Application of data-driven models In consequence of the complexity of traffic situations and the manifold of possibilities, it is beneficial to derive prediction models from data rather than from expert knowledge alone. Deriving models from traffic observations enables realistic and consistent models. Since the dynamics of traffic are only partly understood, manually formulated models are limited in their scope and accuracy. In addition, data-driven models have the advantage of potentially scaling to new situations and handling general changes in behavior patterns.

2.7 Conclusion

In this chapter, the related work in the state-of-the-art of state estimation and prediction in the context of traffic scenarios was discussed. Related work in the subdomains of behavior recognition, situation recognition and plan recognition was analyzed. We discussed the pros and cons as well as the limitations of the approaches and derived a set of requirements that enable the prediction of traffic situations from noisy measurements. While some of the proposed requirements are met by state-of-art approaches, none of the discussed work satisfies them all. The majority of approaches does not model the interactions between traffic participants. Only few works investigate the learning of prediction models that are able to generalize well. In this thesis, we develop an approach that meets all the identified requirements.

3 State Estimation and Prediction of Traffic Situations

We present a hierarchical Bayesian model that resembles the decision making of traffic participants in order to anticipate future developments of traffic situations.

3.1 Approach

This thesis tackles the problem of estimating the current state and predicting future developments of traffic situations based on histories of noisy measurements. We approach this problem by modeling the evolution of traffic situations as a stochastic process. The fully probabilistic treatment enables dealing with the noise in the measurements and, more importantly, reasoning about latent aspects that are not directly observable. Aspects of interest are for example, the internal states of drivers, which involve their goals and plans. The stochastic process is represented by a dynamic Bayesian network (DBN) (Murphy, 2012; Koller and Friedman, 2009), which represents the different aspects of a situation as random variables. These are connected through conditional distributions that comprise the models of the DBN. A temporal slice of the network represents the state of the environment at a specific time, i.e., a situation.

The key part of the DBN are the models. They define the semantics of the DBN and determine the relationships between the random variables. We use a combination of automatically learned and manually formulated models. By combining prior domain-specific knowledge with knowledge learned from data, we obtain a system that is able to generalize to a wide variety of situations. The basic idea behind the Bayesian model is the recursive formulation of the reasoning and decision making process of the traffic participants. By taking the perspective of each traffic participant, it is possible to reason about what an average driver would do in their position. With a fully defined Bayesian model of the stochastic process, distributions over current, future and past states of the environment given the evidence can be obtained by means of probabilistic inference methods.

In the following sections of this chapter, we develop a Bayesian model for the stochastic process and derive a suitable Monte-Carlo inference method for probabilistic reasoning. We first present a general model for plan recognition for multiple agents and with multiple layers of abstraction (Section 3.2). This model brings out the general idea behind the approach and is applicable in other domains as well. We then develop a specialization of this model for the traffic domain in Section 3.3. It addresses the domain specific properties and is tailored to support the policy model learning presented in Chapter 4. In Section 3.4, we derive probabilistic inference formulas for this model based on likelihood weighting to solve state estimation and prediction tasks. Based on the developed Bayesian predictive model, we outline in Section 3.5, how it can be embedded in Markov decision processes to solve stochastic decision making problems, e.g., for autonomous driving.

3.2 Hierarchical Policy Recognition Model for Multi-Agent Environments

In traffic constellations, there are a number of traffic participants involved, also called agents, which are assumed to act in order to achieve some goals, for example, to reach a specific location in a safe and comfortable way. Each agent is confronted with a sequential decision making problem, forcing him to continuously take actions until he reaches a terminal state, e.g., his final destination. Since we assume agents to act rationally, each agent chooses the action that he thinks is best to reach his goals w.r.t. to the current situation. Each action influences the state of the environment and thereby, the overall situation development. Even though each traffic participant makes choices on his own, their actions are highly coupled over time since any change of the environmental state can influence the decision making of the other agents. Examples for this interrelation range from cars that decelerate to keep a safety distance to slower driving cars, yielding at intersections to more subtle actions like extending a gap to provide a car enough space to merge in safely. In order to make accurate predictions of situation developments, interactions are considered in our model.

To model the decision making of road users, we adopt the concept of policies from Markov decision process (MDP) (Bellman, 1957; Puterman, 2009). A policy describes a mapping from states of the world to actions. In the field of reinforcement learning (Sutton et al., 1999), where MDPs are often used to describe the sequential decision making, policies are often modelled not as deterministic functions but as conditional distributions which define, how likely it is for an agent to execute a specific action for every possible state. We adopt this form of stochastic policies.

To be more precisely, we use a hierarchy of stochastic policies to describe the decision making of road users. If policies are allowed to be actions themselves (i.e. a policy can invoke other more refined policies), it is possible to build policy hierarchies. These hierarchies define the decision making of an agent on different abstraction levels comparable to plan hierarchies in classical planning (Sacerdoti, 1974) where high-level plans can execute sub-plans to handle specific parts. Policies of this kind are called abstract policies, a term coined by (Bui et al., 2002) in their work on AHMMs. The hierarchical ordering of policies can yield compact representations and policy hierarchies are in most cases easier to learn than a complex flat policy (Osentoski et al., 2004).

As a general model for policy recognition and prediction in multi-agent environments with a continuous state space, we propose the DBN depicted in Figure 3.1. The model consists of the states of the agents X, their policies Π together with termination nodes T, context nodes C and observations Z. Each random variable stands for a vector of nodes with one instantiation for each agent. Temporal dependencies are depicted as dashed arcs (directed edges) and direct dependencies within a time step are shown as solid arcs. The DBN is structured in n abstraction layers. On the lowest level, the state and action space is continuous, whereas on the higher levels, the spaces are discrete. Higher order policies control the selection of lower order policies. On the lowest level, the policies are primitive actions, which are executed for one time step. The higher the abstraction level the longer the policies are potentially executed.

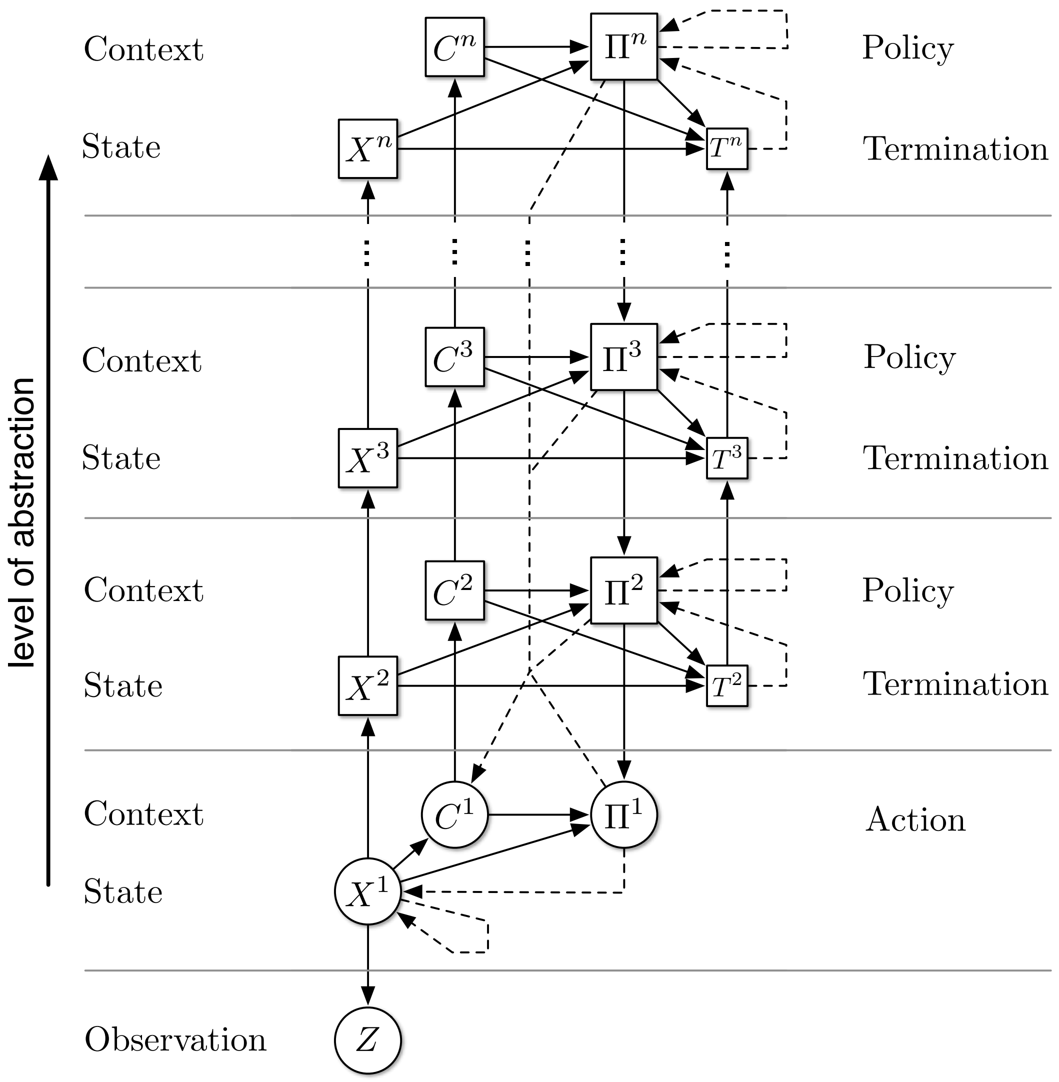


Figure 3.1: Hierarchical policy recognition and prediction model for multiple agents with n layers of abstraction. Solid arcs represent direct dependencies and dashed arcs represent temporal dependencies. Higher order policies control the policy selection on the levels below. The decision making of the individual agents is coupled through the context nodes.

The termination nodes have the function to detect if the policies of the corresponding level and the levels below have reached a terminal state. In this case, a policy transition is triggered. To give an illustrative example from the robotics domain, the hierarchization of the model for describing a robot arm executing manipulation tasks could look like follows: On the lowest level, the actions are the motor controls for the joints. On the second level, the policies form basic primitives like pick and place operations and on the highest level, the policies are higher order behaviors like the assembly steps for building a machine.

As shown in the structure of the DBN, it is assumed that only the low-level states X^1 of the agents are observable through measurements Z and even that only partially. Especially the executed policies are not directly observable. However they can be inferred over time through their indirect coupling with the agent states by reasoning from state changes to underlying causes.

The context nodes play an important role in this model. They serve two purposes. First, they describe the overall situation from the perspective of each traffic participant and second, they evaluate features from the configuration of agent states. Since the context nodes also depend on the policies of the other agents, their behavior is taken into account in the interpretation of the situational context and therefore, they directly influence the decision making of each agent. Figure 3.2 shows parts of the detailed structure of the model for multiple agents and the dependencies between context, state and policy nodes. Without the context nodes each agent would be treated independently and interactions could not be considered. We already made the point that this is a crucial requirement for making accurate predictions.

This model can be seen as a generalization of AHMMs that extend their use to multi-agent scenarios and domains with continuous state and actions spaces. In the next section, it is shown how this general model can be further developed for the traffic domain.

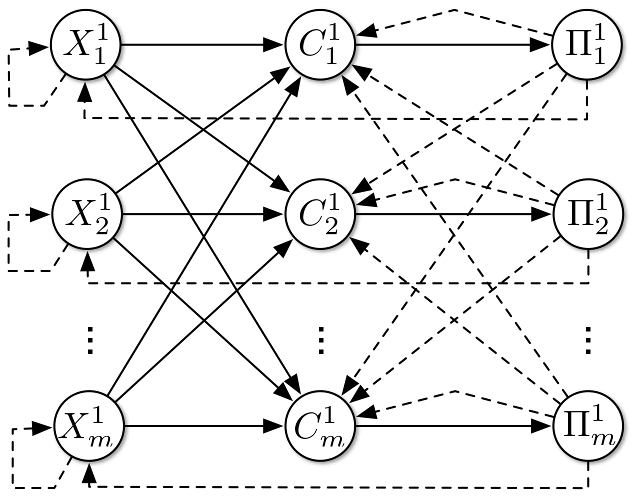


Figure 3.2: Detailed structure of the lowest level of the general DBN model with nodes instantiated for m agents. The policy choices of an agent depends on his context that subsumes the situation he is facing in a feature representation. The connections to the higher levels are omitted to prevent clutter.

3.3 Bayesian Model

To model the decision making of traffic participants in traffic situations, we use a policy hierarchy with three layers: Goals, plans and actions. The highest and most abstract layer considers the intermediate goals of a driver in form of target locations, like a junction of an intersection or a highway exit. The second layer is concerned with the routes a driver pursues to reach his goals. Depending on the situation, some routes are more likely than others. The primitive actions that a driver can conduct, namely steering and accelerating, form the lowest level. Depending on the overall situation, the drivers conduct actions that lead to driving trajectories that follow their planned routes in order to reach their goals. Figure 3.3 illustrates the relationships between goals and routes.

In the following subsections, we develop the Bayesian model for describing traffic episodes as a stochastic process and give details to all aspects. We first define the state spaces of the random variables governing the Bayesian model (3.3.1) and then explain the structure of the DBN (3.3.2) and finally define the models (3.3.3). The resulting Bayesian model can be used in multiple ways by solving the corresponding inference task:

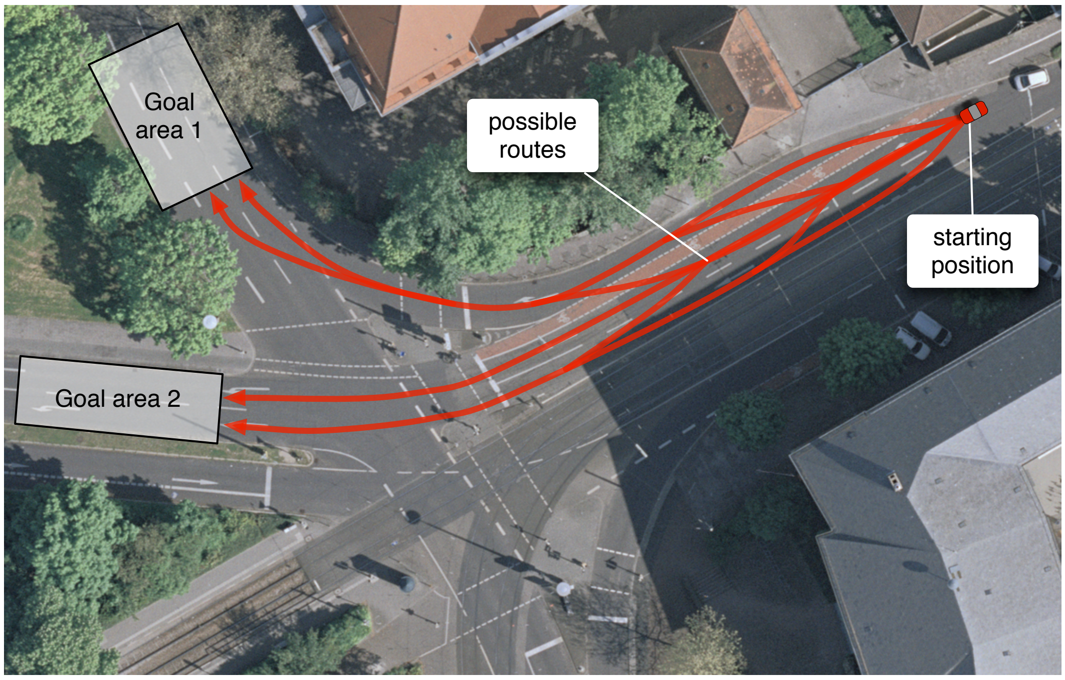


Figure 3.3: Illustration of the many possible routes a driver can take to reach one of the goal areas in an intersection scenario. The Bayesian model we are presenting reasons about the goals of drivers and the probabilities of choosing routes to reach their goals. (Aerial image provided by (City of Karlsruhe, 2010)).

State estimation Based on a history of measurements, the distribution over the current state of a situation can be inferred.

Policy recognition As a by-product of state estimation, distributions over the goals and plans of traffic participants can be derived.

Predictions Starting from a given situation, possible future developments can be inferred together with their probability of actually happening.

Reconstruction Distributions over past traffic states or only the most likely state sequence given a history of noisy measurements can be obtained with smoothing methods.

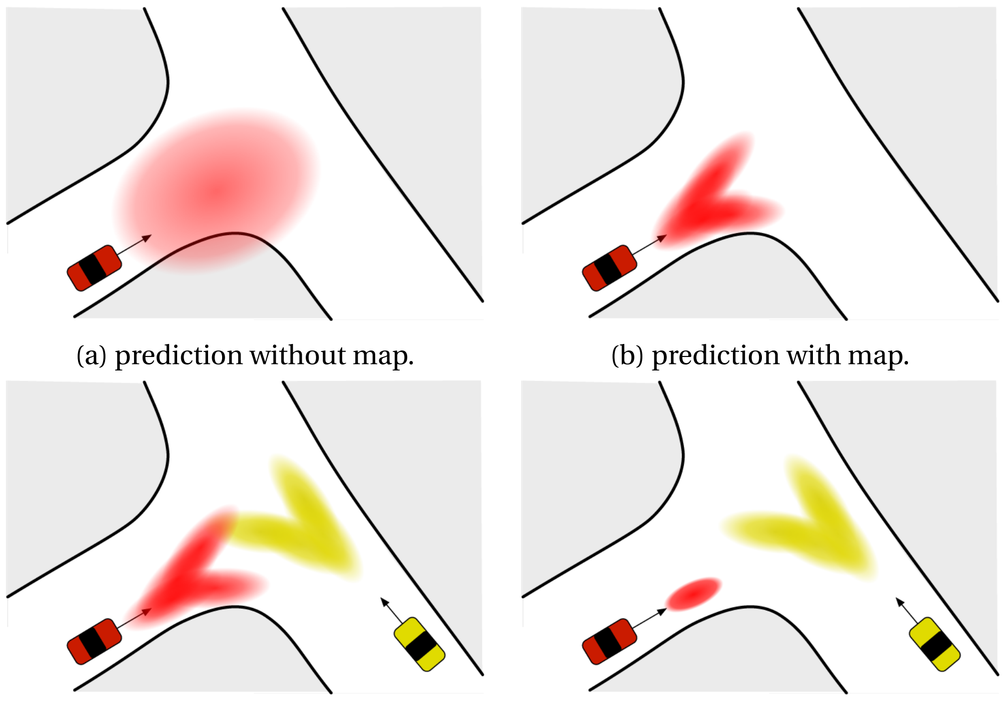
3.3.1 State Space

The joint state space of the Bayesian model consists of random variables describing the different aspects of a situation. The joint state space comprises a mixed space combining states of continuous, discrete and categorial nature. The Bayesian model consists of the listed variables:

X	traffic participants' states
Z	measurements
C	situational context
Π^A, Π^R, Π^G	action, route and goal policies
$T^R,\ T^G$	policy termination states
M	road network

All these aspects are taken into account to improve the prediction accuracy. In contrast to approaches that neglect the road network structure or predict the behavior of road users independently, interpreting situations and reasoning about the intentions of others can significantly reduce the uncertainty of predictions. Figure 3.4 schematically illustrates the resulting prediction uncertainty when different sources of information are used in the reasoning process. In the first case (a), the motion of a car is predicted based only on its current velocity and orientation. Since all possible motions have to be considered, the uncertainty in the prediction is very high. Taking into account the lanes and the prior knowledge that cars mostly drive on lanes improves the prediction accuracy substantially (b). However, the risk of an accident is not assessed correctly if the road users are predicted independently (c). If the context is additionally considered in the reasoning process together with knowledge about traffic regulations and how these regulations affect the decisions of drivers, the predictions become more realistic (d).

The random variables of the Bayesian model are defined in the following subsections. For clarity, we describe the semantics of the random variables for a single traffic participant indexed by the variable i at the time step t without loss of generality. The resulting dynamic Bayesian network for a situation comprises an instantiation of these random variables for all n traffic participants which are coupled through the models (3.7). For example, the context of a traffic participant takes the states of all others into consideration. This is one aspect that



(a) prediction without map. (b) prediction with map.

(c) prediction without context. (d) prediction with context.

Figure 3.4: Schematic illustration prediction capabilities when considering different aspects of the environment. If the structure of the road network is not considered, the prediction has to account for all possible movements, which leads to a high degree of uncertainty (a). Taking the road network into account improves the prediction accuracy (b,c). Only when context is taken into account, it can be predicted that the left car will most likely stop due to right-of-way regulations (d).

differentiates this approach from classical multi-target tracking, where objects are tracked individually.

Conventions To prevent clutter in the notion of the random variables, we omit the index i that refers to the i'th traffic participant and the time index t whenever there are no ambiguities. Consequently, where not stated otherwise, X stands for $X_i^{(t)}$. To refer to random variables from the previous time step $t-1$ we use the notion X^-. We also use the short form $X_{1:n}$ to denote the vector $\left(X_1, X_2, \ldots, X_n\right)$ and $X_{1:n\setminus i}$ for the vector $X_{1:n}$ without X_i, i.e., $X_{1:n\setminus i} = \left(X_1, X_2, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n\right)$.

Traffic participants' states X

A state x of a single traffic participant is described by his position $\left(x_1, x_2\right)$, heading ψ (orientation) and velocity v in a global world frame (see Figure 3.5). The state also comprises the width w and length l of the bounding box, enclosing the shape of the traffic participant. The position of a traffic participant coincides with the center of his rear axle in compliance with the motion model we use (3.3.3). The state is defined as

$$x = \left(x_1, x_2, \psi, v, w, l\right)^{\mathsf{T}} \in \mathbb{R}^6 .$$

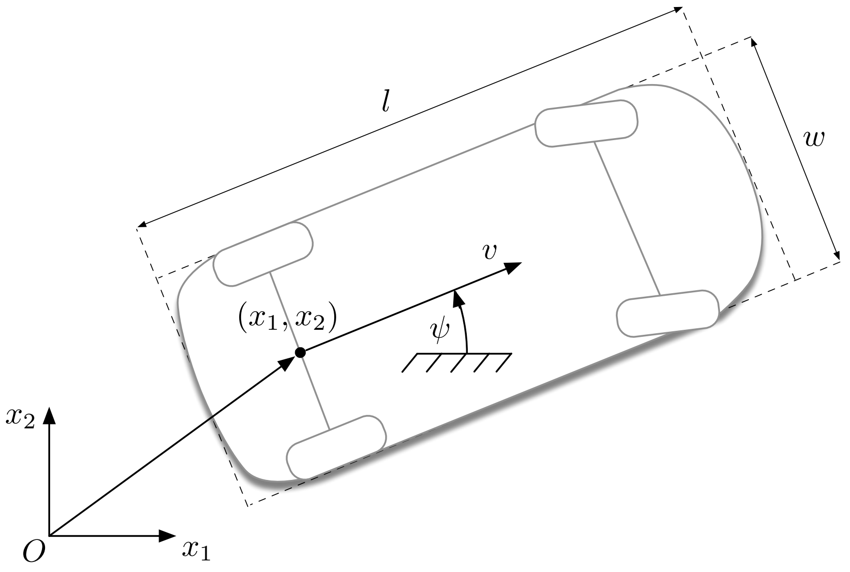


Figure 3.5: Representation of a traffic participants' state.

Measurements Z

We assume that all basic states of a traffic participant can be measured directly with noise, such that Z and X have the same domain, yielding

$$z = \left(x_1, x_2, \psi, v, w, l\right)^{\mathsf{T}} \in \mathbb{R}^6 .$$

We abstract in this work from specific sensors and their preprocessing of the raw sensor data to keep the model more general. The specific characteristics of a sensor are incorporated in the measurement model that also specifies the noise of the sensor.

Road map M

The road map is given as graph as described in the Appendix A.1. The road map is assumed to be known and accurate. It therefore must not be part of the state space and can be used implicitly in the models. The graph consists of vertices and edges representing geospatial points and ways. Since we focus on traffic participants driving on roads in this work, we only consider edges representing lane segments in the following. The geometry of a lane is approximated by lane segments with rectangular shapes. While other representations exist that better account for the curved nature of roads, this approximation offers some benefits like fast lookups when matching traffic participants to lanes. The set of all lane segments is denoted as $\mathscr{L}$. Relations between lane segments describe the nature of their relationship, e.g., if lanes are adjacent or crossing. These relationships are queried via helper functions and become useful in the definitions of the models.

Action Policies Π^A

The actions represent the traffic participants' influence over their own motion by controlling their yaw rate ω and acceleration a. Other actions of minor influence on the development of a situation, like activating the headlights or the windshield wipers are not considered in this work. The action policies are defined as

$$\pi^a = \left(a, \omega\right)^\mathsf{T} \in \mathbb{R}^2 .$$

Route Policies Π^R

The route policies form the intermediate level of abstraction and connect the strategical goal policies layer with the tactical action layer. A route policy is represented by a route that leads towards a goal area that a driver wants to reach. A route is a sequence of lane segments that comprise a valid path in the route network graph. This means that the lane segments along the path are directly connected and that there exist drivable transitions between them. Figure 3.3

shows an example, illustrating the possible paths and goals in an intersection scenario.

Routes abstract from the specific dynamics and give rise to many different realizations by the action layer, i.e., they make no proposition about the trajectory of a car along the route. The concrete trajectories are chosen by the action policy model depending on the actual situation. Routes thereby contain no information on how fast a driver will drive along the route or if he will drive in the center of the lane or more at the boundary. What they provide is a frame in which a driver is going to act together with the transitions like lane changes necessary to reach the end of the route. Based on the routes, it is possible to derive which traffic participants are possibly going to interact but not when. Therefore, additional information from the action layer has to be considered. By abstracting from the dynamic realization, routes are not as volatile as trajectories and provide more dependable information for longer prediction horizons.

The set of all routes $\mathscr{R}$ is derived by enumerating all possible ways from each lane segment to each lane segment of each goal region. Depending on the branching factor of the road network graph, the set can become quite large. We reduce the set to a manageable size by using a finite planning horizon and considering only routes up to a length of h lane segments. This also guarantees that the set is finite. Formally, the set of routes is given by

$$\begin{aligned}\mathscr{R} = \{\langle l_1, l_2, \dots, l_h\rangle \mid (\forall k \in \{1, \dots, h\} : l_k \in \mathscr{L}) \wedge \\ (\forall k \in \{1, \dots, h-1\} : \mathit{transition}(l_k, l_{k+1}) \wedge l_k \neq l_{k+1})\} \ . \end{aligned} \quad (3.1)$$

The function $\mathit{transition}(l_k, l_{k+1})$ queries the route network graph to determine if there exists a valid transition between lane segment l_1 and l_2, i.e., it tests if the lane segments are geometrically connected, so that it is possible to drive directly from l_1 to l_2. The route policies are defined through the set of routes $\pi^R \in \mathscr{R}$.

Goal Policies Π^G

The goal policies form the highest level of abstraction. The set of goals $\mathscr{G}$ describes regions that are potential targets of drivers. The regions are derived from the road network graph by clustering larger groups of adjacent lanes. The clus-

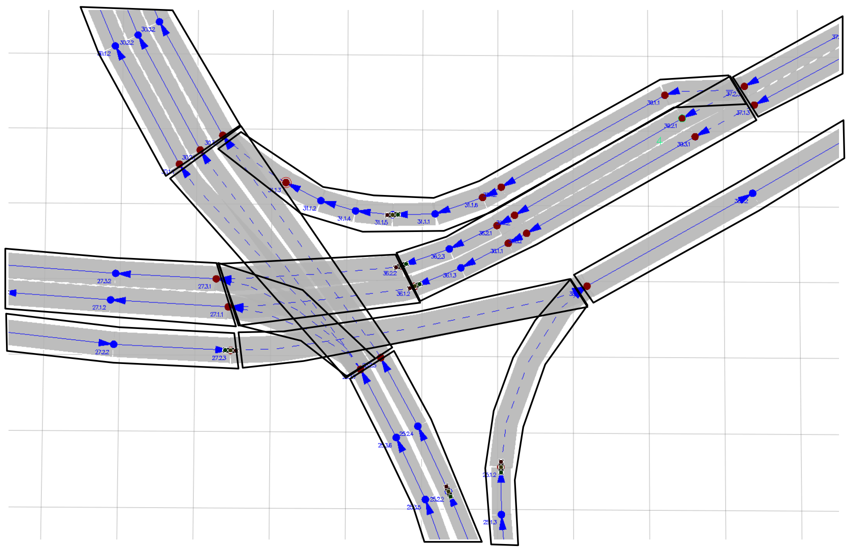


Figure 3.6: Partitioning of a road network graph into goal regions.

tering forms a partitioning of the set of lanes, i.e., it divides $\mathscr{L}$ into sets that are mutually exclusive and collectively cover all elements of $\mathscr{L}$. We use a criterion that clusters lanes which head into the same direction and start a new region as soon as lanes split up or merge. For example, the junctions of an intersection form goal regions to enable the estimation of the most likely exit junction a driver is heading to. Figure 3.6 shows a route network graph and the partitioning into goal regions. The connectivity between goal regions is directly derived by examining the connectivity between the associated lane segments in the road network graph. The goal policies are defined through the set of goals $\pi^G \in \mathscr{G}$.

Policy Termination States T^G, T^R

The termination states are binary random variables, indicating if a policy has either reached a terminal state or is continued. For example, in the case of route policies, the termination state is set to *true*, if the traffic participant deviates from the planned route. In that case, a new new route has to be chosen in accordance to the goal policy. State t^G indicates the termination status of the goal policy and t^R that of the route policy:

$$t^G,\ t^R \in \{true,\ false\}\ .$$

Context C

The context describes the situation from the perspective of a traffic participant through relations. The representation of the context is heterogeneous and combines features of continuous, discrete and categorial nature. It provides a high-level level representation of the situation as the basis to resemble the decision making of each traffic participant and support the learning of the policy models. The relations are extracted by feature functions from low-level information and background knowledge. For example: Based on the position of each traffic participant together with the road network configuration and the traffic rules, one can derive the right-of-way relations between traffic participants at an intersection. Knowing the right of way relationships provides strong cues on how the traffic participants are going to act. Section 3.3.3 explains the feature functions and dependencies in detail.

The context is further factorized into several random variables, describing features of a traffic participant as well as relations between traffic participants and their planned routes. The context $C = \left(C^L, C^R, C^X\right)$ comprises the following elements:

- C^L: The current lane
- C^R: The features of the planned route
- C^X: The relations between traffic participants and their planned routes

The elements are described in the following paragraphs and in Section 3.3.3, which defines the corresponding models for the calculation of the context features.

Current lane C^L The current lane or more precisely, the current lane segment, is the one lane segment that the vehicle is currently *on*, i.e. $c^L \in \mathscr{L}$. Since a car can be physically on several lanes at the same time, e.g., in intersections, the lane segment a car is currently *on* means in this context, the lane segment that servers as the main spatial guidance for the driver.

Route features C^R The route features describe the properties of the planned route of a traffic participant. These properties are described through relations

R^N between the lane segments of the route as well as relations R^L between the traffic participant and the lane segments of the route.

The relations between lane segments describe how consecutive lane segments of the route are connected in the route network graph, e.g., if a lane directly proceeds another or is an adjacent lane, making it necessary to change lanes. The intended driving direction is also considered, which makes it possible to distinguish a normal lane change from an overtaking maneuver, where the car has to drive on an oncoming lane in the opposite direction.

The other features describe relations between the traffic participant and the lane segments of the planned route together with features of the individual route segments. Features of a lane segment comprise the width and the length, the type of lane boundaries, the type of lane and everything else that is specific for this segment. Regulations like speed limits or the state of traffic lights are directly mapped to the lane segments to which they apply and appear as lane features. The route feature vector is given by

$$c^R = \left(r^N_{1:h-1}, r^L_{1:h} \right) .$$

Relations between traffic participants C^X The most important aspects to reason about the interactions between traffic participants are the relationships between traffic participants. The vector c^X comprises the relations between traffic participant i and all other traffic participants. It is defined as

$$c^X = \left(r^X_{i,1:n\setminus i} \right)$$

with $r_{i,j}$ being a feature vector of relations between the i's and j's traffic participant.

The set of relations between traffic participants ranges from basic relations, such as their relative distance, their difference in speed and their relative orientation, to complex ones like right-of-way relations and relations that take into account the planned routes of traffic participants. An important relation of this type is the interaction between routes, which is described in more detail in Section 3.3.3. It tells how routes interrelate, e.g., if they overlap, merge or doesn't intersect at all. Taking intended routes into account can significantly improve the expressiveness of relations. For example, a common time-to-collision (TTC) estimate (Hayward, 1972) becomes more accurate when relaxing the assumption of a constant heading and considering the intended driving corridor instead.

Another important class of relations are traffic rule relations. Based on the current positions, the planned routes and the overall road network graph, traffic rules can be evaluated and relations like right-of-way can be inferred.

The list of relations named so far comprising the context of traffic participant is by far not exhaustive. And while we give more examples and details on relations in Section 3.3.3, many more are thinkable and have been proposed in the literature. See for example (Bonnin et al., 2012) and (Demčenko et al., 2009). The important aspect to keep in mind is that the context features provide the basis for explaining the behavior decisions of traffic participants. Any useful information included in the context can foster the learning process of the policy models and thereby improve generalization.

3.3.2 Dynamic Bayesian Network Structure

With the state spaces of the random variables defined, we can now define the state space of a situation by joining them together. The joint distribution over all states comprising a situation s is given by

$$P(S) = P(X, C, \Pi^G, \Pi^R, \Pi^A, T^G, T^R) \,. \tag{3.2}$$

The random variables are all vectors of random variables, joining together the instances of the individual traffic participants. The development of a situation over a period of T time steps can be described as a sequence of situations $s^{(t)}$. The joint distribution is $P(S^{(1:T)})$. We assume that the process is first order Markovian, i.e., $P(S^{(t)}|S^{(1:t-1)}) = P(S^{(t)}|S^{(t-1)})$ (Murphy, 2002). The joint distribution then factorizes to

$$P(S^{(1:T)}) = P(S_1) \prod_{t=2}^{T} P(S^{(t)}|S^{(t-1)}) \,. \tag{3.3}$$

If we consider the situations together with the measurements $z_{1:T}$ and assume that the measurements are conditionally independent knowing the true state of a situation, the overall joint distribution factors as follows

$$P(S^{(1:T)}, Z^{(1:T)}) = P(S^{(1)})P(Z^{(1)}|S^{(1)})\prod_{t=2}^{T} P(S^{(t)}|S^{(t-1)})P(Z^{(t)}|S^{(t)}) \,. \tag{3.4}$$

The conditional joint distribution $P(S^{(t)}|S^{(t-1)})$ can be further decomposed under assumptions of conditional independence and factors as follows

$$\begin{aligned}
&P(S|S^{-}) = \\
&\quad P(X, C^L, C^R, C^X, T^G, \Pi^G, T^R, \Pi^R, \Pi^A | X^{-}, \Pi^{G-}, \Pi^{R-}, \Pi^{A-}) = \\
&\quad \underbrace{P(C^L|X, \Pi^{R-})}_{\text{lane matching model}} \; \underbrace{P(C^R|X, \Pi^R)}_{\text{route context model}} \; \underbrace{P(C^X|X, C^R, \Pi^R, \Pi^{A-})}_{\text{relations model}} \\
&\quad \underbrace{P(\Pi^G|X, C^L, T^G, \Pi^{G-})}_{\text{goal policy model}} \underbrace{P(T^G|C^L, T^R, \Pi^{G-})}_{\text{goal termination model}} \\
&\quad \underbrace{P(\Pi^R|X, C^L, C^{X-}, \Pi^G, \Pi^{R-}, T^R)}_{\text{route policy model}} \; \underbrace{P(T^R|C^L, \Pi^{R-})}_{\text{route termination model}} \\
&\quad \underbrace{P(\Pi^A|X, C^R, C^X, \Pi^R, \Pi^{A-})}_{\text{action policy model}} \underbrace{P(X|X^{-}, \Pi^{A-})}_{\text{motion model}} \,.
\end{aligned} \tag{3.5}$$

The factors of the conditional joint distribution form the models of the overall Bayesian model.

Figure 3.7 shows the resulting structure of the dynamic Bayesian network representing the development of traffic situations as a stochastic process. The network is organized hierarchically in the order of the considered abstraction levels. On the highest level, the goals are allocated. They represent the target areas, which traffic participants are trying to reach. On the intermediate level, corresponding routes represent feasible ways to the goals but do not consider the concrete dynamics of the objects. These are considered on the lowest level through the actions of the traffic participants. All decisions of the traffic participants, resembled on the different levels of abstraction, are influenced by their situational context. Only basic features of traffic participants can be measured, like their position or orientation. The higher level decision making states are

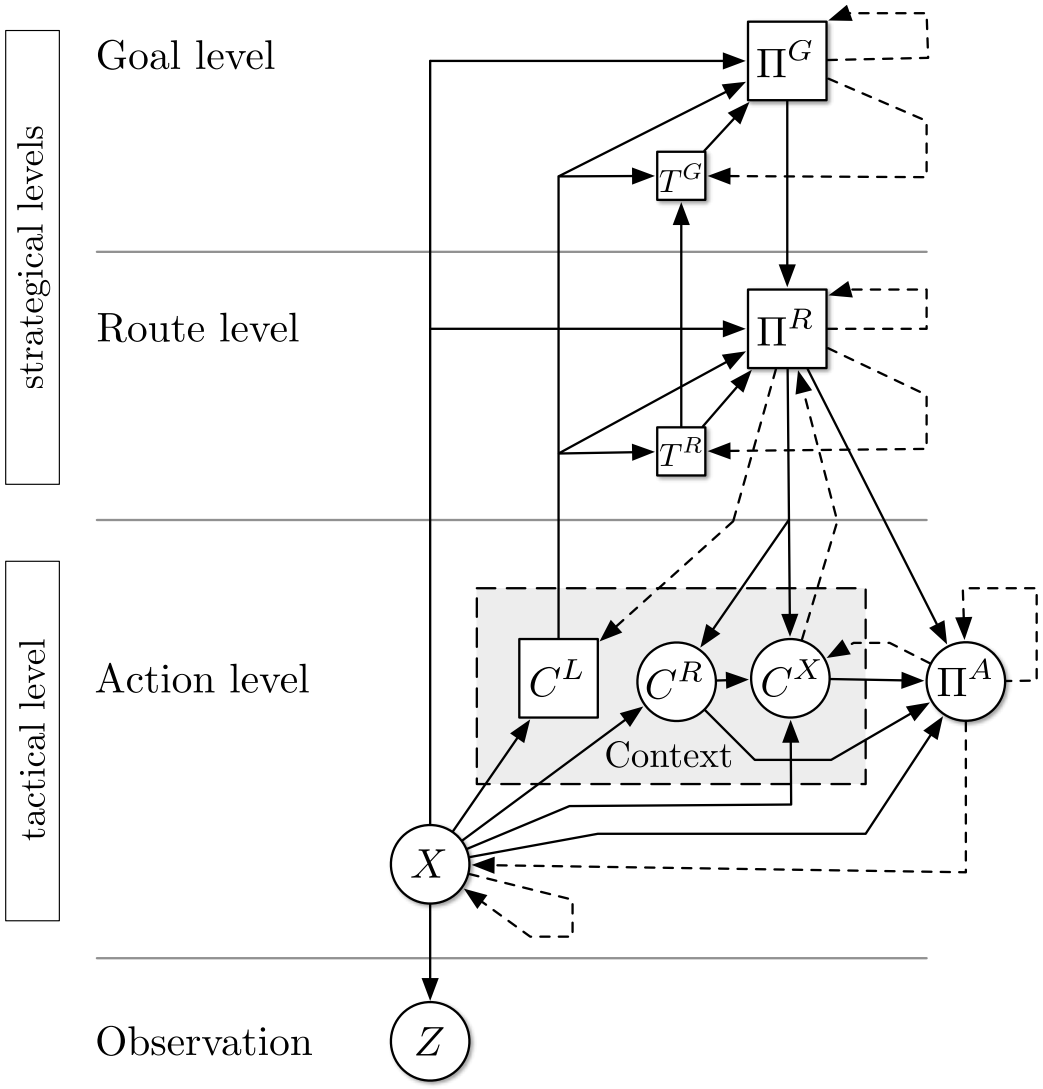


Figure 3.7: DBN representation of the policy execution and recognition model for multiple interacting traffic participants. The decision making is modelled on three abstraction levels. On the highest level, the goals Π^G of the traffic participant influence their route choices Π^R on the intermediate level, which in turn determine their situation-specific action choices Π^A on the lowest level. All decisions are grounded on the context information (C^L, C^R, C^X). From all the nodes in the DBN, only the basic road user states X are observable through the measurements Z. All other aspects have to be inferred over time.

not directly observable and can only be indirectly deduced by integrating information over time.

The models are even further factorized as shown in the Figure, since most models only depend on the aspects of a single traffic participant. The coupling of traffic participants mainly happens through the context model $P(C^X|X, C^R, \Pi^R, \Pi^{A-})$ that takes all traffic participants into account. After describing all the models in the following sections in detail, we will show how to perform inference in the DBN in Section 3.4.

3.3.3 Models

This section addresses the models of the DBN (Figure 3.7). The models have the form of conditional distributions and relate the random variables of the DBN to each other. Besides the definitions, we provide insights and the reasons that led to the specific modeling choices.

Context Models

The context models establish the relationships between all relevant aspects of a situation and makes them explicit in form of relations. These relations describe the situation from the perspective of each traffic participant. The features used to describe a situation are derived using all the given information from the last and current estimate together with background knowledge such as map data, traffic rules and geometrical and physical principles.

The reason for making these relationships explicit are twofold. First of all, for some applications, these relations are directly of interest. A lane change assistant for instance requires the information whether other cars are blocking the target lane and also their relative velocities to decide if a lane change is safe.

The main reason nonetheless is to provide background knowledge to the learning process, which we present in the next chapter (4), and thereby enabling and improving generalization. This works in multiple ways. Since the context directly feeds into the action policy model, the context features form the basis of decision making. By providing non-linear, information rich features, the learning problem itself is simplified, since the relationship between situations and chosen actions can be captured more easily by the learning algorithm. For example, a resulting stopping maneuver can be expressed much easier by a policy model for a multitude of traffic constellations if features such as right-of-way relations are directly available. While such non-linear dependencies could also, in principle, be learned, making the context models obsolete, a lot more data and the right inductive bias would be necessary to achieve the same generalization accuracy (Domingos, 2012).

The information extraction and interpretation in form of context features is especially useful for making map knowledge accessible to the learning algorithm. If the whole map would be presented in raw encoding, it could not be expected that useful dependencies are learned that generalize to other maps with current state-of-the-art methods. But since background knowledge about geometrical concepts is available, we can supply geometrical features derived from the map data to the learner, facilitating it to find concepts that generalize to other maps and road configurations.

As stated in definition of the state spaces (Section 3.3.1), the context C is subdivided into three types of random variables, namely the current lane C^L, the features of the current route C^R and the relations between the traffic participants C^X. The corresponding models are defined in the following three sections.

Lane Matching Model $\boldsymbol{P(C^L|X,\Pi^{R-})}$

The lane matching model defines a conditional distribution over all lane segments of the route network. It quantifies the probability of a traffic participant being *on* a specific lane segment.

This association step is necessary to establish the connection between the traffic participants represented in a global word frame and the map data. A traffic participant can be *on* several lane segments at the same time but only one serves as the main spatial guidance at a time. Only the lanes with geometries that intersect with the shape of the car can have a probability greater than zero and are called matching candidates. Figure 3.8 shows an example of the lane matching in an intersection. The shape of the red car intersects with multiple lanes depicted in green. These lanes are the possible matching candidates, all other lanes have zero probability. Depending on the driving area, the number of

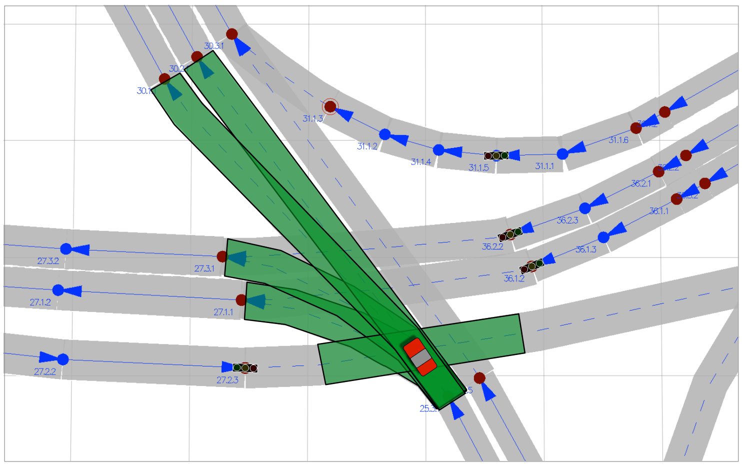


Figure 3.8: Illustration of the lane matching. Lane segments that serve as matching candidates marked in green. The probabilities are assigned depending on the planned route and the position and orientation relative to the lane segments frames. Lane segments that do not intersect with the car shape have a probability of zero.

candidates can vary widely. In intersections with many overlapping lanes, there are far more candidates than for example on a highway, where the mapping is unambiguous most of the time. In case that a traffic participant leaves the defined area of the route network, e.g., if exiting a lane to drive into a garage, the search radius is expanded and the lanes next to the current location are considered.

The conditional distribution is defined as follows

$$p(C^L = l|x,\pi^R) = \begin{cases} p_m & \text{if } \mathit{intersect}(x,l) \wedge l \in \mathscr{L}_{\pi^R} \\ \sigma(l|x) & \text{if } \mathit{intersect}(x,l) \wedge \\ & \nexists l_j \in \mathscr{L} : (\mathit{intersect}(x,l_j) \wedge l_j \in \mathscr{L}_{\pi^R}) \\ 0 & \text{otherwise} \end{cases} . \quad (3.6)$$

The helper function $\mathit{intersect}(x,l)$ evaluates to *true iff* the shape of the lane segment l intersects with the shape of a traffic participant determined by x. The set $\mathscr{L}_{\pi^R}$ denotes all lane segments that are part of the planned route π^R.

Three cases are distinguished in the definition of the distribution. The first one is the most relevant and ensures consistency of the matched lane with the planned route of a traffic participant: i.e., if some of the matching candidates coincide with segments of the planned route, then only these have a probability greater than zero and are equally probable, which is expressed by the constant p_m. If a traffic participant has departed from his planned route, then the probability of a matching candidate relies on the relationship between the traffic participant and the lane segment expressed by $\sigma(l|x)$. Depending on the relative orientation and the distance of a traffic participant to borders of the lane, segments are more or less likely. Intuitively, lanes with the intended driving direction matching the heading of a traffic participant are more likely than lanes that are perpendicularly oriented.

In order to calculate the probabilities in an efficient manner, we use a k-d tree (Bentley, 1975) to retrieve the matching candidates without having to search the whole road network graph.

Route Features Model $P(C^R|X,\Pi^R)$

The route features are described by deterministic functions. The conditional density can therefore be expressed as a multivariate Dirac distribution

$$p(c^R|x,\pi^R) = \delta(c^R, \left(r^N_{1:h-1}(\pi^R), r^L_{1:h}(x,\pi^R)\right)) \,. \tag{3.7}$$

Two sorts of route features are calculated: the neighboring relations r^N between consecutive lane segments along the route and the relations between the traffic participant and the segments of the route r^L. The feature functions depend on the route and the state of a traffic participant.

The following relations are evaluated:

transition type indicates how the two lane segments are connected. It can take one of the following values {*proceeding_lane*, *preceding_lane*, *left_lane*, *right_lane*, *left_opposite_lane*, *right_opposite_lane*}.

delta angle rates the difference in orientation and approximates the road curvature.

The other type of relations $r^L_l(x,\pi^R)$ provides properties of the lane segments and relations between the traffic participants and the lane segments. For every segment l of the route, the following relations are evaluated based on the current state x:

length, width quantify the dimensions of the lane segment.

markings indicate the type of lane markings for the left and right border of the lane. Values are {*none, solid_white, broken_white, double_white*}.

speed limits The limits are retrieved from the traffic signs or are known in advance and mapped to the corresponding lanes. See (Nienhüser, 2014) for techniques of identifying and mapping traffic signs from on-board sensors.

lane type provides information about the type of the lane. Possible values are {*normal, priority_lane, intersection_lane, construction_lane*}.

traffic lights state indicates the state of the corresponding traffic light, in case that the lane leads to a signaled intersection. Possible values are {*green, yellow, red, yellow_red, blinking, off, none*}.

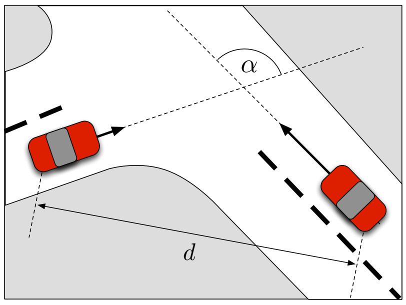
road network properties aggregates properties of connected lanes and areas based on the road network graph, e.g., if the lane segment has a left or right neighboring lane or if it intersects with other lanes or pedestrian crossings.

relative vehicle pose The vehicle pose is transformed into the local coordinate system of the lane segment, which has its origin at the beginning of the lane segment and the x_1-axis aligned with the intended driving direction as depicted in Figure 3.9b.

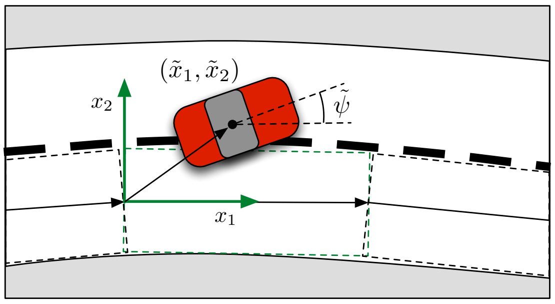
distance to segment measures the distance of the vehicle to the entry point of the segment.

time to enter Estimate of the time needed until a traffic participant enters the lane segment based on his current velocity. If he is already on that lane segment the value is zero.

The list of relations and their possible values is not complete, but give an understanding of what the context information consists of. However, most situations can be handled with the listed relations. It is easy to expand the set of relations to cover more special cases, e.g., if a speed limit only applies to a specific vehicle class or weather condition.



(a) Relations between traffic participants.



(b) Relations between traffic participants and lane segments.

Figure 3.9: Visualization of some of the context relations evaluated by the context model. This representation of a situation from the perspective of each traffic participant serves as input for the action policy model to derive the likely actions each traffic participant is going to conduct.

Traffic Participants Relations Models $P(C^X|X, C^R, \Pi^R, \Pi^{A-})$

Analogously to the route features model, the traffic participants relations model is defined as a multivariate Dirac distribution. It additionally depends on the states and policies of the other traffic participants.

$$p(c^X|x_{1:n}, c^R_{1:n}, \pi^{R-}_{1:n}, \pi^{A-}_{1:n}) = \delta\left(c^X, \left(r^X_{i,1:n\setminus i}(x_{1:n}, c^R_{1:n}, \pi^{R-}_{1:n}, \pi^{A-}_{1:n})\right)\right) . \tag{3.8}$$

The relations feature function $r^X_{i,j}$ calculates the relations between the two traffic participants i and j from the perspective of participant i. The planned routes of the traffic participants play an important role for the derivation of the relations, since they allow detecting and interpreting interactions between them. The following relations are determined by the function $r^X_{i,j}$:

relative vehicle pose projects the global pose of traffic participant j into the local coordinate frame of traffic participants i. The x_1-axis of the local coordinate frame is aligned with the heading of the traffic participant. This non-linear transformation provides a view of the surroundings relative to traffic participant i, which is invariant to global translations and rotations.

distance calculates the euclidean distance between the two traffic participants.

interaction type describes the interrelation of routes and can take values from the set {*nointeraction*, *overlap*, *merge*, *diverge*, *cross*, *confront*}. The types of interactions are depicted in Figure 3.10. The classification of the interaction type is realized using the road network graph and the geometry of the lanes. The type of interaction carries valuable information for the decision making, e.g., if the routes of two traffic participants do not interact (3.10a), it is likely that they do not influence their next maneuver at all. On the other hand, if two routes overlap (3.10b), the actions of the preceding car strongly influences the behavior of the car behind.

time to interaction Estimate of when the traffic participants are going interact based on their current velocities and their routes.

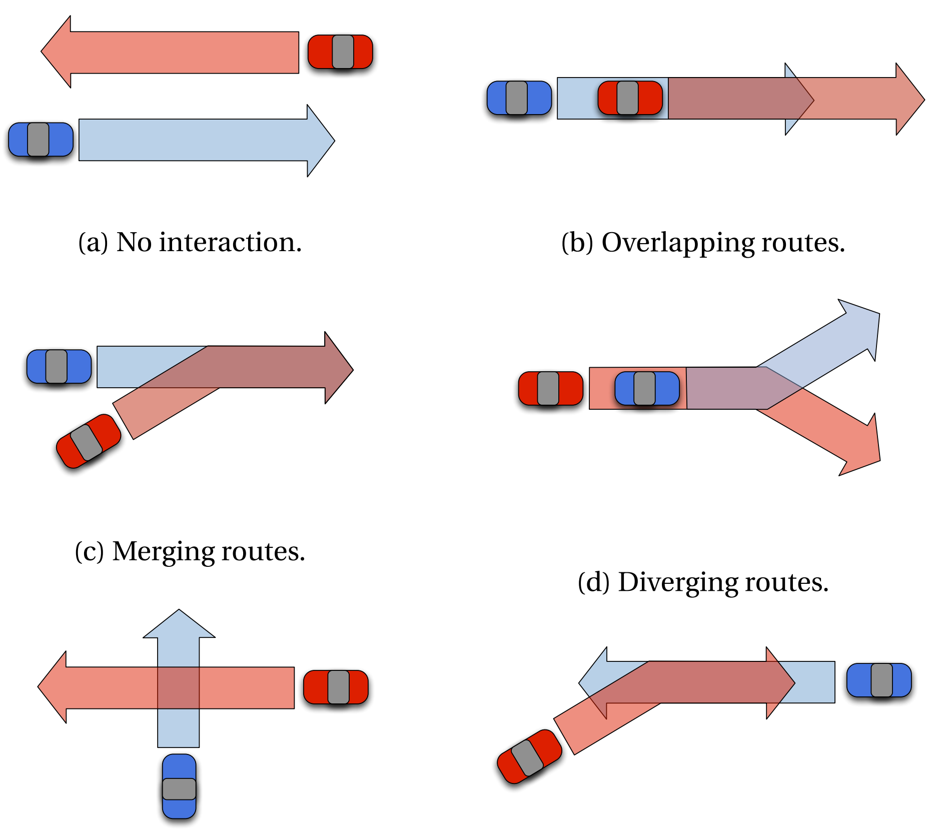
right of way indicates if traffic participant i has right of way over the other vehicle. The right of way is determined according to the traffic regulations, the current positions, the planned routes of the traffic participants, the route network and the type of intersection they are approaching. The relation can evaluate to *true*, *false* or *not_applicable*.

All these features become important when it comes to learning because they extract and interpret implicit information and also make use of present invariants, such as the rotations and translations of whole situations.

Goal Policy Model $\boldsymbol{P(\Pi^G|X, C^L, T^G, \Pi^{G-})}$

The goal policy model describes the transition probabilities between goals. Transitions between goals are necessary since the estimated goals of traffic participants have the semantics of intermediate goals rather than final destinations. Since intermediate goals have a more direct influence on the tactical decisions than final goals, they are of more interest for the task of predicting the development of a traffic situation.

From the perspective of a moving car, other traffic participants can often be seen only for a limited period of time, especially in inner cities. These periods are often too short and the observed behaviors not indicative enough to infer the final destinations with enough certainty. For example, seeing a car turning at an intersection does not tell much about whether it is going to the next mall or is heading for Berlin.



(a) No interaction.

(b) Overlapping routes.

(c) Merging routes.

(d) Diverging routes.

(e) Crossing routes.

(f) Confronting routes.

Figure 3.10: Schemata of the different types of interaction relationships between the routes of traffic participants.

The transition between goal policies depends on the state of a road user x, his current lane c^L and his goal π^{G-}. The goal policy model is defined as follows

$$p(\pi^G|x, c^L, t^G, \pi^{G-}) = \begin{cases} \sigma_+^G(\pi^G|x, c^L, \pi^{G-}) & \text{if } t^G = \mathit{true}, \\ \sigma_-^G(\pi^G|\pi^{G-}) & \text{otherwise} \end{cases} . \tag{3.9}$$

As long as the current goal is not reached but still reachable, which is indicated by the goal termination flag t^G being *false*, the current goal policy is pursued, yielding the definition of σ_-^G as

$$\sigma_-^G(\pi^G|\pi^{G-}) = \begin{cases} 1 & \text{if } \pi^G = \pi^{G-} \\ 0 & \text{otherwise} \end{cases} . \tag{3.10}$$

Otherwise, a new goal policy is chosen according to the distribution $\sigma_+^G(\pi^G|x, c^L, \pi^{G-})$. This sparse distribution assigns probabilities greater than zero only to goal regions that are directly reachable from the current position of a traffic participant, i.e., the goal regions considered are adjacent to the current region and there exists at least one route from the current location to the target goal region. The preferences between reachable goal regions are set according to the properties of the goal regions. For instance, following a main street is more likely than taking an exit into a side alley. Section 4.3.1 explains how these preferences are learned.

Route Policy Model $\boldsymbol{P(\Pi^R|X, C^L, C^{X-}, \Pi^G, \Pi^{R-}, T^{R-})}$

From the viewpoint of abstraction, the route policy model is located below the goal policy model and expresses the probabilities of route choices when a traffic participant tries to reach a specific goal. The distribution over all possible routes depends on the current goal π^G of a traffic participant and his situational context. The model is defined as

$$p(\pi^R|x_{1:n}, c^L, c^{X-}, \pi^G, \pi^{R-}, t^R) = p(\pi^R|s^R, t^R) \tag{3.11}$$

$$= \begin{cases} \sigma_+^R(\pi^R|s^R) & \text{if } t^R = \mathit{true} \\ \sigma_-^R(\pi^R|\pi^{R-}) & \text{otherwise} \end{cases} \tag{3.12}$$

with s^R summarizing the situation dependencies as

$$s^R = (x_{1:n}, c^L, c^{X-}, \pi^G, \pi^{R-}) \,. \tag{3.13}$$

Two cases are distinguished similarly to the goal policy model. In the easier case, the previously planned route π^{R-} remains valid, indicated by t^G being *false*. In that case, the traffic participant continues the same route, which is expressed by the conditional distribution

$$\sigma_-^R(\pi^R|\pi^{R-}) = \begin{cases} 1 & \text{if } \pi^R = \pi^{R-} \\ 0 & \text{otherwise} \end{cases} \,. \tag{3.14}$$

In the other case, the current route policy has reached a terminal state, which happens if the traffic participant deviates from his planned route, progresses along the planned route or by a fundamental change in situation, enforcing the traffic participant to replan his route. The transition to a new route is modeled by the distribution σ_+^R, defining the probabilities for a driver of choosing a specific route. The distribution is sparse since only routes that begin at the current lane segment and reach the current goal have probabilities greater than zero.

To define σ_+^R, we start from the premise that when a driver selects a route, he chooses among the alternatives according to some utility measure $q()$. Unfortunately, the route a traffic participant selects in a specific situation cannot be derived unambiguously. Only a distribution over possible routes can be derived. This stems from the facts that drivers not always select the optimal route and more importantly, that the model has to account for different types of drivers whose preferences may differ and are unknown. We make the further assumption that the higher the utility of a route, the higher its probability of being chosen. This connection is expressed by $p(\pi^R) \propto \phi(q(\pi^R))$ with $\phi : \mathbb{R} \to \mathbb{R}_0^+$ being some strictly increasing function describing the transformation of route utilities into unnormalized probability measures. These assumptions are conform with the assumptions made in the expected utility hypothesis (Russell et al., 1995).

We define the utility of a route on the basis of its constituents, i.e. properties of transitions between lane segments and properties of the lane segments itself. The utility of a transition is measured with the reward function $r()$. Summing up the rewards of all route elements yields the expected utility

$$q_s(\pi^R, s^R) = \sum_{l=1}^{|\pi^R|-1} r(seg(\pi^R, l), seg(\pi^R, l+1), s^R)\,. \tag{3.15}$$

The function $seg(\pi^R, l)$ returns the l'th lane segment of route π^R.

By making the probability of routes dependent on its properties in a factored way, we promote the learning of partial relationships that can be transferred to other routes that share some similarities and thereby improve the generalization of the route policy model.

The utility of transitions between lane segments take into account the type of transition as well as the properties of the next lane segment. This reflects preferences like staying on a lane over lane changes by making a transition to a successive lane segment more likely than one to an adjacent lane. Additionally, properties of the target lane segment, such as its length or its type, also influence the transition probability. These preferences can change depending on the current situation. For example, most drivers avoid overtaking maneuvers because of the increased risk of an accident when driving on an opposite lane. So the a priori probability for a route including an overtaking maneuver is quite low but if a driver is confronted with a far slower driving vehicle ahead, the probability of overtaking the obstacle becomes much higher. This situation-specific dependency is learned from data as shown in Section 4.3.2 because it is difficult to model.

In the common case that a traffic participant has progressed along the route, only routes that extend the old route are considered to cause continuity. The set of all routes whose first lane segments coincide with the remaining lane segments of the old route and which can also reach the goal is defined as

$$\begin{aligned}\mathscr{R}_{\pi^G,c^L,\pi^{R-}} = \Big\{\pi^R | \pi^R \in \mathscr{R}_{\pi^G,c^L} \wedge c^L \in \mathscr{L}_{\pi^{R-}} \wedge \\ \forall_{l=idx(\pi^{R-},c^L)}^{|\pi^{R-}|} : seg(\pi^{R-}, l) = seg(\pi^R, l - idx(\pi^{R-}, c^L) + 1)\Big\}\end{aligned} \tag{3.16}$$

with $\mathscr{L}_{\pi^R}$ being the set of all lane segments of route π^R and with $idx(\pi^R, c^L)$ indicating the index of the lane segment c^L in route π^R and $\mathscr{R}_{\pi^G,c^L}$ being the subset of all routes $\mathscr{R}$ that begin at c^L and can reach the goal area π^G.

The distribution σ_+^R captures the situation-specific selection of a new route, either because of progression along the route or because of deviating from it. It is defined as

$$\sigma_+^R(\pi^R|s^R) \quad = \quad \begin{cases} \eta_c \exp(q(\pi^R, s^R)) & \text{if } \pi^R \in \mathscr{R}_{\pi^G,c^L,\pi^{R-}} \wedge c^L \in \mathscr{L}_{\pi^{R-}} \\ \eta_r \exp(q(\pi^R, s^R)) & \text{if } \pi^R \in \mathscr{R}_{\pi^G,c^L} \wedge c^L \notin \mathscr{L}_{\pi^{R-}} \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

with normalization constants η_c, η_r and using a Gibbs measure for $\phi(x)$ to transform the expected route utilities into probabilities. In practice, the calculation of route utilities is further complicated through the finite planning horizon. We refer to Appendix A.2 for details.

Action Policy Model $\boldsymbol{P(\Pi^A|X, C^R, C^X, \Pi^R, \Pi^{A-})}$

The action policy model is the core of the probabilistic formulation of the stochastic process. It aims to resemble the decision making of traffic participants on the most fine-grained level. The model takes into account all information of the situational context to predict the next action of a traffic participant. The model is defined as

$$p(\pi^A|x_{1:n}, c^R, c^X, \pi^R, \pi^{A-}) \,. \tag{3.18}$$

The states of all traffic participants as well as their interrelations, including right-of-way or potential conflicts, all add to the prediction of the action. In case of a car, the actions are continuous controls, such as acceleration and steering, that generate the intended driving trajectory. While each traffic participant reacts on the current state of the environment individually, actions of all road users are coupled over time. Their actions affect the state of the environment and consequently, the future decisions of the other participants.

Since manually defining a consistent action policy model is difficult, error-prone and does not scale to the complexity of real world traffic, we propose to learn the model from traffic observations. We explain the learning in detail in Section 4.3.3.

Motion Model $\boldsymbol{P(X|X^-, \Pi^{A-})}$

The motion model describes the density over possible motions of a traffic participant when applying action π^{A-} in state x^-. Deterministic motion models for cars have been subject of research in control theory for a long time. There exists a large variety of models that differ in their level of detail and the effects they consider. Since good motion models already exist, there is no need to learn a model from data. We use the well known single track model, also known as one track model or bicycle model (Campion and Chung, 2008). The model describes the motion of a four wheeled car with fixed back wheels and steerable front wheels according to the non-holonomic motion constraints stemming from the kinematics of the car. Non-linear friction like slipping wheels is neglected.

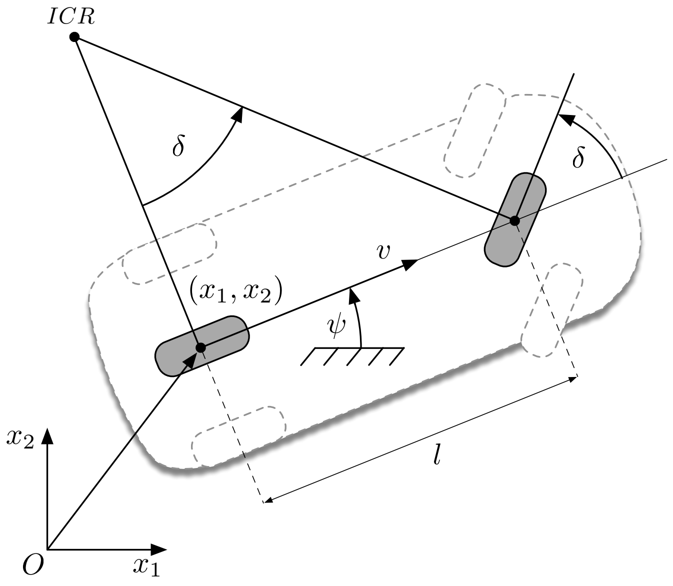


Figure 3.11: Simplified one-track model with instantaneous center of rotation (ICR).

As depicted in Figure 3.11, the four wheels are virtually replaced with two wheels placed at the center of each axis. At any point in time the car can be seen as moving on the arc of an circle around the so called instantaneous center of rotation. In the formulation of the model, we omit the steering angle since it is of no particular interest and assume that the yaw rate can be directly controlled. This is possible without loss of generality since the steering angle δ can be calculated from the yaw rate ω and vice versa if the wheelbase l is known: $\omega = v\frac{\tan\delta}{l}$. The motion of the car over time is described by the differential equations

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\psi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \omega \\ a \end{bmatrix} \tag{3.19}$$

with initial conditions x^-. The predicted state of a traffic participant is evaluated by integrating over the time interval Δt. The controls are fixed between two time steps. In practice, it suffices to use a simple Euler integration (Hairer et al., 1993) to approximately calculate the change in state. We assume no significant noise in the motion, so the motion model can be expressed as a Dirac density.

There are some physical constraints that are not expressed by the differential equations that need to be considered, e.g., that the front wheel cannot be oriented arbitrarily. These constraints effectively limit the range of yaw rates ω and accelerations a that can be applied. These additional physical constraints are addressed in detail in Appendix A.3.

Policy Termination Models $P(T^G|C^L,\Pi^{G-},T^R)$, $P(T^R|C^L,\Pi^{R-})$

The policy termination models are used to detect whether the route or goal policy has reached a terminal state. In that case, a transition in the corresponding policy model is triggered.

The current goal policy is finished as soon as the traffic participant reaches his intended goal area or ends up on a lane segment from which he cannot reach the goal any more. To account for the semi-Markov property of the policy layers, i.e., policies can be active over several time steps, the termination flag becomes *true iff* the goal policy as well as the instantiated route policy at the lower-level has reached a terminal state. Formally, the goal policy termination model is defined as

$$p(t^G = \mathit{true} \,|\, c^L, \pi^{G-}, t^R) = \begin{cases} 1 & \text{if } (c^L \in \mathscr{L}_{\pi^{G-}} \vee \mathscr{R}_{c^L,\pi^{G-}} = \emptyset) \wedge t^R = \mathit{true} \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

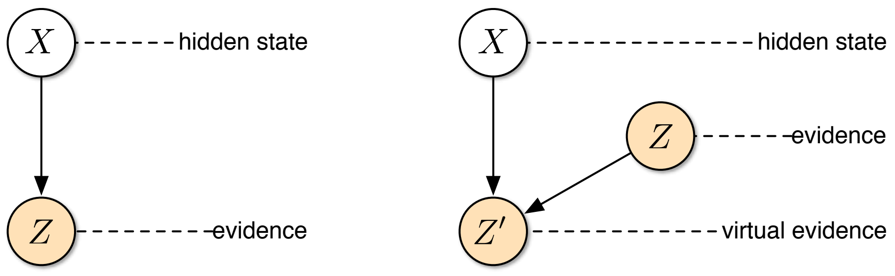
with $\mathscr{L}_{\pi^{G-}}$ being the set of all lane segments that are part of the goal area π^{G-}, and $\mathscr{R}_{c^L,\pi^{G-}}$ being the set of all routes that begin at lane segment c^L and can reach the goal area π^{G-}.

The route policy termination model is very similar to the goal policy termination model. Two cases can happen which set the termination state to *true*. In the first case, the traffic participant progresses along the route, so that the current lane segment c^L is no longer the first segment of the route. In the other case, the traffic participant has deviated from the planned route. This leads to the following definition of the route policy termination model

$$p(t^R = true \,|\, c^L, \pi^{R-}) = \begin{cases} 1 & \text{if } c^L \neq seg(\pi^{R-}, 1) \\ 0 & \text{otherwise} \end{cases} . \tag{3.21}$$

Measurement Model $P(Z|X)$

The measurement model relates the hidden states of the traffic participants with the measurements of the observable quantities. It quantifies the likelihood of receiving measurement z given the hidden state x. Both the hidden states and the measurements are continuous, which resolves in a conditional density. The measurement model has to account for the specific sensor properties, such as its noise characteristics. We are abstracting from a specific sensor type in this work and assume that the measurements are preprocessed and direct mappings of the true states with additive white noise $e_Z \sim \mathcal{N}(0, \Sigma_Z)$, i.e. normally distributed noise with zero mean.



(a) Forward model. (b) Modified model.

Figure 3.12: The measurement model is modified (b) to handle the periodicity of the angular orientation instead of a regular forward model (a).

Due to the periodicity of the angular orientation, the measurement model cannot be correctly modeled as a linear Gaussian forward model $p(z|x)$ (Kurz et al., 2013). To formulate the relationship between states and measurements, we use a modeling technique which introduces an additional random variable together with a virtual evidence. This technique is commonly used to formulate (soft) constraints and correlations in Bayesian networks (Pearl, 1988). The additional helper random variable Z' represents the difference between X and

Z and is connected as depicted in Figure 3.12. The corresponding model is defined as

$$p(z'|x,z) \sim \mathcal{N}(d(x,z), \Sigma_Z) \tag{3.22}$$

with

$$d(x,z) = \begin{bmatrix} x_{x_1} - z_{x_1} \\ x_{x_2} - z_{x_2} \\ dangle(x_\psi, z_\psi) \\ x_v - z_v \\ x_w - z_w \\ x_l - z_l \end{bmatrix}. \tag{3.23}$$

The function *dangle*(ψ^x, ψ^z) yields the minimal difference angle between ψ^x and ψ^z in the interval $[-\pi, \pi)$. The evidence generated by an observation then consists of the measured evidence on Z and a zero valued evidence on Z', weighting the difference between the hidden state x and the measured state z.

For the measurement model, it is assumed that the traffic participants and their corresponding measurements can be uniquely identified. This can be achieved in the preprocessing stage of the perception with data association methods. See (Blackman and Popoli, 1999) and (Cox, 1993) for further reading.

With the Bayesian model now fully defined, we turn to probabilistic inference in the next section to show how the model can used for reasoning.

3.4 Inference in the Bayesian Model

Having a probabilistic model of the development of traffic situations allows reasoning about the distributions over current, past and future states based on the acquired measurements. The properties of interest may vary between different applications but there are two cases of special importance, namely filtering and predicting.

Filtering is concerned with estimating the current belief state, i.e., the distribution over the traffic situations, given all measurements $z^{(1:t)}$ up to the current point in time. The posterior distribution is derived from (3.4) using Bayes' rule

$$\begin{aligned} b^{(t)} &= p(s^{(t)}|z^{(1:t)}) \\ &= \frac{p(z^{(t)}|s^{(t)}, z^{(1:t-1)})p(s^{(t)}|z^{(1:t-1)})}{p(z^{(t)}|z^{(1:t-1)})} \\ &= \eta^{(t)} p(z^{(t)}|S^{(t)}) \int p(s^{(t)}|s^{(t-1)}) \underbrace{p(s^{(t-1)}|z^{(1:t-1)})}_{\text{recursive term } b^{(t-1)}} ds^{(t-1)} . \end{aligned} \tag{3.24}$$

This yields the recursive Bayesian filter formula for estimating the current belief by marginalizing over the prior belief states and incorporating the current observation. The normalization constant $\eta^{(t)} = p(z^{(t)}|z^{(1:t-1)})^{-1}$ ensures that the resulting belief is a proper probability distribution. The current belief $b^{(t)}$ subsumes all knowledge available about the current situation, i.e., all previous measurements and prior knowledge in terms of predictive models.

The full belief update step consists of two sub steps: Predict and correct. The prediction step computes the one-step-ahead prediction $p(s^{(t)}|z^{(1:t-1)})$. The correct step incorporates the current measurement and yields $p(s^{(t)}|z^{(1:t)})$. Due to the Markov property of the process, the prior belief (and all previous measurements) can be neglected once the current belief is updated. This yields an update procedure with constant time and space complexity (Murphy, 2012).

Predicition In addition to knowing the current state of a situation, one is often interested in predicting the future states. Given the current belief, one is interested in the distribution over situations h steps in the future

$$p(s^{(t+h)}|z^{(1:t)}) = \int \prod_{i=1}^{h} p(s^{(t+i)}|s^{(t+i-1)}) p(s^{(t)}|z^{(1:t)}) d s^{(t:t+h-1)} . \tag{3.25}$$

The procedure is very similar to filtering with the difference that correction steps are neglected after time step t and only predictions steps are applied from this time step on.

The practical issues that arise when trying to carry out the filtering and prediction tasks are: How can the integration be solved and how can distributions be represented. The problem is that the integral cannot be solved analytically in the general case (Koller and Friedman, 2009), i.e., for arbitrary distributions and non-linear transition and measurement models.

Sequential Monte Carlo inference Since the presented Bayesian model uses a mixed state space and non-linear models, exact inference is not possible. We propose applying approximate inference from the class of sequential Monte Carlo (SMC) methods also known as particle filters (Doucet et al., 2001). To be more specific, we apply likelihood weighting (LW) (Koller and Friedman, 2009; Fung and Chang, 1989), which is an instance of importance sampling (MacKay, 2003).

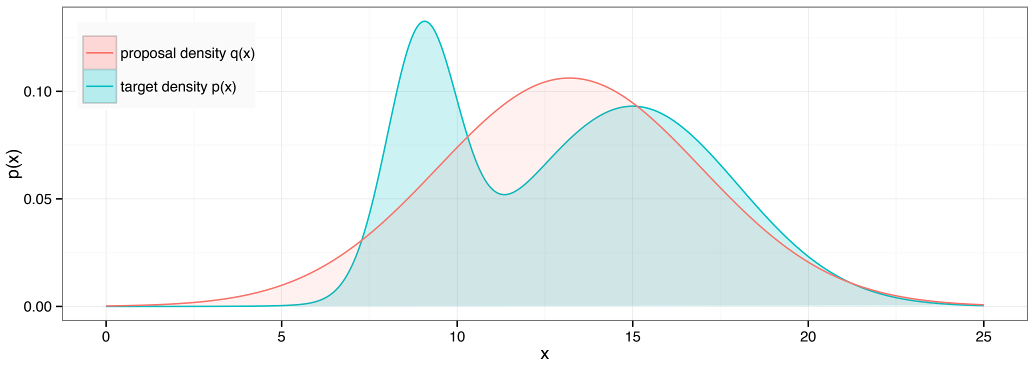


Figure 3.13: Example of importance sampling: Samples are drawn from a Gaussian proposal density $q(x)$ and weighted according to the ratio of the target density $p(x)$ and the proposal. The weighting accommodates for the fact that samples from areas where $p(x) < q(x)$ holds are over represented and the other way around in the sample set drawn from $q(x)$.

Importance sampling can be applied when samples from a target distribution $p(x)$ are needed but sampling directly from this distribution is not an option because it is not possible or computationally too demanding. So instead of using $p(x)$ directly, one draws samples from a simpler but related distribution called the proposal distribution $q(x)$. The samples are then weighted with $w \propto \frac{p(x)}{q(x)}$ according to the relation between the target and the proposal distribution to accommodate for the fact that the samples were drawn from a surrogate distribution and thereby ensuring the correct expectation of the estimator. The result is a set of weighted samples $\mathscr{M} = \{\langle x^{(i)}, w^{(i)}\rangle\}_{i=1}^{m}$, which are often called particles, that approximates the target distribution through a Dirac mixture

$$p(x) \approx \sum_{i=1}^{|\mathscr{M}|} w^{(i)} \delta(x, x^{(i)}) \tag{3.26}$$

with $\sum_{i=1}^{|\mathscr{M}|} w^{(i)} = 1$. Figure 3.13 shows an example of importance sampling with a Gaussian proposal density.

An importance sampling scheme is completely defined by a target distribution and a suitable proposal distribution where $\forall x : p(x) > 0 \implies q(x) > 0$. Both the target and the proposal distribution must be evaluable at any chosen point x, at least within a multiplicative constant. Likelihood weighting defines the proposal distribution as a mutilated version of the target Bayesian network distribution without the evidence nodes. Applying this to the task of filtering, yields the following recursive update procedure: Every belief $b^{(t)}$ is approximated through a set of weighted particles $\mathscr{M}^{(t)} = \{\langle s^{(t)(i)}, w^{(t)(i)}\rangle\}_{i=1}^{m}$. In the LW prediction step, the predicted belief samples

$$\{\hat{s}^{(t)(i)}\}_{i=1}^{m} \sim p(s^{(t)}|z^{(1:t-1)}) = \int \underbrace{p(s^{(t)}|s^{(t-1)})}_{\text{transition model}} \underbrace{p(s^{(t-1)}|z^{(1:t-1)})}_{\text{prior belief } b^{(t-1)}} ds^{(t-1)} \tag{3.27}$$

are drawn according to the prior belief and the transition model. By first sampling from the prior belief distribution before applying forward sampling according to the transition model, one effectively performs resampling, which prevents degeneration of the particle weights (Thrun et al., 2005). Techniques like low variance sampling (Thrun et al., 2005) can be applied at this point to improve the variance of the sample set in some cases. The resampling causes the weights of the predicted samples to be uniformly distributed $\hat{w}^{(t)(i)} \equiv \frac{1}{m}$.

In the LW correct step, the predicted samples are weighted according to the importance weighting scheme using the measurement model and the current observation. Since the proposal distribution $p(s^{(t)}|z^{(1:t-1)})$ only differs from the target distribution $p(s^{(t)}|z^{(1:t)})$ by the measurement, the weighting simplifies to (using 3.24 and 3.27)

$$w^{(t)(i)} \propto \frac{p(\hat{s}^{(t)(i)}|z^{(1:t)})}{p(\hat{s}^{(t)(i)}|z^{(1:t-1)})} \propto p(z|\hat{s}^{(t)(i)}) \,. \tag{3.28}$$

The resulting belief distribution is then $\mathscr{M}^{(t)} = \{\langle s^{(t)(i)}, w^{(t)(i)}\rangle\}_{i=1}^{m}$ with $s^{(t)(i)} \equiv \hat{s}^{(t)(i)}$ and $\sum_{i=1}^{m} w^{(t)(i)} = 1$.

The use of the one-step prediction $p(s^{(t)}|z^{(1:t-1)})$ as the proposal distribution for $p(s^{(t)}|z^{(1:t)})$ has several implications. The proposal will generate samples from the prior belief according to the transition model, reflecting the systems belief about the current situation's state. If the received measurement does not match the expectation, the particle weights get very small according to 3.28, i.e., samples from the true distribution are not well represented by the particle set. In order to get a reasonable estimate, one may need to draw a lot of samples, which consequently leads to an increased computational effort. For more information on this topic, we refer to (Koller and Friedman, 2009). For a general discussion on the performance of importance sampling schemes in Bayesian networks see (Yuan and Druzdzel, 2006).

Another factor that influences the amount of particles needed to approximate the target distributions is the uncertainty present in the process. An important finding of (MacKay, 2003) is that the variance of a Monte Carlo expectation estimate of a function $\phi(x)$ under a distribution $p(x)$ only depends on the variance of ϕ and not on the dimensionality of the space sampled. This implicates that the less uncertainty is present in a process, the less particles are needed to represent the belief distributions. This coherence is exploited in (Fox, 2001) to dynamically adapt the sample set size in particle filters. There are two ways to reduce the uncertainty in a process. One is reducing the noise in the measurement model by using better sensors and the other is making better predictions. In this work, we aim for making better predictions by using data-driven methods. Every improvement in prediction accuracy reduces the variance of the resulting belief estimates and thereby the number of required particles.

Important features of SMC methods are, that they can cope with non-linear models, represent multi-modal belief distributions, can handle mixed state spaces and can satisfy real-time constraints. These features make them well suited for inference tasks in the Bayesian model presented in this work and the intended domains of application. In practice, SMC methods have been applied quite successfully to a whole range of real world problems, e.g., in the field of robotics (Thrun, 2001). Other methods make stronger assumptions about the underlying process, e.g., an extended Kalman filter (Thrun, 2001) can use non-linear models through linearization but is only able to represent uni-modal beliefs and also does not work with mixed state spaces. The ability to represent multi-modal distributions becomes especially important when making long term predictions of traffic situations. If for example, the lane splits up into several ways in an intersection, multiple modes are needed. Another source for multiple modes are the decisions of other traffic participants. Depending on their choices, a situation can develop in different directions. Other options of inference methods can be found in (Guo and Hsu, 2002; Koller and Friedman, 2009; Murphy, 2002).

3.5 Planning using Predictive Models

In the context of autonomous driving or ADAS, planning is needed on various levels of abstraction. These can be coarsely divided into route planning, behavior decision making, motion planning and control. On the lowest level, the subject of control is to realize intended driving trajectories by providing suitable inputs to the car system. On the next level, motion planning aims to find the best trajectory to a target location under temporal, spatial and physical constraints. Figure 3.14 shows an example where motion planning is used to find a suitable trajectory for merging into moving traffic. On a more abstract level, behavior decision making is used to find the best maneuver in a specific situation according to an objective function. The planning horizon on this level is longer compared to the level of motion planning, making it possible to take long term effects of decisions into account but at the cost of considering less detail. Applied to the situation depicted in Figure 3.14, the decision could be between choosing to merge in front of the approaching car or to slow down and wait for the next gap to merge. On the highest abstraction level, route planning is concerned with finding the optimal route to a target location.

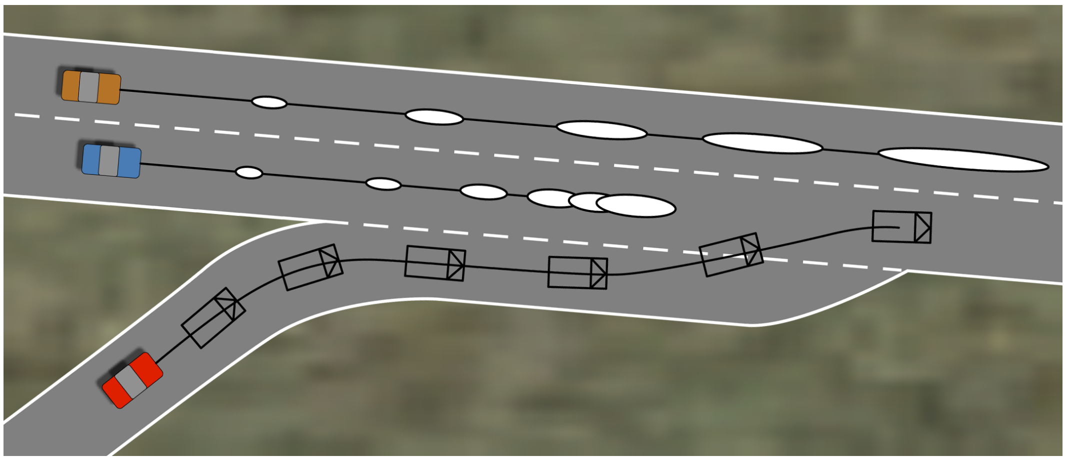


Figure 3.14: Motion planning for a merging maneuver that considers the reaction of other traffic participants on the behavior of the ego vehicle (red).

In all cases, the consequences of decisions have to be anticipated and evaluated in order to derive the optimal decisions via planning. The presented Bayesian model can be used to obtain the necessary predictions of how situations will develop. This is especially useful in the context of motion planning and behavior decision making. The advantage of this approach is that the model makes accurate predictions by considering how traffic participants affect each others decisions. Control can benefit from the Bayesian model when using model predictive control techniques (Camacho and Alba, 2013) but control frequency requirements may prohibit its application. Since route planning takes a macroscopic view on traffic and the planning problem, it cannot directly profit from the Bayesian model.

Sequential decision making in stochastic domains can be modelled with a Markov decision process (MDP). MDPs model time discrete stochastic processes that can be influenced via actions in each time step. State transitions are described with a transition model in form of a conditional distribution. A reward function defines the objectives of an agent in terms of scalar rewards and punishments, which the agents receives when conducting specific actions in certain states. Solving an MDP means finding a policy that maximizes the received reward over time. In the context of autonomous driving and ADAS, the reward function can be defined to balance safety, efficiency, comfort, compliance and progress. In MDPs, the process state is assumed to be fully observable

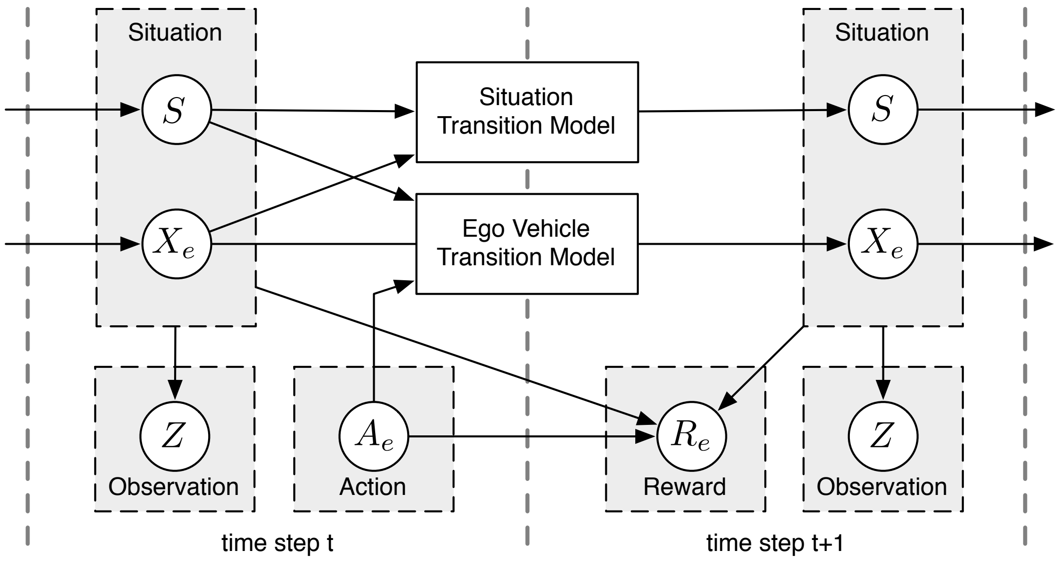


Figure 3.15: Embedding of the Bayesian model into a partially observable Markov decision process. By modeling one traffic participant X_e to be directly controllable through actions A_e and using the presented Bayesian model for the other traffic participant, the transitions between situations can be calculated. In case that S and X_e are fully observable, one obtains an embedding for a Markov decision process.

at any point in time, a property which is not met in the traffic domain. MDPs can be extended to partially observable Markov decision processs (POMDPs) to become applicable in such domains. In POMDPs, a measurement model describes the relationship between states and observations. Since the process state cannot be determined with full certainty, the policy is defined on the belief space instead of the state space like in MDPs. The belief is a distribution over the state space, describing the agents subjective belief about the state of the world (3.4).

To model the decision making of an autonomous car with a (PO)MDP the Bayesian model presented in this chapter can be used to describe and anticipate the reactions of the environment on the actions of the autonomous car. The *ego vehicle* is directly controlled through actions A_e and represented by state X_e. The other traffic participants as well as their state transitions are covered by the presented Bayesian model. Figure 3.15 depicts the embedding in the decision process. A situation in this setting consists of the states of the ego vehicle x_e together with the states of the other traffic participants summarized by s (3.2).

In the following, we outline the necessary extensions and modifications to the Bayesian model to complete the (PO)MDP model.

Ego vehicle transition model $\boldsymbol{P(X_e|A_e^-, X_e^-, S^-)}$ A transition model has to be defined for the set of actions the ego vehicle can conduct, describing the possible state transitions as consequences of the conducted action as a conditional distribution. Dependencies on the overall situation have to be considered.

Situation transition model $\boldsymbol{P(S|S^-, X_e^-)}$ The stochastic transitions of the other traffic participants are described by the Bayesian model (3.7). Since the ego vehicle is part of their context, its state is considered in their decision making as they react to it. Taking this mutual influence into account is of great importance for deriving the right conclusions in the traffic domain, particularly in difficult situations, like merging into moving traffic.

Reward function $\boldsymbol{R(S^-, X_e^-, A_e^-, S, X_e)}$ The reward function quantifies the objectives of the ego vehicle. It is a scalar function that simultaneously combines several criteria by summing them up. The function depends on the previous environment state, the conducted action and the resulting situation. The objectives of an autonomous car can be encoded by assigning positive rewards for reaching a target destination and negative rewards for crashing. Secondary objectives like compliance with traffic regulations, driving comfortably and economically can also be incorporated by assigning corresponding rewards, e.g., for fuel consumption. The magnitudes of the rewards quantify their importance and hence determine, for example, the risk aversion of the autonomous car.

Measurement model $\boldsymbol{P(Z|S, X_e)}$ The measurement model accounts for the sensors' inaccuracies. In addition to the model described in Section 3.3.3, the measurement model needs to be extended to account for more complex sensor limitations such as occlusions in order to obtain policies with information gain strategies. For sensors like lidar sensors and cameras, occlusion can be modeled by checking lines of sights between traffic participants. Similar measurement models also exist in the field of localization and mapping (Hähnel et al., 2003;

Hoffman et al., 2005; Hester and Stone, 2008). With such measurement models, policies can be derived where the ego vehicle actively considers its lack of information in its action choices. For instance, it may not not overtake another vehicle because it cannot see behind it. In ambiguous situations it performs information gain to reduce risks, e.g., such as slowly approaching an intersection that is occluded by obstacles.

MDP evaluation and discussion In a collaborative work (Brechtel et al., 2011), we embedded a simpler version of the Bayesian model into an MDP and showed how optimal decisions for an autonomous car can be automatically derived. Solving the MDP is made feasible by transferring the continuous models to an adaptively growing discrete space and applying a combination of online and offline planning. Figure 3.16 shows an example of the behavior resulting from the planned policy in a highway scenario. The policy showed an excellent performance in the evaluated simulations but heavily relies on the MDP assumption that the environment state is fully observable. This assumption cannot be satisfied in real world traffic situations. A key insight of the experimental results is that realistic prediction models are mandatory for obtaining useful policies.

In reality, measurements are limited by the range of sensors, sensor noise and occlusions caused by other objects, e.g., other vehicles or buildings. We therefore expect future decision making systems in the automotive field to take into account the partial observability. First works in this direction are from (Brechtel et al., 2013; Ulbrich and Maurer, 2013; Bai et al., 2013; Forbes et al., 1995).

Advances in solution and approximation techniques allow solving POMDPs with discrete spaces and a manageable amount of states in reasonable time (Shani et al., 2013; Kurniawati et al., 2008; Spaan and Vlassis, 2005). Among the reported problems that have been solved are some with several hundreds of states (Poupart et al., 2011). However, most real word problems can be more accurately described with continuous state and observation spaces which makes the solution finding more complex (Porta et al., 2006). Solution techniques for continuous POMDPs are a field of active research. With current state-of-the-art methods only low-dimensional continuous POMDPs can be solved in reasonable time (van den Berg et al., 2012; Zhou et al., 2010; Bai et al., 2011; Porta et al., 2006). We expect future advances in this field to be able to fully exploit the benefits arising from sophisticated prediction models with continuous state spaces.

As a step in this direction, we presented a new value iteration approach for solving POMDPs with continuous state spaces in the collaborative work with (Brechtel et al., 2013). It allows solving POMDPs that use Bayesian models like the one presented directly without the need of prior discretization. The method employs machine learning methods to automatically find a symbolic embedding of the continuous state space. The detection and exploitation of the hidden problem structure reduces the computational complexity and makes it possible to tackle higher dimensional problems like navigating at intersections. More details on the approach as well as its application to autonomous driving can be found in the paper and (Brechtel et al., 2014).

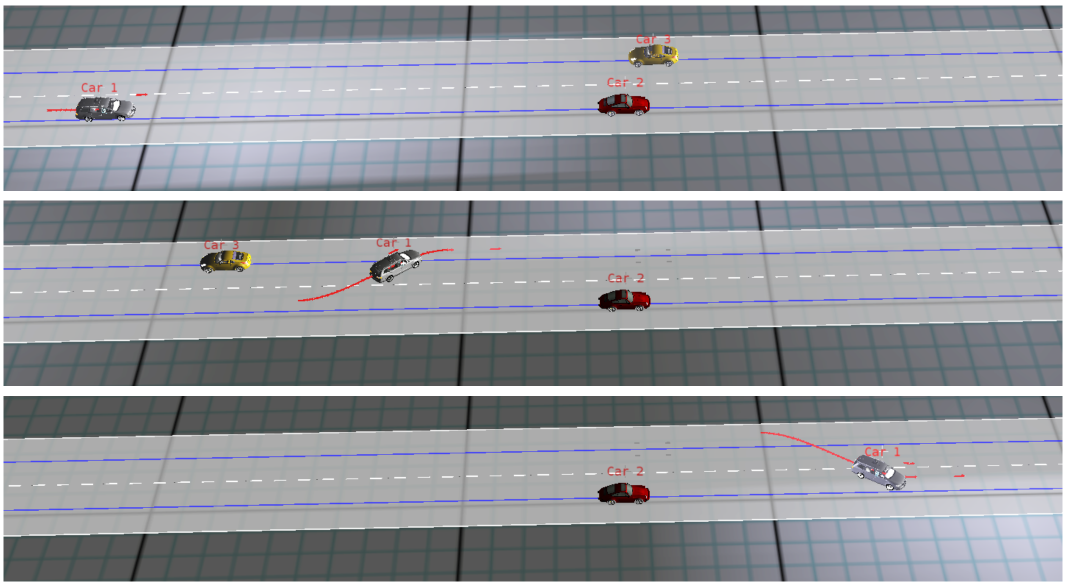


Figure 3.16: Simulation of an overtaking scenario. The policy of the ego vehicle (grey) is derived from solving a corresponding MDP. The planned trajectory is indicated by the red line. (Brechtel et al., 2011).

3.6 Summary and Conclusion

In this chapter, we presented a Bayesian model that describes the evolution of traffic situations as a stochastic process. Derived from a novel general hierarchical model for policy recognition and prediction making in multi-agent environments, we developed a model for traffic scenarios that addresses and exploits the specific characteristics of the domain. The key idea was to resemble the decision making of traffic participants and explicitly consider their interactions.

The presented model goes beyond the state-of-the-art in several ways. It describes the evolution of complete traffic situations in a unified statistical model and thereby combines several levels of abstraction. This allows fine grained predictions on the continuous dynamics level as well as long term predictions on the level of routes and goals. In contrast to approaches that only consider a specific abstraction level, the hierarchical approach can yield more accurate predictions since the flow of information between the different layers improves the predictions at each level of detail. By combining domain specific knowledge with learned models for dependencies that are not well understood and difficult to model manually, a prediction accuracy and generalization to new situations of yet unseen quality can be achieved.

One of the most promising areas of application for such a model is decision making. We outlined how the Bayesian model can be incorporated in a (PO)MDP to anticipate the consequences of decisions and derive optimal policies for autonomous traffic participants.

4 Policy Model Learning from Observations

We present parametric and non-parametric models for representing the policy models of the Bayesian model together with corresponding estimation techniques for learning these conditional distributions from observations.

4.1 Approach

The key components of the Bayesian model presented in the last chapter are the policy models. They resemble the decision making process of the traffic participants and model the dependencies between situations and resulting actions. Due to the manifold of different traffic situations that can occur, the derivation of these models are the most challenging part in the statistical modeling of the dynamic process. We pursue the approach to learn these models from data and support the learning process by providing the information necessary to find the dependencies between specific situational aspects and their implications on the decision making.

While in principle the models can be defined manually, there are several reasons that militate against this approach. There are that many variations of traffic situations that it is difficult for a human expert to formulate a model that covers all of them. The second problem is that the dependencies are partly unknown and often only available in an implicit form. This makes the modeling difficult and error prone. The encoding of traffic rules is a good start but only helps up to a certain degree since in reality, they are not always strictly followed and there is room for interpretation. For example, in some situations a traffic participant may cross a solid white line in order to prevent a severe accident. One of the main difficulties for humans is the quantification of conditional densities. An approach to solve this difficulty can be found in the collaborative work (Gindele et al., 2010) where behaviors of traffic participants are predicted based on manually defined models in the domain of highway scenarios. All the named reasons limit the scalability, accuracy and generality of approaches with solely manually specified models.

Using a data driven approach to derive the models, on the other hand, does not suffer from these limitations. The idea is to derive the models of interest with machine learning methods by observing the behavior of traffic participants in all kinds of situations and identify the characteristic patterns. Since the goal of learning in this context is finding models that best explain the data, it is ensured that the resulting models are consistent with reality under the prior assumptions. Hence, in contrast to manually specified models, learned models are based on what the system actually observes and not on what an expert thinks how traffic participants behave.

Statistical learning theory teaches that using more data can improve the quality of the results (Vapnik, 2000). With enough data, even subtle nuances of the relationship of interest can theoretically be learned. Since manual labeling of data is costly, it practically limits the amount of data available for learning. Thus, we introduce an algorithm that does not require manual labeling. Unlabeled data can be obtained at low cost in the traffic domain by observing and recording traffic, e.g., from sensor-equipped cars or using traffic surveillance systems. The machine learning algorithms we employ belongs to the class of inductive learning methods, allowing to generalize from the training examples and making suitable predictions in similar situations that have never been encountered before. By adding new observations to the data, the models can be easily extended to cover whole new situations.

In this chapter, we derive a learning algorithm for learning the policy models of the presented Bayesian model. Since the data needed to learn the models is not directly observable, we use an expectation maximization scheme to solve the maximum likelihood estimation for the incomplete data problem. We first give an overview over the expectation maximization framework (4.2.1) and suitable specializations (4.2.2) before we derive the concrete algorithm to solve the task at hand (4.3). We provide parametric and non-parametric models and learning algorithms to solve the conditional density estimation problems that occur as substeps of the overall learning algorithm (sections 4.3.1 to 4.3.3). The model learning tasks range from estimating discrete dependencies as in the case of the goal policy model to discovering complex and highly non-linear dependencies as in the case of the action policy model.

4.2 Model Learning with Generalized Expectation Maximization

The goal of learning the policy models is to find the set of models that best predict the evolution of traffic situations. One factor that makes the learning of the policy models challenging is that one cannot observe what other traffic participants are thinking. The current goals, planned routes and executed actions of other road users are not observable. Only the basic states of traffic participants such as their positions, velocities and orientations can be measured with noise. So the dependencies between situations and the resulting decisions of traffic participants which are represented by the policy models cannot be learned directly since the true values are unknown. Figure 4.1 illustrates the problem of learning the policy models from incomplete data for the Bayesian model. The policy models are identified through their parameters $\theta = (\theta_G, \theta_R, \theta_A)$ with θ_G being the parameters of the goal policy model, θ_R the parameters of the route policy model and θ_A the parameters of the action policy model.

4.2.1 Expectation Maximization Framework

A general class of optimization techniques that can deal with incomplete data problems is known under the name expectation maximization (EM) (Dempster et al., 1977; Meng and Van Dyk, 1997; McLachlan and Krishnan, 2007). EM can provide maximum likelihood (ML) and maximum a posteriori (MAP) estimates for generative models in cases where not all aspects of the model are observable. EM exploits the fact that if all values were fully observable, ML/MAP estimates would be simpler to compute. Starting from an initial parameter guess, the EM algorithm finds a (locally) optimal ML/MAP estimate by iterating between an expectation step (E-step) and a maximization step (M-step). The E-step calculates expectations (distributions) over the hidden variables based on the current model parameters to obtain an estimate of the complete data. With the full data available, the model parameters are maximized (optimized) in the M-step. These steps are repeated until convergence is reached.

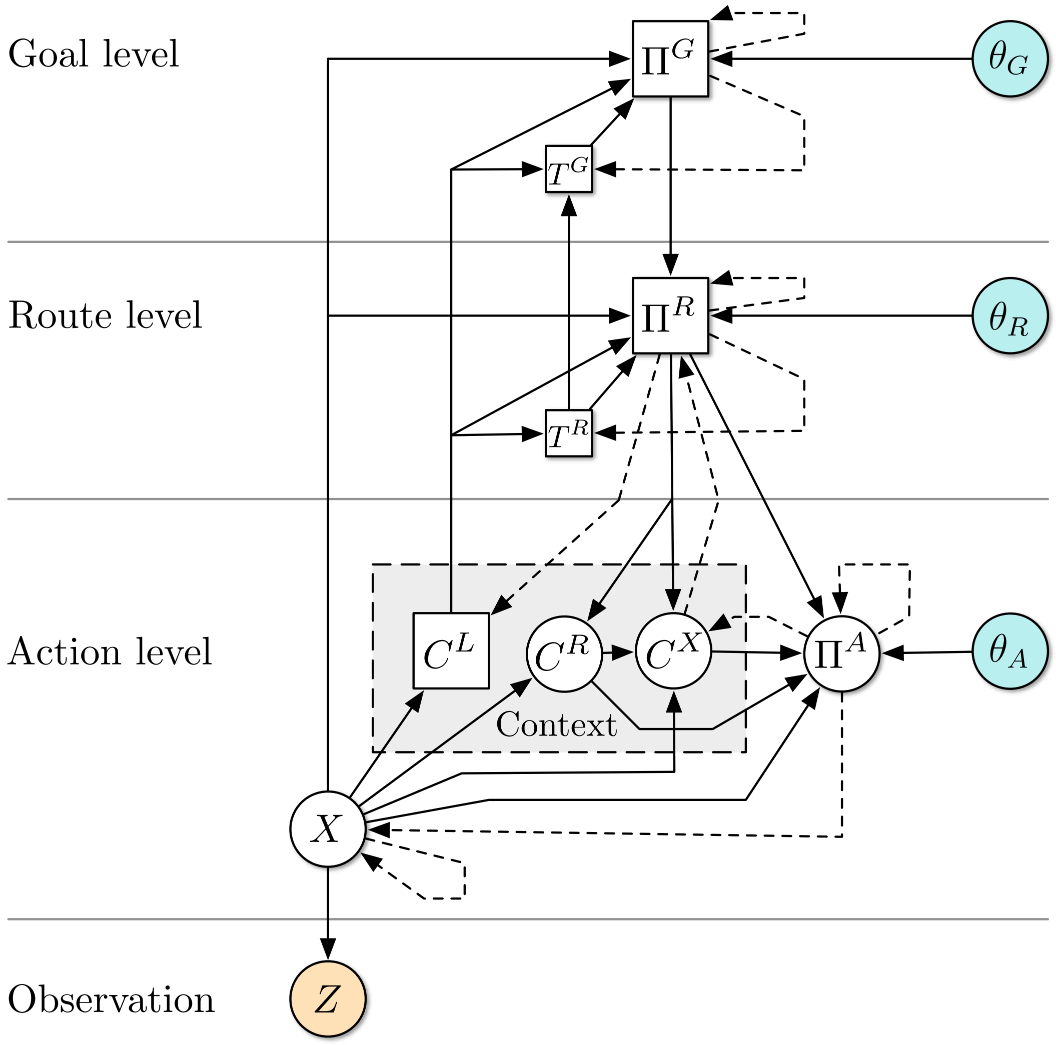


Figure 4.1: Visualization of the incomplete data learning problem. Only the Z nodes representing the measurements are observable. The state of all other random variables comprising a situation can only be inferred with uncertainty. The learning problem is to find the model parameters of the policy models θ_G, θ_R and θ_A that best explain the data.

Let X be the set of hidden variables and Z be the set of observable variables. $P(Z, X|\theta)$ denotes the joint distribution over the complete data dependent on the model parameters θ. The observations are given as a set $\mathscr{D} = \{z_i\}_{i=1}^N$ of independent observations. The goal of ML estimation in this setting is to maximize the log likelihood of the observed data

$$l(\theta) = \mathop{\mathbb{E}}_{z_i} \log p(z_i|\theta) = \sum_{i=1}^{N} \log \int_x p(z_i, x|\theta) dx \; . \tag{4.1}$$

Since $l(\theta)$ is hard to optimize directly, due to the log of the integral, EM optimizes the expected complete data log likelihood instead

$$\begin{aligned} Q(\theta|\theta^{(t-1)}) &= \mathop{\mathbb{E}}_{z_i} \mathop{\mathbb{E}}_{p(x_i|z_i,\theta^{(t-1)})} \big[\log p(z_i, x_i|\theta)\big] \\ &= \sum_{i=1}^{N} \int_{x_i} p(x_i|z_i, \theta^{(t-1)}) \log p(z_i, x_i|\theta) dx_i \; . \end{aligned} \tag{4.2}$$

This function is often called the *auxiliary function* (Murphy, 2012). As a result of Jensen's inequality, Dempster et al. showed that improving $Q(\theta|\theta^{(t-1)})$ also improves the true likelihood $l(\theta)$ or leaves it unchanged (Dempster et al., 1977).

The EM algorithm therefore iterates between two steps. The E-step calculates the distributions over the hidden variables X dependent on the observed values z_i and the parameter estimate $\theta^{(t-1)}$ from the last iteration $t-1$:

$$\text{E-step:} \qquad \text{calculate } p(x_i|z_i, \theta^{(t-1)}) \, . \tag{4.3}$$

The M-step then uses the estimated distributions over the hidden variables to maximize the expected complete data log likelihood and obtain an improved parameter estimate for the next iteration:

$$\text{M-step:} \qquad \theta^{(t)} = \mathop{\mathrm{argmax}}_{\theta} Q(\theta|\theta^{(t-1)}) \, . \tag{4.4}$$

For most models, the sequence of parameter estimates $\theta^{(t)}$ converges to a local optimum (or saddle point) of the $l(\theta)$ under regularity conditions (Little and Rubin, 2002; Wu, 1983; Dempster et al., 1977). In some cases even the global optimum is guaranteed to be found (Wu, 1983). These convergence properties even hold if $\theta^{(t)}$ is only improved in each M-step instead of maximized (Neal and Hinton, 1998), e.g., by a gradient ascent step. These variants are referred to as generalized expectation maximization (GEM) algorithms. The fact that it suffices to improve θ in the M-step will become important in Section 4.3, when we exploit it to show that the models can learned separately.

(Neal and Hinton, 1998) showed that EM can be interpreted as maximizing a joint function of the parameters and of the distribution over the unobserved variables, where the E- and M-step partially optimize the joint function. This view justifies incremental, sparse, and other variants of the EM algorithm.

The EM algorithm can also serve to find a MAP estimate by adding a prior $p(\theta)$ over the parameter space. In this case, the M-step (4.4) is replaced by

$$\theta^{(t)} = \underset{\theta}{\operatorname{argmax}} \big(Q(\theta|\theta^{(t-1)}) + \log p(\theta) \big) \; . \tag{4.5}$$

4.2.2 Monte Carlo Expectation Maximization

The analytical calculation of the E-step poses a problem for most models since a closed-form solution of the involved integral

$$p(x_i|z_i, \theta^{(t-1)}) = \frac{p(x_i, z_i|\theta^{(t-1)})}{\int p(x_j, z_i|\theta^{(t-1)})\, d x_j} \tag{4.6}$$

rarely exists. There are exceptions to this, e.g., for some cases where the complete data distribution is a member of the exponential family (Booth and Hobert, 1999). Unfortunately, the presented model in this work does not fall under this class, which is why we propose to use an approximate inference technique based on Monte Carlo simulation in correspondence to the Monte Carlo inference presented in Section 3.4.

Research in the area of learning the parameters of Bayesian networks and Markov random fields (aka Markov networks) has led to a number of approximations that share the common idea of approximating the integrals that arise in the EM framework by Monte Carlo estimates. One of these developments is the Monte Carlo expectation maximization (MCEM) (Wei and Tanner, 1990), which uses samples from the distributions $p(x_i|z_i, \theta^{(t-1)})$ to approximate the integrals in the E- and M-step. The distribution over the hidden variables derived in the E-step is expressed by a Dirac mixture of M drawn samples x_j

$$p(x_i|z_i,\theta^{(t-1)}) \approx \frac{1}{M}\sum_{j=1}^{M}\delta(x_i,x_j)\,. \tag{4.7}$$

Plugging 4.7 into (4.2) yields the approximated version of the auxiliary function

$$Q_{MC}(\theta|\theta^{(t-1)}) = \frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M}\log p(z_i,x_j|\theta) \tag{4.8}$$

with the integral being replaced by a finite sum. Q_{MC} is then maximized in the adapted M-step

$$\theta^{(t)} = \underset{\theta}{\operatorname{argmax}}\, Q_{MC}(\theta|\theta^{(t-1)})\,. \tag{4.9}$$

While this approximation eliminated of the integral, one problem still remains: How to obtain samples from $p(x_i|z_i,\theta^{(t-1)})$? Several solutions have been proposed, including the popular suggesting to use Markov chain Monte Carlo (MCMC) algorithms like Gibbs sampling or Metropolis Hastings (Robert and Casella, 2004; Ibrahim et al., 2004). Using MCMC algorithms leads to the well known issues of slow state space explorations if the dimensions of the state space are correlated (MacKay, 2003). Additionally, their use in the scope of the EM framework is computationally expensive since in every iteration the Markov chain has to converge again before it yields samples from the distribution of interest. Also the error induced by Monte Carlo estimates is hard to quantify for MCMC methods, as pointed out by (Levine and Casella, 2001). Other approaches employ rejection sampling, which becomes inefficient in high dimensional spaces (Booth and Hobert, 1999). These properties render the named approaches unsuitable for performing the E-step in the considered case, instead, we employ another class of MCEM algorithms that utilizes importance sampling.

Importance sampling MCEM In the importance sampling variant (IS-MCEM) of MCEM as presented by (Booth and Hobert, 1999), the samples in the E-step are not directly drawn from the target distribution but instead from a proposal distribution $q(x_i|z_i, \theta^{(t-1)})$ (see Section 3.4 for details on importance sampling). The Monte Carlo estimate is then given by

$$p(x_i|z_i, \theta^{(t-1)}) \approx \sum_{j=1}^{M} w_{i,j}\, \delta(x_i, x_j) \tag{4.10}$$

with importance weights

$$w_{i,j} \propto \frac{p(x_i|z_i, \theta^{(t-1)})}{q(x_i|z_i, \theta^{(t-1)})} \tag{4.11}$$

and normalized to 1. Accordingly, the importance sampling approximation of the auxiliary function (4.2) is

$$Q_{IS}(\theta|\theta^{(t-1)}) = \sum_{i=1}^{N}\sum_{j=1}^{M} w_{i,j} \log p(z_i, x_j|\theta)\,. \tag{4.12}$$

Analogously, the M-step maximizes the approximated auxiliary function Q_{IS} with respect to the parameters, which yields

$$\theta^{(t)} = \operatorname*{argmax}_{\theta} Q_{IS}(\theta|\theta^{(t-1)})\,. \tag{4.13}$$

Gradient-based M-step The M-step for the route policy model as well as the goal policy model is realized through a gradient-based optimization. A partial maximization, which is already sufficient for the EM to converge under regularity conditions, can be obtained with gradient ascent methods (see Appendix A.4). To define a gradient-based optimization, the gradient of the objective of interest has to be derived. In this case, the objective function is $Q_{IS}(\theta|\theta^{(t-1)})$ with the following partial derivatives

$$\frac{\partial Q_{IS}(\theta|\theta^{(t-1)})}{\partial\theta} = \sum_{i=1}^{N}\sum_{j=1}^{M} w_{i,j} \frac{\partial \log p(z_i, x_j|\theta)}{\partial\theta} \tag{4.14}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} w_{i,j} \frac{\frac{\partial p(z_i, x_j|\theta)}{\partial\theta}}{p(z_i, x_j|\theta)}\,. \tag{4.15}$$

As can be seen, the key element for gradient-based optimization of the expected complete data log likelihood are the derivatives of the joint density with respect to the parameters. We derive these derivatives for the used parametric models in later sections (4.3.1, 4.3.2).

Incremental EM variants In practice, stochastic approximations of the EM can be used to achieve faster convergence rates. Popular examples are stochastic EM (SEM) (Celeux and Diebolt, 1985; Gilks et al., 1996) and stochastic approximate EM (SAEM) (Delyon et al., 1999). They calculate the E-step only partially and reuse the estimated statistics of previous iterations to reduce the computational costs of the initial iterations. It is proven that these approximations also converge to local maxima under mild regularity conditions.

4.3 Learning the Models

In the general presentation of the EM framework, the Bayesian network was considered static so far and all the model parameters were summarized by a parameter vector θ. In order to learn the models of the Bayesian model presented in the last chapter (3) with EM, two additional aspects have to be considered.

The first thing to consider is that the Bayesian model is a dynamic Bayesian network, which is recursively defined over time (see Section 3.3.2). To apply EM to a dynamic Bayesian network, one can "unroll" it by instantiating a time slice for each step of an episode. This results in a Bayesian network where the E-step can be regularly executed. In practice, the inference of the E-step is calculated recursively with smoothing techniques to minimize the memory demand, since episodes can become quite long. We refer to (Doucet et al., 2001; Doucet and Johansen, 2009) for more details on smoothing techniques. In order to learn effectively and get reliable estimates for the model parameters, we make a stationary model assumption, meaning that the models do not change over time. As a consequence, the model instances of all time steps and all episodes share their parameters. We further assume that all traffic participants of the same type, e.g., cars, can be described by the same model, so we don't have to learn an individual model for each traffic participant and can generalize the observed behavior patterns. Sharing model parameters over all time steps, episodes and traffic participants allows pooling the statistics of each E-step and results in more reliable model parameter estimates in the M-step. Figure 4.2 visualizes the concept of parameter sharing for a transition function in a multi-agent Bayesian filter.

The second thing to consider is that there are several different models in the Bayesian model (as described in Section 3.3.3). The parameter vector θ therefore subsumes multiple parameter vectors, one for each model $\theta = (\theta_{m_1}, \theta_{m_2}, \dots, \theta_{m_n})$. Since each model has its own set of parameters and the generalized EM allows partial optimization in the M-step (Neal and Hinton, 1998), it follows that the models can be individually optimized and different learning methods can be combined.

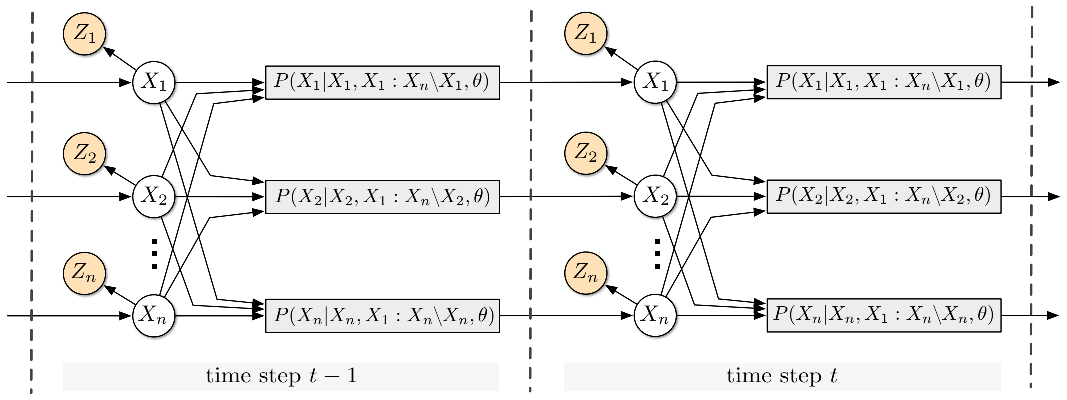

Z_1
Z_2
Z_n
X_1
X_2
X_n
$P(X_1|X_1, X_1 : X_n\backslash X_1, \theta)$
$P(X_2|X_2, X_1 : X_n\backslash X_2, \theta)$
$P(X_n|X_n, X_1 : X_n\backslash X_n, \theta)$
time step $t-1$
time step t

Figure 4.2: Learning a transition model in a multi-agent environment is more effective if all instances of the model share their parameters θ and if the models are time invariant. The transition statistics calculated in the E-step can then be pooled which results in more reliable model parameter estimates.

IS-MCEM model learning procedure The sufficient statistics that are needed for the optimization in the M-step are calculated in the E-step of the IS-MCEM procedure. The E-step estimates the posterior distribution over all hidden states of the dynamic Bayesian network with importance sampling based on the current estimates of the model parameters. The used importance sampling technique is likelihood weighting (see Section 3.4). The posterior distributions of interest are inferred in form of sets of weighted samples. These samples are used to optimize the models in the M-step of the IS-MCEM as described in the following sections. The M-step is carried out in a supervised manner since the

E-step has produced estimates for all non-observable states. After updating the model estimates, a new EM iteration starts using the improved models. This procedure is repeated until convergence.

4.3.1 Learning the Goal Policy Model

The goal policy describes the conditional probability that a traffic participant is heading for a specific goal region as his next intermediate goal (Section 3.3.3). The part of the goal policy model (3.9) that has free parameters and is therefore target of the learning process is σ_+^G. It defines the conditional distribution over the next goals for the cases that the traffic participant has reached his current goal or that he deviated from the intended course making his current goal no longer reachable. At this point, we reduce σ_+^G to

$$\sigma_+^G(\pi^G|x, c^L, \pi^{G-}) = \sigma_+^G(\pi^G|c^L)\,, \tag{4.16}$$

meaning that the next goal only depends on the current lane of the traffic participant and not on the other traffic participants. This is a mild assumption since the strategical decisions mainly depend on the final goal of a traffic participant. This assumption simplifies the model and improves the speed of convergence in the learning process.

We define σ_+^G to take the form of a sparse Gibbs distribution

$$\sigma_+^G(\pi^G = g|c^L = l) = \begin{cases} \exp(\theta_{g,l}) / \left(\sum_{g_i \in \mathscr{G}_l} \exp(\theta_{g_i,l})\right) & \text{if } \mathit{reachable}(g,l) \\ 0 & \text{otherwise} \end{cases} \tag{4.17}$$

with $\mathscr{G}_l = \left\{g | g \in \mathscr{G} \wedge \mathit{reachable}(g,l)\right\}$ being the set of goals that are directly reachable from lane l. The parameters of the model are $\theta_G = \left(\theta_{l_1}, \ldots, \theta_{l_{|\mathscr{L}|}}\right)^\mathsf{T}$ with $\theta_{l_k} \in \mathbb{R}^{|\mathscr{G}_{l_k}|}$. Larger parameter values result in higher probabilities for goals to be predicted.

The parameters are optimized with gradient-based learning (see Appendix A.4) in the M-step. Therefore, we derive the partial derivatives of σ_+^G

with respect to the parameters in order to optimize the expected complete data log likelihood (see (4.15))

$$\frac{\partial \sigma_{+}^{G}(g_t|l_t)}{\partial \theta_{l,g}} = \begin{cases} \dfrac{\exp(\theta_{g,l})\sum_{g_i \in \mathcal{G}_l \setminus g}\exp(\theta_{g_i,l})}{(\sum_{g_i \in \mathcal{G}_l}\exp(\theta_{g_i,l}))^2} & \text{if } g_t \in \mathcal{G}_{l_t} \wedge g_t = g \wedge l_t = l \\ \dfrac{\exp(\theta_{g_t,l} + \theta_{g,l})}{(\sum_{g_i \in \mathcal{G}_l}\exp(\theta_{g_i,l}))^2} & \text{if } g_t \in \mathcal{G}_{l_t} \wedge g_t \neq g \wedge l_t = l \\ 0 & \text{otherwise} \end{cases} . \tag{4.18}$$

The parameters are initialized with some constant, which results in a uniform distribution over all reachable goal regions and expresses an non-informative prior. In the course of learning, the transition probabilities are adapted to meet the observed goal transition frequencies.

4.3.2 Learning the Route Policy Model

The route policy model (3.3.3) describes the distribution over the possible routes a driver can take to reach a given goal based on the current situation. To learn a route model that is able to generalize to different route networks, it is not enough to just count the frequencies at which individual plans have been executed by traffic participants. What needs to be learned is the relationship between the properties of the available routes, the current situation and the preferences of drivers. To enable the learning of this relationship the route policy model is defined based on two reward functions that measure the utility of different aspects of the routes for an average driver.

Recapitulating the definition of the route policy model, the probabilities of the routes are calculated via a Gibbs distribution (3.17) from the expected utility measure $q_s(\pi^R, s^R)$ (3.15). The expected utility measure depends on a reward function that assesses the constituents of the routes. Learning the route policy model thus concentrates on learning the reward function.

We apply gradient-based learning to obtain the maximum likelihood estimates of the route policy model parameters, which consist of the parameters of the reward function θ_R.

Deriving the route policy model (3.12) with respect to the parameters yields

$$\frac{\partial p(\pi^R|s^R,t^R)}{\partial \theta_R} = \begin{cases} \frac{\partial \sigma_+^R(\pi^R|s^R)}{\partial \theta_R} & \text{if } t^R = \mathit{true} \\ 0 & \text{otherwise} \end{cases} . \tag{4.19}$$

The partial derivatives of σ_+^R with respect to the parameters for the case $t^R = \mathit{true}$ result in

$$\frac{\partial \sigma_+^R(\pi^R|s^R)}{\partial \theta_R} = \begin{cases} \frac{\partial}{\partial \theta_R}\left(\frac{\exp(q(\pi^R,s^R))}{\sum_{\pi_j^R \in \mathscr{R}_{\pi^G,c^L,\pi^{R-}}} \exp(q(\pi_j^R,s^R))}\right) & \text{if } \pi^R \in \mathscr{R}_{\pi^G,c^L,\pi^{R-}} \\ & \quad \wedge c^L \in \mathscr{L}_{\pi^{R-}} \\ \frac{\partial}{\partial \theta_R}\left(\frac{\exp(q(\pi^R,s^R))}{\sum_{\pi_j^R \in \mathscr{R}_{\pi^G,c^L}} \exp(q(\pi_j^R,s^R))}\right) & \text{if } \pi^R \in \mathscr{R}_{\pi^G,c^L} \\ & \quad \wedge c^L \notin \mathscr{L}_{\pi^{R-}} \\ 0 & \text{otherwise} \end{cases} . \tag{4.20}$$

The partial derivatives for the first two cases of 4.20 only differ in the set of routes over which to normalize. We can therefore derive the partial derivatives of the expression together using a general set of routes $\mathscr{R}_g$, which results in

$$\frac{\partial}{\partial \theta_R}\left(\frac{\exp(q(\pi^R,s^R))}{\sum_{\pi_j^R \in \mathscr{R}_g} \exp(q(\pi_j^R,s^R))}\right) = \frac{\exp(q(\pi^R,s^R))}{\left(\sum_{\pi_j^R \in \mathscr{R}_g} \exp(q(\pi_j^R,s^R))\right)^2} \left[\frac{\partial q(\pi^R,s^R)}{\partial \theta_R} \sum_{\pi_j^R \in \mathscr{R}_g \setminus \pi^R} \exp(q(\pi_j^R,s^R)) - \sum_{\pi_j^R \in \mathscr{R}_g \setminus \pi^R} \frac{\partial q(\pi_j^R,s^R)}{\partial \theta_R} \exp(q(\pi_j^R,s^R))\right] \tag{4.21}$$

with the partial derivatives of the expected utility being

$$\frac{\partial q_s(\pi^R, s^R)}{\partial \theta_s} = \sum_{l=1}^{|\pi^R|-1} \frac{\partial r}{\partial \theta_s} \left(seg(\pi^R, l), seg(\pi^R, l+1), s^R \right) . \tag{4.22}$$

As a parametric model for the reward functions, we use artificial neural networks (ANNs) (Rojas, 1996). ANNs can be trained with gradient-based methods (LeCun et al., 1998; Martens, 2010) for supervised learning tasks. In the case of the reward function, the ANN cannot be trained directly, since the true reward values are unknown. However, it can be indirectly trained as part of the route policy model. For optimizing the route policy model with gradient-based methods the partial derivatives of the reward function (4.22) with respect to the network parameters are needed. The partial derivatives of interest can be derived by recursively applying the chain rule and efficiently calculated using back-propagation. For details regarding ANN functions and their partial derivatives, we refer to Appendix A.5.

With the parametric model and its partial derivatives defined, the route policy model can be optimized with gradient-based methods. Since the input of the model comprises features of the route and the situation and not only a route index, general relationships for route selection preferences of drivers can be learned and applied to new network configurations and situations. This way, it can be learned for example that drivers in most situations prefer routes with no lane changes or overtaking maneuvers.

4.3.3 Learning the Action Policy Model

The action policy model resembles the decision making of a traffic participant on the level of continuous controls, i.e., in terms of acceleration and turning. It is the key model and also the most complex one of the regarded models. The action policy model ties together all the information required to formulate the relationship between a situation and the distribution over actions that a traffic participant can conduct. Based on the state of all other traffic participants, the individual perspective of a traffic participant on the situation and his goal and plan, the model defines how probable it is that this driver conducts a specific action. We argued in Section 4.1 that it is difficult to model this relationship manually and that it is beneficial for several reasons to use a data-driven approach to obtain the model. In this section, it is shown how to learn the action policy model from traffic observations.

In the presented Bayesian model, the transition model is divided into several sub models, mainly the action policy model and the motion model. While it also would have been possible to learn a transition model directly, this modeling choice was made to promote learning through the following aspects: The use of existing domain knowledge in form of motion models simplifies the learning task. Not only has this relationship not to be learned but more importantly, it makes the exploitation of invariance directly accessible to learning, e.g., like rotational and translational invariance. Since the behavior of a traffic participant is the same under affine transformations of the global world frame, the dependencies learned in one situation can be generalized more easily to similar situations. All these aspects can be seen as an improvement of the inductive bias of the learner and therefore help to enhance generalization. As a result, less samples are needed to reach the same accuracy compared to a transition model learned directly.

We model the action policy model through a conditional multivariate normal density

$$p(\pi^A | x, x_{1:n\backslash i}, c, \pi^R, \pi^{A-}) = p(\pi^A | s^A) \quad \sim \quad \mathcal{N}(\mu_s, \Sigma_s) \tag{4.23}$$

with s^A summarizing the situational dependencies $s^A = (x, x_{1:n\backslash i}, c, \pi^R, \pi^{A-})$. Even though the density itself is unimodal, the joint distribution together with the plans and goals is multimodal and covers the different possible ways in which a traffic a participant can behave.

Note that not only the mean μ_s of $p(\pi^A | s^A)$ depends on the situation but also the covariance Σ_s of the normal density, which represents the uncertainty in choice of action in a given situation. It is crucial for this learning task to consider this input-dependent noise, since the uncertainty can vary heavily depending on the situation. In cases where a lot of variability in the possible set of rational actions exists, the uncertainty mass spreads over a larger area than in situations where traffic participants behave uniformly. Modeling this heteroscedasticity of the data is even more important in time series prediction tasks since inaccuracies multiply over time. Figure 4.3 shows a comparison between the predictions of a homoscedastic model and a heteroscedastic model learned from data of an intersection scenario. In this example the yaw rate ω is predicted depending on the current velocity v. To emphasize the point, it can be seen that the homoscedastic model largely overestimates the variance of the yaw rate for velocities $v < 4.5\,\frac{\mathrm{m}}{\mathrm{s}}$ and underestimates it elsewhere. The heteroscedastic model on the other hand provides significantly more accurate estimates of the input-dependent variance.

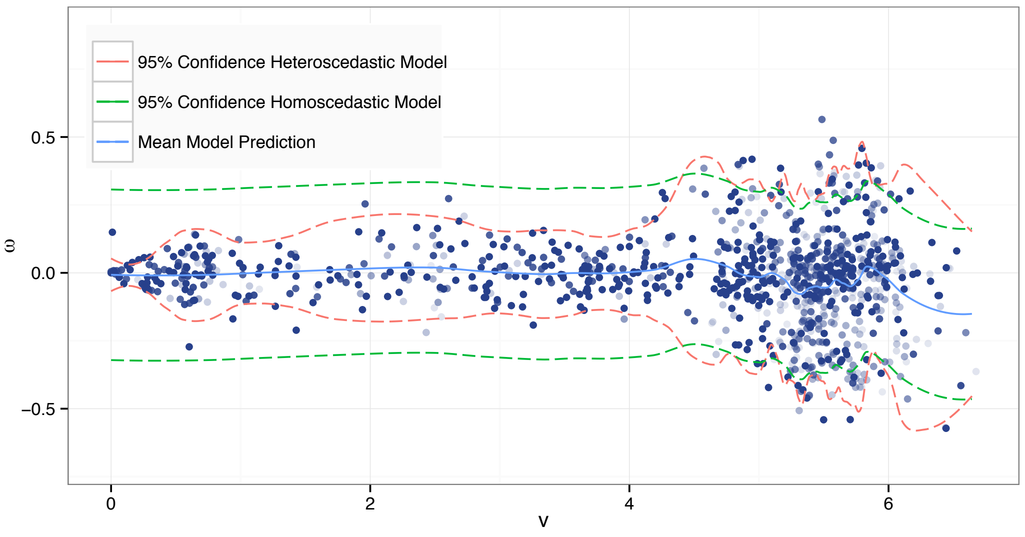


Figure 4.3: Example of input-dependent noise present in the data of an intersection scenario. The homoscedastic model largely overestimates the variance of the yaw rate for velocities $v < 4.5\,\frac{\mathrm{m}}{\mathrm{s}}$ and underestimates it elsewhere. The heteroscedastic model on the other side is able to provide significantly more accurate estimates of the input-dependent variance.

Taking the input-dependency into account, the conditional multivariate normal density representing the action policy model has the following form

$$p(\pi^A|s^A) = (2\pi)^{-\frac{d}{2}} \det(\Sigma_s)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\pi^A - \mu_s)^\mathsf{T} \Sigma_s^{-1} (\pi^A - \mu_s)\right) \tag{4.24}$$

with $d = 2$ being the dimensionality of the action policy space. The input-dependent mean μ_s and covariance Σ_s are represented by the two non-linear functions:

$$\mu_s = m(s^A) : \mathbb{R}^{|s^A|} \mapsto \mathbb{R}^{|\pi^A|} \qquad \text{(mean function)} \qquad (4.25)$$

$$\Sigma_s = k(s^A) : \mathbb{R}^{|s^A|} \mapsto \mathbb{R}^{|\pi^A| \times |\pi^A|} \qquad \text{(covariance function)} \qquad (4.26)$$

The parameters of the action policy model θ_A therefore consist of the parameters of the mean and the covariance function: $\theta_A = (\theta_\mu, \theta_\Sigma)$. In the next section, it is described how these two non-linear functions and, with it, the action policy model can be learned with non-parametric learning techniques.

Learning The data basis for learning the action policy model are the samples retrieved in the E-step of the EM procedure (4.2.2). The samples have the form $\mathcal{D}_A = \{\langle \pi_i^A, s_i^A, w_i \rangle\}_{i=1}^N$ and relate situations s_i^A withs actions π_i^A. A vector s^A mainly consists of the context relations and forms the input features for the learning procedure. The objective of the M-step is the optimization of the expected complete data log likelihood (4.8), which for the action policy model reduces to the optimization of the log likelihood of the data $\mathcal{D}_A$ under the action policy model (4.24)

$$\begin{aligned} l_A &= \sum_{i=1}^{|\mathcal{D}_A|} \log p(\pi_i^A | s_i^A) \\ &= -\frac{|\mathcal{D}_A| d}{2} \log(2\pi) - \frac{1}{2} \sum_{i=1}^{|\mathcal{D}_A|} \log(\det(\Sigma_{s_i})) - \frac{1}{2} \sum_{i=1}^{|\mathcal{D}_A|} (\pi_i^A - \mu_{s_i})^\top \Sigma_{s_i}^{-1} (\pi_i^A - \mu_{s_i}) \,. \end{aligned} \qquad (4.27)$$

The parameters $\hat{\theta}_A = \text{argmax}_{\theta_A} (l_A)$ that maximize l_A cannot be calculated analytically. We propose finding $\hat{\theta}_A$ with an iterative procedure that estimates the parameters of the mean and covariance function in an alternating fashion. It follows the intuition that if we knew the parameters of the covariance function, it would be easier to optimize the mean function and vice versa. And in fact, as shown in the following section, if the covariances are known, the optimal parameters of the mean can be analytically estimated under some conditions.

The algorithm carries out the following steps:

1. Estimate the initial parameters $\hat{\theta}_\mu = \underset{\theta_\mu}{\operatorname{argmax}}\,(l_A)$ with all covariances $\Sigma_{s_i} = I$ set to the identity matrix.
2. Estimate $\hat{\theta}_\Sigma = \underset{\theta_\Sigma}{\operatorname{argmax}}\,(l_A)$ using the previously estimated mean function to calculate the residuals.
3. Estimate $\hat{\theta}_\mu = \underset{\theta_\mu}{\operatorname{argmax}}\,(l_A)$ using the previously estimated covariance function.
4. Iterate from step 2 until convergence.

The algorithm optimizes the model parameters θ_A by partially improving the estimate with respect to its constituents θ_μ and θ_Σ. This is possible due to the fact that the mean and covariance functions share no parameters. Since l_A is increased in each step or at least remains unchanged, it follows that the algorithm eventually converges to a local optimum under the condition that l_A is bounded upwards.

Depending on the type of model chosen for the mean and covariance function, the optimization steps can be further simplified. Analysing the optimization of the mean function yields

$$\underset{\theta_\mu}{\operatorname{argmax}}\, l_A = \underset{\theta_\mu}{\operatorname{argmin}}\,(l_\mu) \quad \text{with} \tag{4.28}$$

$$l_\mu = \sum_{i=1}^{|\mathscr{D}_A|} (\pi_i^A - \mu_{s_i})^\top \Sigma_{s_i}^{-1} (\pi_i^A - \mu_{s_i})\,, \tag{4.29}$$

meaning that the maximum likelihood estimate of the mean function is equivalent to the minimum of the well known sum of squared errors objective. If the mean function $m(x)$ has a form that is linear in the parameters (e.g. polynomials), the minimization problem can be analytically solved. In this case, the mean function can be written as $m(s^A) = \phi(s^A)\theta_\mu$ with $\phi(s^A)$ being some function of s^A. The optimal solution is derived by setting the derivative to zero and solving for the parameters. The derivative is

$$\frac{\partial l_\mu}{\partial \theta_\mu} = -2 \sum_{i=1}^{|\mathscr{D}_A|} (\pi_i^A - m(s_i^A))^\top \Sigma_{s_i}^{-1} \frac{\partial m(s_i^A)}{\partial \theta_\mu} \tag{4.30}$$

with $\frac{\partial m(s_i^A)}{\partial \theta_\mu} = \phi(s^A)$. Setting the expression to zero ultimately yields the optimal mean function parameters

$$\hat{\theta}_\mu = \left(\sum_{i=1}^{|\mathscr{D}_A|} \phi(s_i^A)^\top \Sigma_{s_i}^{-1} \phi(s_i^A) \right)^{-1} \left(\sum_{i=1}^{|\mathscr{D}_A|} \phi(s_i^A)^\top \Sigma_{s_i}^{-1} \pi_i^A \right) . \tag{4.31}$$

This corresponds to the solution of a multivariate weighted least squares problem with partially correlated observations. The weighting in this interpretation is inverse to the covariance of the noise $W_i = \Sigma_{s_i}^{-1}$ for the given input s_i^A, i.e., the greater the noise, the less the influence of the observation on the estimate. The difference to (feasible) generalized least squares is that the variance-covariance is derived from the (non-linear) covariance function $k(x)$ instead of being directly estimated from the residuals (Björck, 1996). Even though the closed form solution only exists in the linear case, we can make use of it for subproblems arising in the chosen learning procedure described later in this section, where locally linear models are estimated.

For the optimization of the covariance function, the objective (4.27) can be transformed to

$$\underset{\theta_\Sigma}{\operatorname{argmax}}\, l_A = \underset{\theta_\Sigma}{\operatorname{argmin}} \left[\sum_{i=1}^{|\mathscr{D}_A|} \Big(\log(\det(\Sigma_{s_i})) + (\pi_i^A - \mu_{s_i})^\top \Sigma_{s_i}^{-1} (\pi_i^A - \mu_{s_i}) \Big) \right] . \tag{4.32}$$

In case that Σ_{s_i} is constant for all s^A, the maximum likelihood estimate can be solved in closed form and is just the covariance of the residuals

$$\hat{\theta}_{\Sigma_{s_i}} = \frac{1}{|\mathscr{D}_A|} \sum_{i=1}^{|\mathscr{D}_A|} (\pi_i^A - \mu_{s_i})(\pi_i^A - \mu_{s_i})^\top . \tag{4.33}$$

Choosing a constant covariance function allows only homoscedastic models and is therefore not able to model input-dependent noise. However, in cases where the covariance function is piecewise constant, the closed form estimator can be applied to obtain maximum likelihood estimates for the partitions.

We use a generalized form of random forests (Breiman, 2001) for learning both the mean and the covariance function. Random forest are ensembles of decision trees and constitute a finite mixture function. We chose random forests for the learning task since they unite several beneficial properties. Random forest as well as decision trees can express non-linear functions with arbitrary precision and can handle mixed domains that involve nominal, categorial and continuous variables. The construction process for the individual decision trees is simple and fast, and has only a small number of hyperparameters. In case of ensembles, the learning process can be easily parallelized. What makes them especially well suited for the application in our case is that function evaluation and sampling is very efficient (Breiman, 1984). This aspect is of great importance since the inference calculates Monte-Carlo estimates of the posterior distributions (3.4), and therefore, the computational demand for sampling is the determining factor for the possible length of prediction horizon and estimation accuracy that is reachable under real-time conditions.

Random forests and bagged decision trees show very good generalization properties in supervised learning tasks and belong to the best state-of-the-art general purpose learning algorithms (Caruana and Niculescu-Mizil, 2006), (Caruana et al., 2008). In comparison to a single decision tree, ensembles of trees significantly better generalize as shown by number of empirical studies (Breiman, 2001). Another important property of decision tree induction and, in turn, of learning ensembles of decision trees is that the learning is non-parametric. This means that the complexity of the model is determined by the data and as a consequence the risk of over- and underfitting is reduced. What makes decision trees especially interesting for our application, is that they can be constructed sequentially (Ikonomovska, 2012). This enables the potential optimization of the models online when new data becomes available, and could enable livelong learning for such a prediction system.

An abstract algorithm for recursive decision tree induction in a non-parametric way is the following:

1. Find the best split and basis functions parameters according to the objective function E and the data $\mathscr{D}$: $(\hat{\theta}_s, \hat{\theta}_{b1}, \hat{\theta}_{b2}) = \operatorname{argmin}_{\theta_s, \theta_{b1} \theta_{b2}} E(\mathscr{D} | \theta_s, \theta_{b1}, \theta_{b2})$
2. If the objective value of the split is not better than the objective value of the current nodes' basis function without the split ($E(\mathscr{D} | \hat{\theta}_s, \hat{\theta}_{b1}, \hat{\theta}_{b2}) \geq E(\mathscr{D} | \theta_{b0})$) then terminate the current recursion
3. Else branch the node and set the split function to $\hat{\theta}_s$ and the two new basis functions to $\hat{\theta}_{b1}$ and $\hat{\theta}_{b2}$. Partition the data accordingly and recurse with step 1 for both branches if no stopping condition is met.

Most greedy decision tree learning algorithms in the literature fall under the described algorithm. What differentiates them from each other is the choice of objective function, the search strategy to obtain the best split in step 1, the choice of basis and split functions and the stopping criteria.

We use this algorithm for learning both the mean and the covariance function but with different objective functions and different basis functions. For the mean function we use linear basis functions together with the previously described objective L_{θ_μ} (4.29). This allows estimating the optimal basis functions analytically for any given split via the derived estimator $\hat{\theta}_\mu$ (4.31). For the covariance function we use constant basis functions and the objective L_{θ_Σ} (4.32). With this choice of basis functions, the optimal parameters can also be estimated analytically with $\hat{\theta}_\Sigma$ (4.33). In both cases, we use oblique split functions which partition the space according to hyperplanes represented by a linear function (Murthy et al., 1994). The search strategy is to generate a set of candidate splits derived from the data, partition the data according to the candidate splits, estimate the optimal basis functions for the splits and choose the best split among the candidates according to the objective function. In the next chapter, it is shown how decision trees can be further optimized to reach better generalizations in high dimensional learning problems.

Feedback loop[1] To learn a meaningful action policy model, another aspect needs to be considered. While the route policies of traffic participants have the semantics that the driver chooses actions that follow a route, the action policy model as presented so far is not constrained to comply with these semantics. Even though the planned route is provided as input to the action policy model, it is up to model to ensure that actions are derived that respect the semantics of the route policy model, i.e. those actions are most likely that lead to trajectories that follow the route. In order to supply this information to the learning algorithm and thereby enabling it to learn the relationship between routes and actions, we introduce an additional feedback loop into the Bayesian model.

The feedback loop is realized through an additional binary random variable C^R that indicates if an action at time step t yields a traffic participant state at $t+1$ that follows the route planned at t. Together with a virtual evidence of *true* that is set on the node during the learning phase, it is guaranteed that only state sequences have high probabilities where the actions are realizations of the planned routes. The constraint model looks as follows

$$P(c^R|c^L,\pi^{R-}) = \begin{cases} \textit{true}\text{:} & 1-\epsilon \quad \text{if } c^L \in \mathscr{L}_{\pi^{R-}} \\ \textit{false}\text{:} & \epsilon \quad \text{else} \end{cases}, \tag{4.34}$$

where the parameter $0 \leq \epsilon \ll 1$ denotes the probability for the exception that the road user does not follow his planned route. As a consequence, if actions and planned routes are inconsistent, the corresponding particles are down-weighted in the E-step. Since consistent state sequences become more likely, the optimization in the M-step makes this constraint implicit and results in an action policy model that will produce actions which are increasingly consistent with each EM iteration. Note that the feedback loop is only needed to guide the learning process. Once learned, the constraint becomes part of the action policy model.

Physical constraints Another source of prior knowledge that can be incorporated to leverage learning and ascertain the semantics of the actions are physical constraints. The motion model described in Section 3.3.3 already consid-

[1]This paragraph has also been published with similar content in (Gindele et al., 2015).

ers some of the non-holonomic constraints induced by the kinematics of a car. What is not expressed by the differential equation 3.19 are the limitations of steering and acceleration. We refer to Appendix A.3 for details.

4.4 Summary and Conclusion

In this chapter, we showed how the policy models that resemble the decision process of traffic participants can be learned from data. We presented a learning approach that finds patterns in the behavior of traffic participants to derive the dependencies between situations and the decisions of traffic participants.

The presented method is able to learn these dependencies just by observing the behavior of participants acting in traffic, even though the true states of interest are never directly observable and the available measurements are noisy. To leverage generalization, we showed how prior knowledge such as expressive context relations, traffic regulations, motion models, map data and physical constraints can be incorporated in the learning process.

One exceptional aspect of the presented learning approach is that it does not require manually labeled data. This makes it cost-effective to use large amounts of data for learning. Combined with non-parametric learning techniques, it enables the learning process to scale up and learn even subtle relationships.

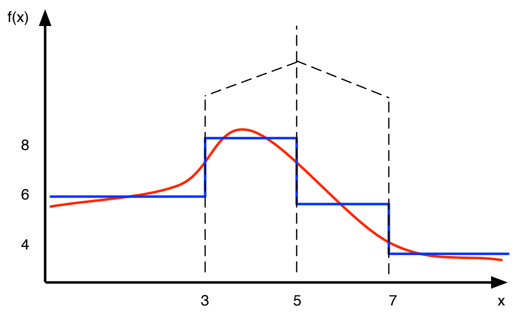
5 Machine Learning Methods

We present a generalization of decision tree models together with novel machine learning algorithms for decision tree induction with optimized decision functions.

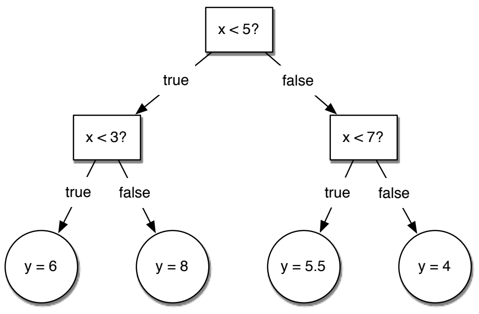
Decision trees are a popular tool for universal function approximation. They represent piecewise defined functions and use a tree structure to define a hierarchical partitioning of the domain of interest. Each partition corresponds to a leaf node of the tree and is associated with a target value. To evaluate the tree function, one walks down the tree following the branch corresponding to the partition the input vector x falls into and returns the value of the leaf node.

Decision trees have a lot of beneficial properties which make them appealing for researches and practitioners. They can be used for classification as well as regression tasks. Their simple concept and versatility gave rise to a lot of variants of the original formulation (Murthy, 1998; Safavian and Landgrebe, 1991). One of the main advantages of decision trees is their computational efficiency due to the recursive partitioning of the input space. If applied in the context of Bayesian networks to represent conditional density functions, they allow to obtain Monte Carlo estimates of posterior distributions at low computational costs. This is one of the reasons why we use them in form of ensembles to represent and learn the action policy model (see Section 4.3.3).

There are three aspects necessary to fully describe a decision tree: The tree structure, the partitioning defined through split functions at each internal node, and the values associated with each leaf. Figure 5.1 shows an example of a decision tree function and its corresponding graph representation. Decision trees are often learned in a non-parametric way, meaning that the complexity of the model depends on the data. Since finding an optimal decision tree for a given data set is NP-complete under several aspects of optimality (Hyafil and Rivest,



(a) Decision tree function.



(b) Decision tree graph.

Figure 5.1: Example of a decision tree (a) together with its graph representation (b). In the left Figure (a), the function depicted by a red line is approximated by a piece-wise constant decision tree function depicted in blue. The dashed lines illustrate the partitioning of input domain. The corresponding graph representation of the decision tree is shown in the right figure (b). The rectangular nodes represent the decision nodes and define the recursive partitioning. The round leaf nodes represent the function values for the associated partitions.

1976), a lot of effort has been put into designing suitable heuristics for the induction of trees. The most common approach is a recursive construction pattern, where the data is partitioned at each node according to a local optimality criterion (see 4.3.3 for algorithm details). Fully grown trees obtained by such greedy procedures tend to overfit the data. This effect can be mitigated by pruning the tree after the construction with the help of a validation data set (Esposito et al., 1997; Mingers, 1989).

In this chapter, we derive new learning algorithms for decision trees that allow incremental learning by stochastic gradient descent and show improved generalization performance in high-dimensional learning problems.

We start out by describing a framework for decision trees with arbitrary basis functions and arbitrary hard split functions for classification and regression tasks. It is shown that classical tree functions like oblique trees (Murthy et al., 1994) and model trees (Quinlan, 1992) can be interpreted as specializations of this general class of tree functions. It is shown that the direct application of gradient-based learning is not able to optimize the splits for a broad class of loss functions, which originates from the piecewise constant nature of the split functions. We approach this problem by relaxing the constraints of the split functions and obtain a new class of generalized decision trees, which can represent smooth functions. For this class, we derive a gradient descent algorithm, which can learn the parameters of both the basis and the split functions simultaneously and in an incremental manner.

Generalized decision trees tend to be smaller in size and enable gradient-based learning but come at the price of increased computational complexity. Since this is not always preferable, we show how generalized regression trees can be used to learn decision trees with hard splits by means of a continuation method (Allgower and Georg, 2003). An evaluation on several data sets shows that the presented algorithms yield trees that generalize better and are smaller in size compared to decision trees that are learned with state-of-the-art heuristic methods.

5.1 Parametric Learning of Generalized Decision Trees

5.1.1 Decision Trees and Induction

The class of decision trees with hard splits describes a function $T(x): \mathbb{X} \to \mathbb{Y}$. It can be formalized in the following way. Let k be an internal node of a tree and $k_1, \dots, k_n$ its n child nodes. The associated n–ary split function

$$s_k(x): \mathbb{X} \to \{k_1, \dots, k_n\}$$

divides the input space $\mathbb{X}$ into n disjoint regions. The final partitioning of the input space into regions R_l encoded by the tree is generated by the conjunction of splits along the pathways from the root to the leaf nodes. Since there exists only one path from each leaf node $l \in \mathit{leafs}$ to the root node, each path can be described by the set of ancestor nodes $\mathit{anc}(l)$ of l. A child node of an internal node a leading to leaf node l is denoted by $c_{a,l}$. Each leaf node l has a basis function $g_l : \mathbb{X} \to \mathbb{Y}$ associated with it, describing the local tree function in the region R_l. In the simplest case, this is just a constant value. The tree function can be stated as follows

$$\begin{aligned} T(x) &= \sum_{l \in leafs} g_l(x) \mathbb{1}_{x \in R_l} \\ &= \sum_{l \in leafs} g_l(x) \prod_{a \in anc(l)} \mathbb{1}_{c_{a,l} = s_a(x)} \end{aligned} \tag{5.1}$$

with $\mathbb{1}$ being the indicator function, which evaluates to 1 if the associated condition is true and 0 otherwise. Due to the crisp nature of split functions, i.e., they can either be 1 or 0, only one term of the sum can be nonzero. To evaluate the whole sum, only the basis function associated with the region x must be evaluated. This locality feature is one of the main properties that makes tree models so attractive. The complexity to find the region which x falls into is $O(\text{depth}(tree))$. In the optimal case of a balanced tree with N nodes, it reduces to $O(\log(N))$.

Notice that in this formulation each split and basis function is defined on the whole input space $\mathbb{X}$, not only on the subspace resulting from previous splits along the path. This yields exactly the same final partitioning of the input space but is more general since each split function is defined independently from the others. This property will become useful in the next section when the model is generalized by making the splits "soft".

Most of the tree models in the literature can be seen as specializations of (5.1). Classic decision trees and regression trees (Breiman, 1984; Quinlan, 1986) are obtained if "axis-parallel" split functions $s_a(x) = \mathbb{1}_{x_i > \theta}$ are used together with constant basis functions $g_i(x) = v_i$. In this case, the input space is divided only in one dimension at each split. By using linear combinations of the input variables as splitting criteria $s_a(x) = \mathbb{1}_{Ax > \theta}$, the class of oblique trees (Murthy et al., 1994) is obtained. Trees applying non-linear transformations to the inputs to partition the input space ($s_a(x) = \mathbb{1}_{\phi(x) > \theta}$) are called multivariate decision trees (Brodley and Utgoff, 1995). Even multi-resolution hierarchies (Moody, 1989) fit this kind of tree function, since one can interpret the sparse grid decomposition at each level as a multi-categorial split function. Exchanging the constant basis functions with more complex functions yields the class of model trees. Well known instances of model trees are piecewise-linear trees (Quinlan, 1992) with linear basis functions and neural trees (Stromberg et al., 1991) that employ artificial neural networks at each leaf.

5.1.2 Gradient-based Learning of Decision Tree Functions

We approach the problem of finding the (locally) optimal parameters of a decision tree with gradient-based optimization techniques. To apply gradient-based learning to decision trees, the first-order partial derivatives (Jacobian) of the tree function $T(x)$ in (5.1) need to be derived. To do so, we first reduce the problem to a parametric learning problem, meaning that the number of parameters of the tree function does not increase with the number of samples. This implies that the size and structure of the tree stays fixed during the course of learning. The parameter vector θ of the function comprises the parameters of all the split and basis functions

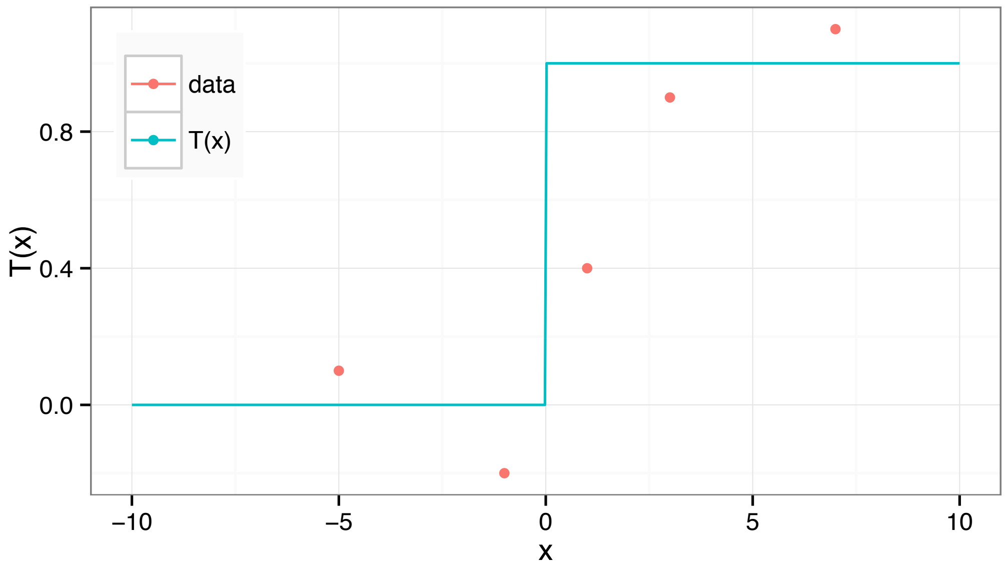
$$\theta = (\theta_s, \theta_g) = (\theta_{s_1}, \ldots, \theta_{s_n}, \theta_{g_1}, \ldots, \theta_{g_m}).$$

To select an initial model (structure and parameters), one can employ for example a standard non-parametric greedy algorithm (Murthy, 1998).

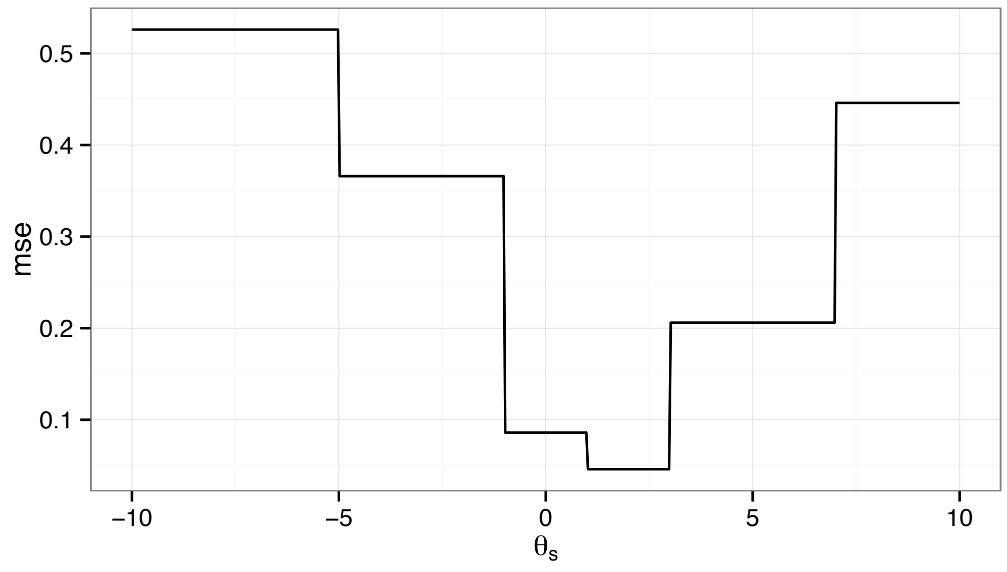
The differentiation of $T(x)$ with respect to the parameters θ yields

$$\frac{\partial T(x)}{\partial \theta} = \sum_{l \in leafs} \mathbb{1}_{x \in R_l} \frac{\partial g_l(x)}{\partial \theta} + g_l(x) \sum_{k \in anc(l)} \frac{\partial \mathbb{1}_{c_{k,l} = s_k(x)}}{\partial \theta} \prod_{a \in anc(l) \setminus \{k\}} \mathbb{1}_{c_{a,l} = s_a(x)} \ . \quad (5.2)$$

Several issues concerning the optimization arise from this formula. The first is the necessary condition that the basis functions $g_i(x)$ need to be differentiable. This is easily resolved by restricting the set of possible basis functions to the class of differentiable functions. All polynomial functions for example belong to this class. The second issue stems from the discontinuities of the indicator functions $\mathbb{1}_{c_{a,l} = s_a(x)}$ associated with the split functions, which render the function non-differentiable. This is only a minor issue because the indicator function belongs to the class of functions which are non-differentiable only on a subset of points with probability zero. It can be shown that the convergence properties are still met, if we set the gradient of the loss function to zero on the non-differentiable subset of points (Bottou, 1998).



(a) Data samples and decision tree function with a split threshold of 0.



(b) MSE loss function with respect to the split function threshold θ_s.

Figure 5.2: MSE loss function for a simple decision tree function with one split and data samples. The loss function is piecewise flat, meaning that the gradient carries no information and as a consequence the split function cannot be directly optimized with gradient-based methods.

Replacing the original loss function with this modified loss function renders it differentiable but discloses the main issue of the current formulation of the tree function: The gradient of the loss function with respect to the parameters of the split functions θ_s are zero for all $x \in \mathbb{X}$. This can easily be seen by taking a look at equation (5.2). The derivatives of piecewise constant indicator functions are zero which renders the right part of the sum zero. The partial derivatives of the left part are also zero because the basis functions do not share any parameters with the split functions. Inserting these partial derivatives of zero into (A.14) results in a loss function gradient of zero for the split function parameters at any point. Since the gradient does not hold any information on how to optimize the split function parameters, they are not adapted during the learning procedure. Hence, applying gradient-based learning to a decision tree function of the form (5.1) will fail to optimize the splits. Figure 5.2 illustrates the problem of a piecewise flat loss function for a simple decision tree with one split. To solve this issue, we generalize the tree function in the next section by relaxing the requirements of the split functions.

5.1.3 Generalized Decision Trees

In order to derive a meaningful gradient, we generalize the tree function $T(x)$ by allowing "soft" split functions. Switching from discontinuous indicator functions to continuous split functions, such as sigmoidal functions, smooths the loss function thereby enabling the optimization of splits. This approach is closely related to the one used to derive a training rule for multi-layer networks (Rumelhart et al., 1986), where similar issues arose before. In the stated case, the discontinuous threshold activation functions of the nodes were replaced by the smooth *logistic* activation functions, which yielded a differentiable multi-layer function optimizable by gradient descent.

The n-ary split function $s_a(x)$ in (5.1) could take on values of its n child nodes. To facilitate soft splits, we change the split functions to $s_a(x) : \mathbb{X} \to [0,1]^n$ with $s_a(c|x)$ denoting the value for child node c. The additional normalization constraint that $\forall x : \sum_{i=1}^{n} s_a(c_i|x) = 1$ enforces a soft partitioning. A tree function with this sort of splits can be stated as

$$\begin{aligned} F(x) &= \sum_{l \in leafs} g_l(x)\, r_l(x) \\ &= \sum_{l \in leafs} g_l(x) \prod_{a \in anc(l)} s_a(c_{a,l}|x)\,. \end{aligned} \tag{5.3}$$

We call this class of tree functions generalized decision tree (GDT). From the convex combination property of the individual split functions follows that the summation over all region values $r_l(x)$ itself yields a convex combination. Hence, the resulting function value $F(x)$ can be seen as an interpolation between the function values of the individual basis functions according to their associated region values. Figure 5.3 shows an example of a GDT that was learned from noisy samples of a sine function.

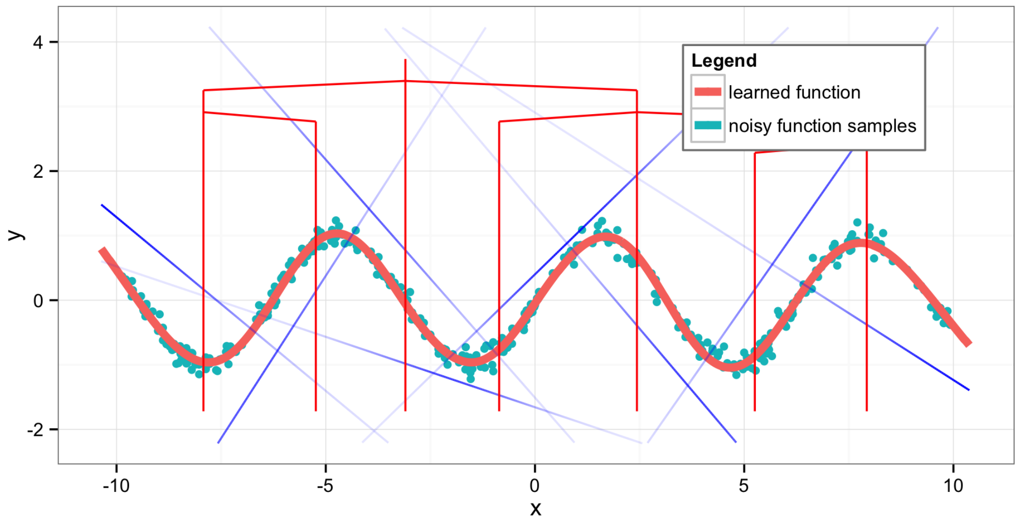
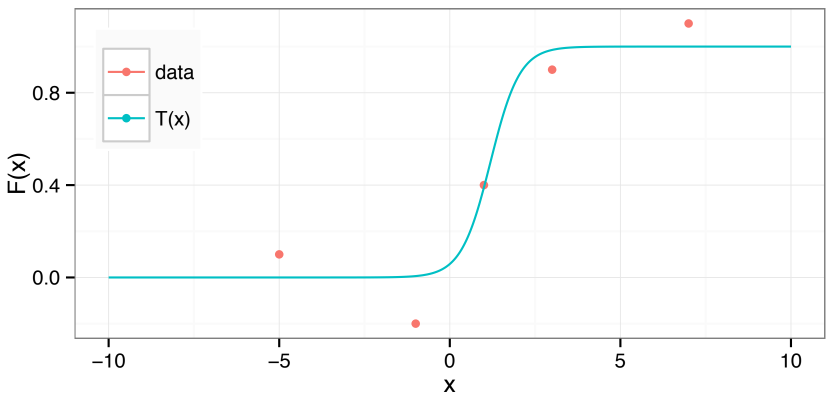
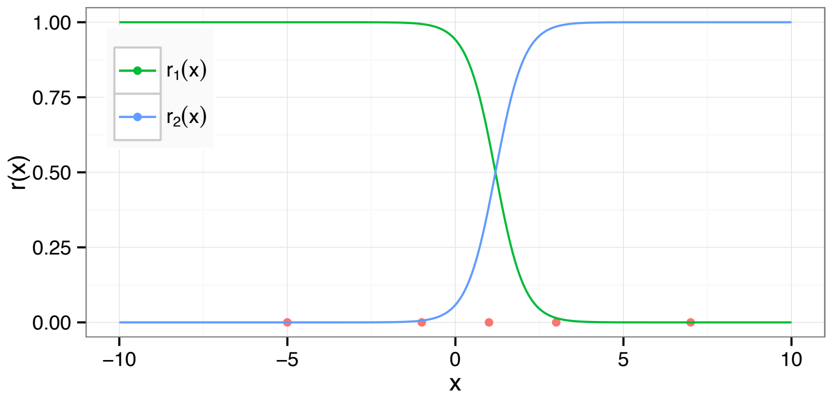


Figure 5.3: Example of a GDT learned with noisy samples from a sine function.

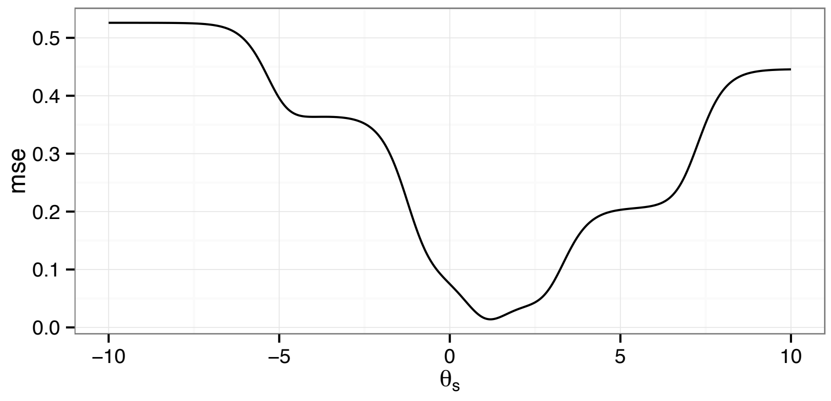
Figure 5.4 shows an example of a generalized decision tree with one split and a logistic split function. The resulting function smoothly interpolates between the constant basis functions according to the region values $r_l(x)$ (5.4b) and fits the data better than the regular decision tree from the previous example (5.2a). In contrast to the regular decision tree, the loss function is smooth with respect



(a) Generalized decision tree function with one soft split.



(b) Region values $r_i(x)$ of the according soft partitioning.



(c) MSE loss function with respect to the split function threshold θ_s.

Figure 5.4: MSE loss function for a simple generalized decision tree function with one split and data samples.

to the split function threshold (5.4) and can be optimized with gradient-based methods.

Since the influence of a region can take on values out of the interval $[0,1]$, regions can overlap. This has a strong implication on the computational complexity of the function evaluation. In the worst case, all basis functions contribute to the function value, i.e. all split and basis functions of the tree have to be evaluated. The complexity of evaluating GDTs is $O(\text{size}(\textit{tree}))$ in contrast to $O(\text{depth}(\textit{tree}))$ for evaluating decision trees with hard splits. However, some functions can be represented more efficiently with GDTs, such as smooth functions, so that the size of a GDT can be smaller than the depth of a DT with equal representational accuracy.

5.1.4 Smooth Split Functions

The requirements of the split functions are the same as for discrete probability distributions. Hence, we can employ suitable distributions as split functions.

A consistent way to derive split functions from general functions $h(x) : \mathbb{X} \to \mathbb{R}^n$ that meet the required criteria, is to use the Gibbs distribution

$$P(Y|X) = \frac{\exp(-\beta E(Y,X))}{\sum_{y\in\mathbb{Y}} \exp(-\beta E(y,X))} \tag{5.4}$$

and to replace the energy function E with $h(x)$. The parameter β is a free scaling parameter. By setting β to -1 the split function becomes

$$s_a(c|x) = \frac{\exp(h_c(x))}{\sum_{i=1}^{n} \exp(h_i(x))}\,, \tag{5.5}$$

which is equivalent to applying the "softmax" function (JS., 1989) to $h(x)$. In the case of a binary tree and $h(x)$ being a linear function $h(x) = Ax + b$ with $h_{c_1}(x) = -h_{c_2}(x)$ and $\beta = -\frac{1}{2}$, one obtains the well known linear logistic function widely used in artificial neural networks (Rumelhart et al., 1986). Other examples of smooth split functions that can be employed for binary trees are Gaussian radial basis functions and scaled sine functions (Haykin, 1994).

A soft variant of a binary oblique tree with linear threshold split functions can be obtained by employing the same linear function together with a sigmoid function like the logistic function $\phi(x) = (1 - e^{-x})^{-1}$.

5.1.5 Partial Derivatives of Generalized Decision Trees

The partial derivatives of $F(x)$ with respect to the parameters of the basis and split functions are

$$\frac{\partial F(x)}{\partial \theta} = \sum_{l \in leafs} r_l(x) \frac{\partial g_l(x)}{\partial \theta} + g_l(x) \sum_{a \in anc(l)} \frac{r_l(x)}{s_a(c_{a,l}|x)} \frac{\partial s_a(c_{a,l}|x)}{\partial \theta}, \tag{5.6}$$

$$\frac{\partial F(x)}{\partial \theta_{g_l}} = r_l(x) \frac{\partial g_l(x)}{\partial \theta_{g_l}}, \tag{5.7}$$

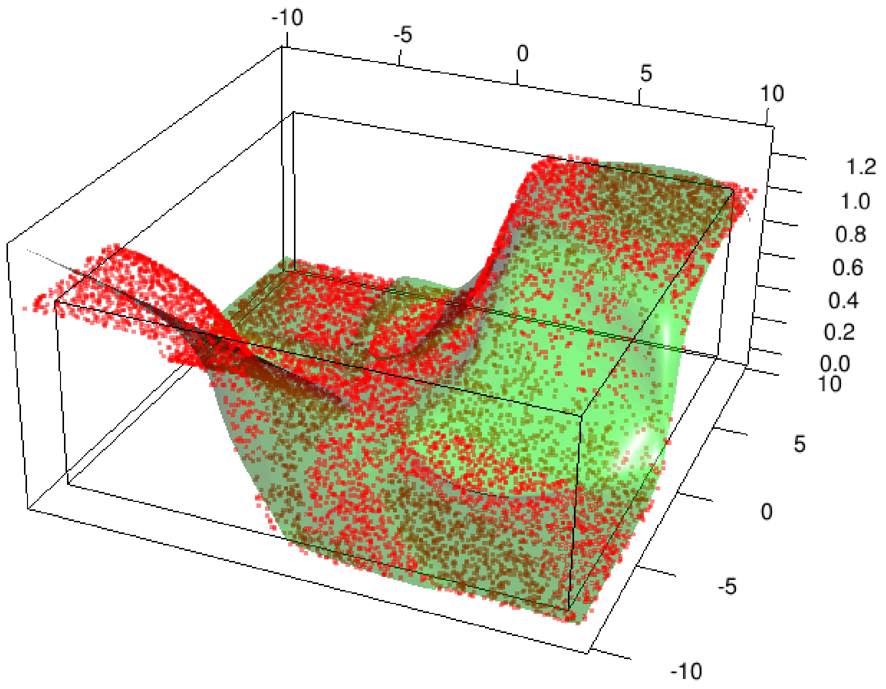
$$\frac{\partial F(x)}{\partial \theta_{s_n}} = \sum_{l \in leafs(n)} g_l(x) \frac{r_l(x)}{s_n(c_{n,l}|x)} \frac{\partial s_n(c_{n,l}|x)}{\partial \theta_{s_n}} \tag{5.8}$$

with *leafs*(n) denoting the set of leaf nodes of the subtree rooted at node n. If using differentiable basis and split functions, the partial derivatives of $F(x)$ are well defined and enable simultaneous learning of both the parameters of the split and basis functions with gradient-based learning.

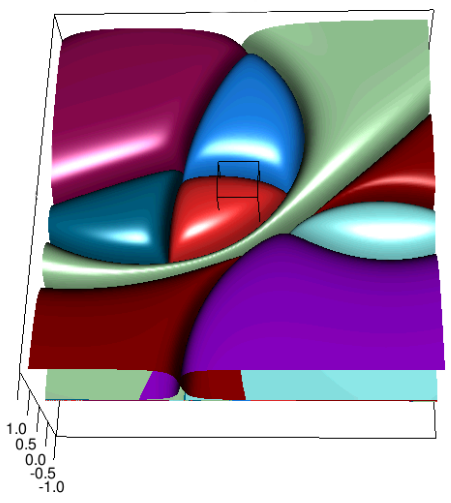
The result of a fitted GDT for a 2d-learning problem can be seen in Figure 5.5. In this example, Gaussian radial basis functions were used as split function, which yield curved regions in the soft partitioning (5.5b).

5.1.6 Relationship of GDTs to other Models

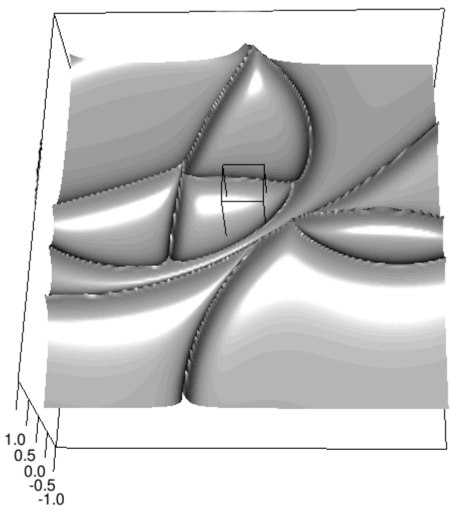
The class of generalized decision trees proposed in this work is closely related to hierarchical mixtures of experts (HMEs) (Jordan, 1994; Jordan and Jacobs, 1994) and fuzzy decision trees (FDTs) (Olaru and Wehenkel, 2003). All three methodologies are tree models and employ split functions that can take on values between zero and one to yield more expressive models. While both HMEs and GDTs use conditional multinomial distributions as split functions, FDTs use membership functions that are in general not normalized. HMEs treat internal decision nodes and leaf nodes as random variables and thereby define a fully probabilistic tree model. The probabilistic nature of HMEs enables the application of likelihood learning methods like expectation maximization. GDTs



(a) Generalized decision tree function (green) fitted to data (red).



(b) Soft partitioning.



(c) Corresponding soft borders.

Figure 5.5: Example of a 2-dimensional function that is fitted with a generalized decision tree. The upper figure (a) shows the training samples and the fitted function. The lower figures show the soft partitioning of the input space (b) and the corresponding soft margins (c).

are more general than HMEs since they impose no restrictions on the choice of basis functions. While HMEs always yield a conditional mixture distribution with one component for each leaf, GDTs can model arbitrary distributions. This can be achieved for example by using the output values of the tree function to parametrize a target distribution, analogously to (Williams, 1996). In the case that GDTs are used with conditional densities as basis functions, GDTs and HMEs become equivalent. As stated before, FDTs use unnormalized membership functions which form a superset of conditional multinomial probability distributions. However, they are restricted to constant basis functions. Since the normalization requirement for split functions of GDTs is only imposed to preserve a soft partitioning and is not required to derive the gradient (5.6), it follows that the learning rules of GDTs also apply to FDTs.

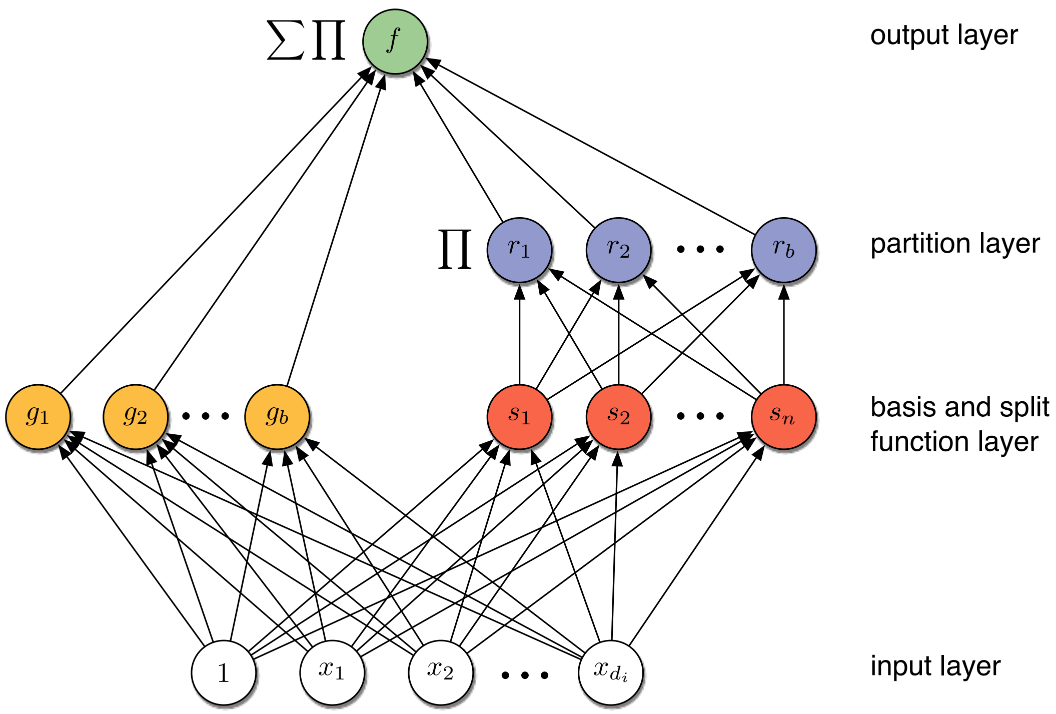


Figure 5.6: A generalized decision tree cast as a feed forward multilayer network. Higher order neurons are necessary to calculate the products appearing in the tree function.

There exists also a connection to artificial neural networks (feed forward multilayer networks). Authors like (Sethi, 1990) and (Shah and Sastry, 1999) showed how a regular decision tree with hard splits and constant basis functions (5.1) can be cast as a three layer feed forward network with perceptron units. The interesting part of this conversion is that the number of layers is constant, i.e. independent of the size of the tree. The size affects only the number of hidden nodes that are needed to encode the tree structure.

Generalized decision trees can also be stated as 3-layer feed forward networks but requires more complex node functions than linear sigmoid functions to encode them efficiently. The structure of a generalized decision tree cast as a feed forward multi-layer network is depicted in Figure 5.6. The first hidden layer calculates the values of all the basis and split functions from the inputs individually. The second layer combines the split values according to the tree structure to evaluate the influence values of the regions. The third and also the output layer combines these values with the basis function values to calculate the weighted sum. Since the combinations in the second and third layer include multiplications of multiple values, higher order neurons are necessary.

5.2 Optimization of Decision Trees with Hard Splits

Using generalized decision trees with smooth split functions enables gradient-based learning but comes at the cost of increased computational complexity. Decision trees with hard splits can be evaluated efficiently due to the non-overlapping regions, but its split function parameters cannot be trained by gradient descent as shown in section 5.1.2. In this section, we derive a method for optimizing non-optimal decision trees with hard splits and thereby retaining the evaluation efficiency. We therefore combine the ideas of generalized decision trees with a continuation method (Allgower and Georg, 2003) and introduce a new specialized family of meta loss functions in order to connect them.

5.2.1 Continuation Approach to Optimizing Flat Objective Functions

In general, continuation methods are used to solve non-linear equations of the form $F(x) = 0$, but can also serve as a strategy to find optima of non-convex cost functions (see for example (Moré and Wu, 1997)). The basic idea is to start with a simplified version of the objective function, e.g., a smoothed version, and then gradually transform it into the objective function of interest. Starting at a minimum θ of the simplified objective, the continuation process keeps θ minimized during the transformation of the objective, guiding it to a dominant minimum of the final objective.

While a continuation method generally serves the purpose of finding better minima for optimization problems by decreasing the risk of getting stuck in local minima early in the optimization process, we utilize it to optimize an objective that is piecewise constant in a subspace of the input variables. Our approach is to start with a soft version of a decision tree and use the gradient of the smoothed objective to optimize the split parameters. We then gradually reduce the softness of the splits, forcing both the basis and split functions to adapt. In the end, the splits become hard and the parameters settle in a local optimum.

In order to apply a continuation method, one has to define a single-parameter family of objective functions C_λ. The scalar parameter $\lambda \in [0,1]$ controls the transition of C_λ from the simplified objective C_0 to the objective of interest C_1

Algorithm 1 Continuation optimization

```
Require: samples $\mathscr{D} = \{z_1, \dots, z_n\}$, parametric cost function $C_\lambda$, starting parameters $\hat{\theta}_0$, steps $m$
1: Init: $\lambda \leftarrow 0,\ \Delta\lambda \leftarrow 1/m,\ i \leftarrow 0$
2: while $\lambda \leq 1$ do
3:     optimize $C_\lambda(\theta_i, \mathscr{D})$, using $\hat{\theta}_i$ as starting value
4:     $\hat{\theta}_{i+1} \leftarrow \theta_i$
5:     $\lambda \leftarrow \lambda + \Delta\lambda$
6:     $i \leftarrow i + 1$
7: end while
8: return $\theta_{i-1}$
```

The simplest version of a continuation algorithm for an optimization task is listed in Algorithm 1. By gradually shifting the objective and using the last found optimum as a starting value for the next iteration, one finally obtains a (local) optimum of the proper objective. Notice that the optimization in line 3 is executed until convergence is reached for the current C_λ. More advanced versions of the predict step (line 4) and the step width adaption (line 5) can be found in (Allgower and Georg, 2003).

5.2.2 Parametric Family of Cost Functions

The first step in deriving the family of cost functions C_λ is the definition of a parametric family of split functions that allows transforming any generalized decision tree with soft splits into one with hard splits. A reasonable way to turn any soft split function $s(c|x)$ into a hard split function $\hat{s}(c|x)$ is to assign to it the child node with the highest influence for a given x: $\hat{s}(c|x) = \mathbb{1}_{c=\text{argmax}_d\, s(d|x)}$. This transformation minimizes the difference between the soft and the hard version of the tree function. To enable a smooth transition between $s(c|x)$ and $\hat{s}(c|x)$ we propose the following parametric form of split functions

$$\tilde{s}(c|x,\lambda) = \frac{s(c|x)^{t(\lambda)}}{\sum_d s(d|x)^{t(\lambda)}} \quad \text{with} \quad t(\lambda) = \frac{1}{1-\lambda} \tag{5.9}$$

and $\lambda \in [0,1]$ and $t(\lambda) : [0,1] \to [1,\infty)$. The parametric split function can shift smoothly between the soft split function $\tilde{s}(c|x,0) = s(c|x)$ and the hard one $\tilde{s}(c|x,1) = \hat{s}(c|x)$. Accordingly, the parametric region function for a leaf node l becomes

$$\tilde{r}_l(x,\lambda) = \prod_{a \in anc(l)} \tilde{s}_a(c_{a,l}|x,\lambda)\,. \tag{5.10}$$

We propose the following parametric loss function L_λ to derive the parametric family of cost functions C_λ from the empirical loss (A.10)

$$L_\lambda(z,\theta) = \sum_{l \in leafs} L_{g_l}(z,\theta)\; \tilde{r}_l(x,\lambda)\,. \tag{5.11}$$

This loss function L_λ weights the loss L_{g_l} of the individual basis functions g_l according to their scaled influence $\tilde{r}_l(x,\lambda)$ given a sample $z=(x,y)$. For $\lambda = 0$, the splits are soft and the loss of the basis functions are weighted proportional to their influence in the overlapping regions. By increasing λ the splits become sharper, assigning more credit to the basis function with the highest influence, while the weighted loss of the other basis functions vanishes. In the end ($\lambda = 1$), only the loss of the basis function with the highest influence contributes to the overall loss L_λ at each point, which equals the loss of the corresponding decision tree function with hard splits.

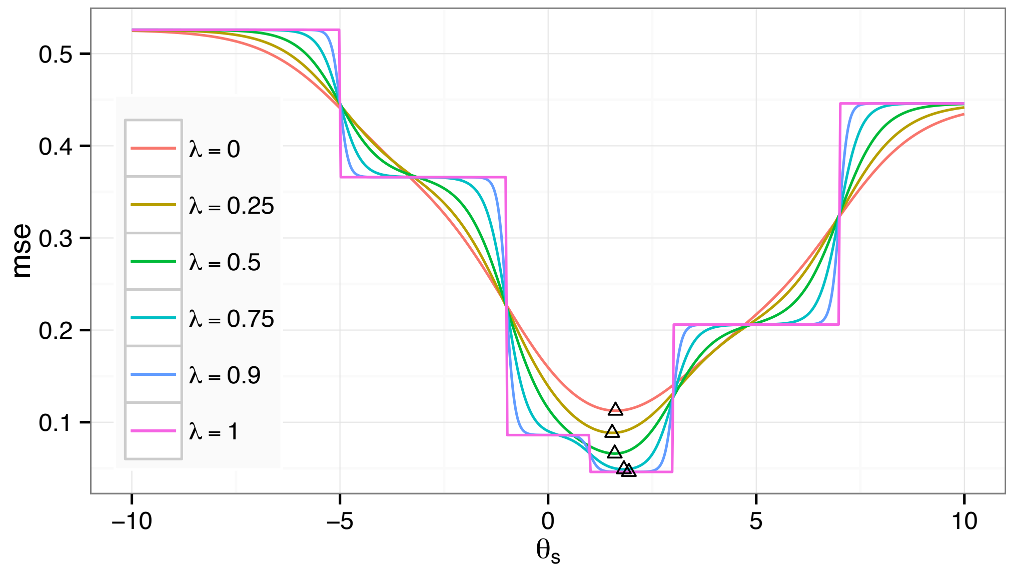


Figure 5.7: Paramteric MSE cost function for increasing values of λ.

Figure 5.7 illustrates the evolution of the parametric cost function for increasing values of λ. The cost function is calculated with respect to the split function threshold and according to the decision tree and data of the previous examples (5.4, 5.2). For $\lambda = 0$ the cost function surface is smooth and has only one optimum that can easily be found with gradient descent. With increasing values of λ, the cost function surface shifts to the actual cost function of interest (compare Figure 5.2b), which is reached for $\lambda = 1$. By tracking the minimum, the optimal split function parameter of the decision tree are found.

5.2.3 Partial Derivatives of the Parametric Loss Function

The gradient of the parametric loss function can be calculated with

$$\frac{\partial L_\lambda(z,\theta)}{\partial \theta} = \sum_{l \in leafs} \left(\frac{\partial L_{g_l}(z,\theta))}{\partial \theta} \tilde{r}_l(x,\lambda) + L_{g_l}(z,\theta)) \frac{\partial \tilde{r}_l(x,\lambda)}{\partial \theta} \right) \tag{5.12}$$

together with the partial derivative of $\tilde{r}_l(x,\lambda)$

$$\frac{\partial \tilde{r}_l(x,\lambda)}{\partial \theta} = \sum_{a \in anc(l)} \frac{\tilde{r}_l(x,\lambda)}{\tilde{s}_a(c_{a,l}|x,\lambda)} \frac{\partial \tilde{s}_a(c_{a,l}|x,\lambda)}{\partial \theta} \tag{5.13}$$

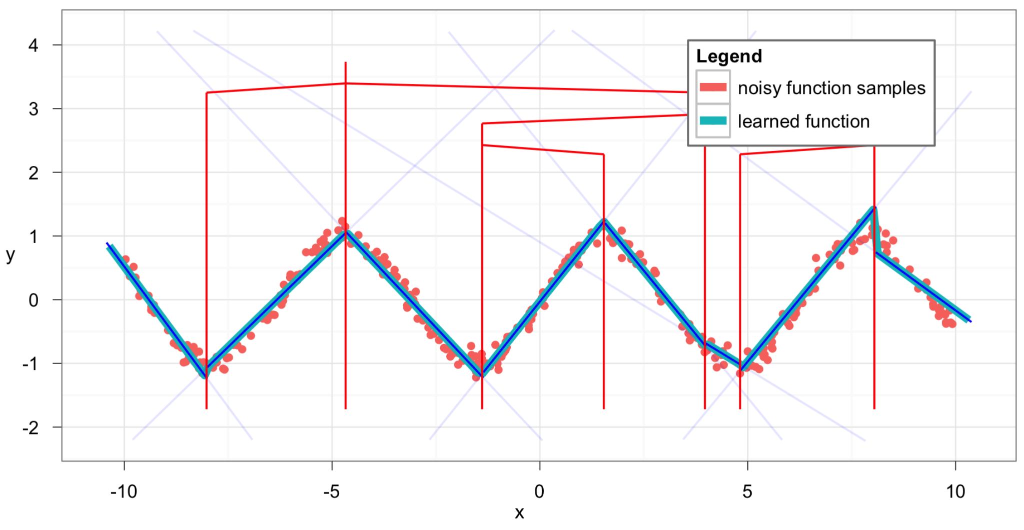


Figure 5.8: Optimized decision tree with hard splits and linear basis functions.

and the partial derivative of $\tilde{s}_a(c_{a,l}|x,\lambda)$

$$\frac{\partial \tilde{s}_a(c_{a,l}|x,\lambda)}{\partial \theta} = \frac{t(\lambda)s_a(c_{a,l}|x)^{t(\lambda)-1}}{\sum_{i \in leafs(a)} s_a(c_{a,i}|x)^{t(\lambda)}}$$

$$\left(\frac{\partial s_a(c_{a,l}|x)}{\partial \theta} - \frac{s_a(c_{a,l}|x)^{t(\lambda)}}{\sum\limits_{i \in leafs(a)} s_a(c_{a,i}|x)^{t(\lambda)}} \sum_{i \in leafs(a)} s_a(c_{a,i}|x)^{t(\lambda)-1} \frac{\partial s_a(c_{a,i}|x)}{\partial \theta} \right) . \quad (5.14)$$

By optimizing C_λ with a gradient descent algorithm (see Appendix A.4) in each step of the continuation algorithm (Alg. 1), we are now able to learn a decision tree with hard splits from any generalized decision tree with differentiable split and basis functions. Figure 5.8 shows an example of an optimized decision tree with hard splits and linear basis functions, which was obtained with the presented continuation method.

5.3 Evaluation of decision tree learning algorithms

The presented learning algorithms are evaluated on three data sets:

Sine (500 samples) A data set consisting of noisy samples drawn from a sine function.

Robot Dynamics (1000 samples) This data set is concerned with the forward dynamics of a robot arm with three joints. The 8-dimensional input consists of the angular positions of the joints, their angular velocities and their torques. The function is highly non-linear and noisy.

Traffic (1000 samples) The data consists of context feature descriptions of traffic situations. The input is 36-dimensional. The prediction variable is the yaw rate of a vehicle acting in the given situations.

Three different decision tree learning methods are compared, the two presented algorithms and a standard DT induction method for comparison.

DT is a decision tree with (hard) linear oblique splits. It serves as a comparison model for assessing the level of improvement gained by the presented algorithms. The used basis functions are constant functions and linear functions respectively. Cost Complexity Pruning is used to prune the tree in order to reduce the risk of overfitting (see (Breiman, 1984) for details).

Opt GDT is a GDT which is derived from *DT*. It uses smooth split functions consisting of compositions of a linear and a sigmoid functions. It is trained with the GDT learning algorithm presented in Section 5.1.

Opt DT is an optimized decision tree with hard splits. It is trained with the continuation based learning algorithm presented in Section 5.2.

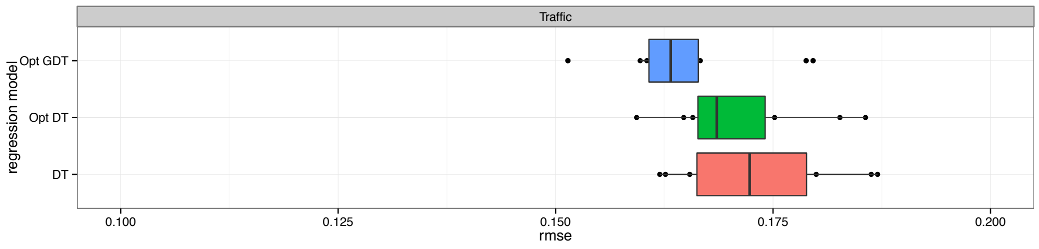
All models minimize the root mean squared error (RMSE) over the data. The performance is measured on a test set consisting of 20% of all samples, the rest is used for training. Each model is trained 10 times, each time randomly drawing a new test set without replacement.

The gradient-based learning of *Opt GDT* and *Opt DT* is realized with stochastic gradient descent and mini batches of 100 samples. To speed-up the convergence, a stochastic diagonal approximation of the Levenberg Marquardt method together with an adaptive learning rate is used (LeCun et al., 1998).

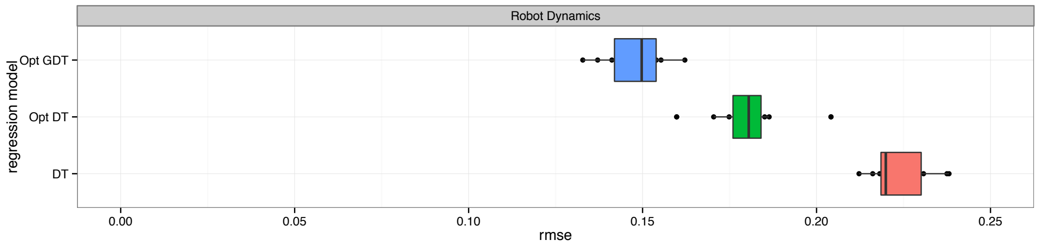
Results and Analysis Table 5.1 reports the results of the compared decision tree learning algorithms. The table states the RMSE achieved by the algorithms on the different data sets together with the standard deviations. On every data set the GDT with soft splits (*Opt GDT*) shows significantly better generalization over the test sets than the decision trees with hard splits. The optimized decision tree with hard splits (*Opt DT*) shows lower errors than the regular decision tree (*DT*). The magnitudes of improvement vary depending on the data set. Figure 5.9 shows the results of the evaluation as box plots.

Table 5.1: RMSE results with standard deviations for different data sets (lower is better)

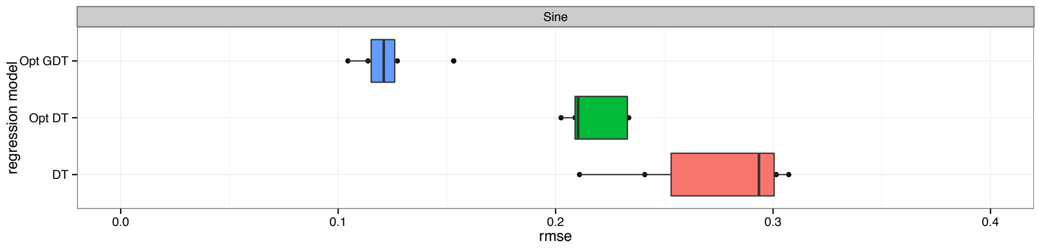
Regression Model	Traffic	Kinematics	Sine
DT	0.173 ± 0.009	0.224 ± 0.009	0.273 ± 0.033
Opt DT	0.171 ± 0.008	0.18 ± 0.011	0.226 ± 0.016
Opt GDT	0.165 ± 0.009	0.148 ± 0.009	0.127 ± 0.014



(a) Evaluation on the *Traffic* data set.



(b) Evaluation on the *Robot Dynamics* data set.



(c) Evaluation on the *Sine* data set.

Figure 5.9: Comparison of decision tree learning methods.

5.4 Summary and Conclusion

In this chapter, we derived novel learning algorithms for decision trees and a newly derived class of generalized decision trees. The developed algorithms show improved generalization properties over existing approaches, which stem from their ability of finding better splits functions through optimization. This becomes increasingly important in high-dimensional spaces where the parameter space of the split functions cannot be searched or sampled exhaustively. Suboptimal split function parameter estimates of greedy induction methods can be corrected with the presented methods by optimizing all parameters of the tree simultaneously. These learning algorithms make the learning of the policy models scalable to complex situations when the dependencies between high-dimensional context descriptions and the resulting decisions have to found.

The learning algorithm for generalized decision trees shows faster convergence than the continuation-based learning algorithm but the gain in learning time comes at the cost of an increased worst-case complexity for function evaluation. Depending on the task requirements each of the presented algorithms can be the right choice.

The incremental nature of the generalized decision tree learning algorithm makes it possible to tackle learning problems with large amounts of data. Although the requirements on the algorithms originated from their application in learning predictive models, the developed algorithms are general machine learning methods for solving classification and regression tasks beyond the traffic domain.

6 Evaluation

In this chapter, the presented approach is evaluated in several different traffic scenarios and several aspects are analyzed. The main interesting aspects are how accurately the Bayesian model can predict the evolution of traffic situations with learned policy models and which context dependencies can be learned. The prediction accuracy is compared to a standard Bayesian filter that does not use any contextual information, but relies solely on the vehicle kinematics and dynamics to predict the future motions of traffic participants. By comparing the two approaches, we investigate the importance of the consideration of interactions between traffic participants, their goals and plans as well as information about the road network. In an additional analysis, the influence of noise in the observations of traffic participants on the learning process accuracy is examined.

It is shown that the presented approach is able to learn the situation-specific behavior of traffic participants. It can provide realistic predictions over time periods of several seconds. The approach is tested in different traffic scenarios without special adaptations which underlines the generality of the approach.

6.1 Evaluation of Prediction Accuracy

6.1.1 Settings

It is proceeded as follows to evaluate how well the presented approach is able to predict future situation developments: Multiple variations of a traffic scenario are recorded in form of traffic episodes. The data set of traffic episodes is split into a training set and a test set. The training set is used for learning the policy models of the presented Bayesian model. The prediction accuracy is estimated on the test set episodes.

Measures The prediction accuracy of the approach is measured in two ways. We calculate the mean log likelihood (MLL) and the root mean squared error (RMSE) of the data.

For an episode of length T with data $\{\bar{y}^{(t)}\}_{t=1}^{T}$ and predicted distributions for the data for each time step $\{p(y^{(t)})\}_{t=1}^{T}$, the MLL measures how good the predictions match the data. The predicted distributions are calculated according to the used model. The MLL is defined as

$$\mathrm{MLL}_y = \frac{1}{T}\sum_{t=1}^{T} \log p(\bar{y}_t)\,. \tag{6.1}$$

Instead of considering the full predicted distributions of the data, the RMSE only compares the expected values of the distributions $\hat{y}^{(t)} = \mathbb{E}\, p(y^{(t)})$ with the data. The RMSE is defined as

$$\mathrm{RMSE}_y = \sqrt{\frac{1}{T}\sum_{t=1}^{T} (\hat{y}^{(t)} - \bar{y}^{(t)})^2}\,. \tag{6.2}$$

While the RMSE is easier to interpret, it has the drawback that it only considers the mean estimation and not the distribution. It does not represent the prediction accuracy well if the predicted distributions have more than one mode. This becomes especially important when measuring the long term prediction accuracy since the prediction then often shows several modes (see for example Figure 6.9). The MLL is a more informative measure since it considers the full distributions of predicted values instead of only the mean.

Data The data for learning and testing is acquired with a simulation framework. Figure 6.1 shows a screenshot of a simulation run. The simulator uses a physics engine to simulate the dynamics of the cars. Cars are driven manually with a steering wheel and pedals as well as by the autonomous driving software that was developed by team *AnnieWAY* for the DARPA Urban Challenge (Gindele et al., 2008; Kammel et al., 2008). For every traffic scenario, multiple episodes are recorded with variations in the driving behavior. For every time step of an episode, the positions, orientations and velocities of all traffic participants are recorded as ground truth data. Before learning, we add

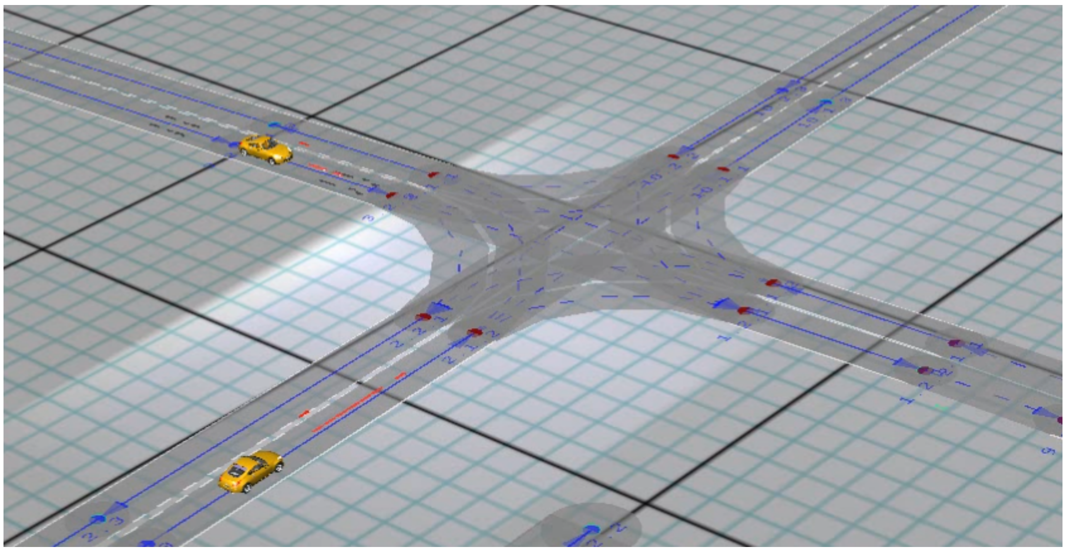


Figure 6.1: Simulation of an intersection scenario with two cars.

white noise to the training data with a standard deviation of $\left(\sigma_{x_1}, \sigma_{x_2}, \sigma_v, \sigma_\psi\right) = \left(0.1\,\text{m},\ 0.1\,\text{m},\ 0.1\,\frac{\text{m}}{\text{s}},\ 0.05\,\text{rad}\right)$ to simulate measurement errors. The episodes are recorded with a frame rate of $3.\overline{3}$ resulting in a time step of $\Delta t = 0.3\,\text{s}$.

Comparison model We compare our approach to a Bayesian filter model which is widely used in tracking applications. It uses a single track motion model (as described in Section 3.3.3) and relies on the vehicle kinematics and dynamics to predict the future motions of traffic participants. The model assumes the current velocity and heading to be constant with some Gaussian noise. The noise parameters are directly estimated from the ground truth control data. The model uses no contextual information. It is therefore well suited to evaluate the impact of mutual influences between road users on the prediction. In the following evaluation, the comparison model is called *single track model* and the presented approach *learned model*.

Model learning The policy models are learned as described in Chapter 4. For learning the action policy model, we use random forests of generalized decision trees with linear oblique splits (Murthy et al., 1994) and linear basis functions (Potts, 2004).

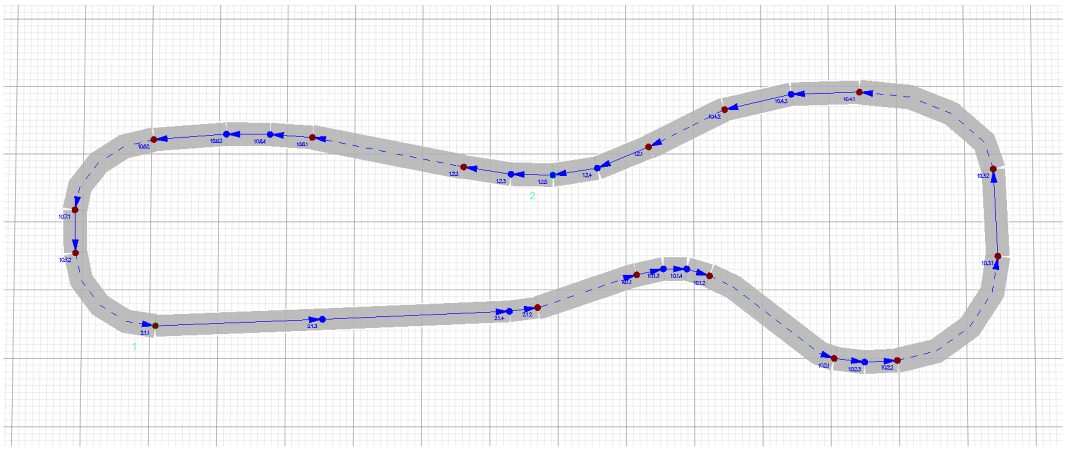


Figure 6.2: Overview of the lane following test track from a bird's-eye perspective.

6.1.2 Lane Following Scenario

The first scenario is a lane following experiment. In this scenario, a single car is observed that drives around on a curvy track with slight variations in velocity. Figure 6.2 shows the map of the track from a bird's-eye perspective. An episode example is depicted in Figure 6.3.

The goal of this scenario is to evaluate how well the approach is able to learn the behavior that regular drivers mainly drive along roads. Since the map data only serves as source of information and not as motion constraint, the dependency between the course of the road and the appropriate control signals in terms of steering and acceleration have to be learned in order to predict the future trajectories correctly.

The models were learned from 50 observed episodes with an average length of 142 time steps ($\sim$43 s). An additional set of 10 episodes was used to evaluate the test set performance.

Results and analysis The tables 6.1 to 6.4 provide the measured prediction accuracy of the learned model and the comparison model. The learned model yields a significantly better accuracy for the state estimation as well for the prediction. The MLL of the single track model prediction with $\Delta t = 6\,\mathrm{s}$ is not defined (NA) since the Monte Carlo estimate yielded a numerical result of 0 due to high uncertainty and the limited amount of samples. For a likelihood of 0 the MLL is not defined due to the logarithm.

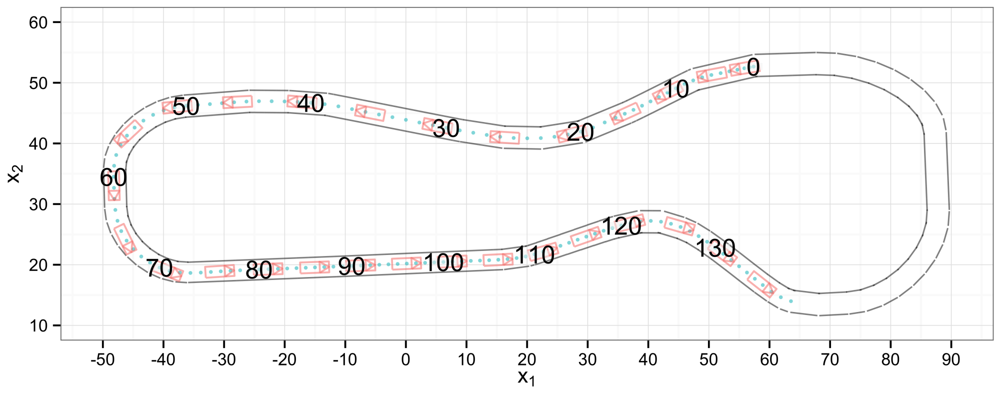


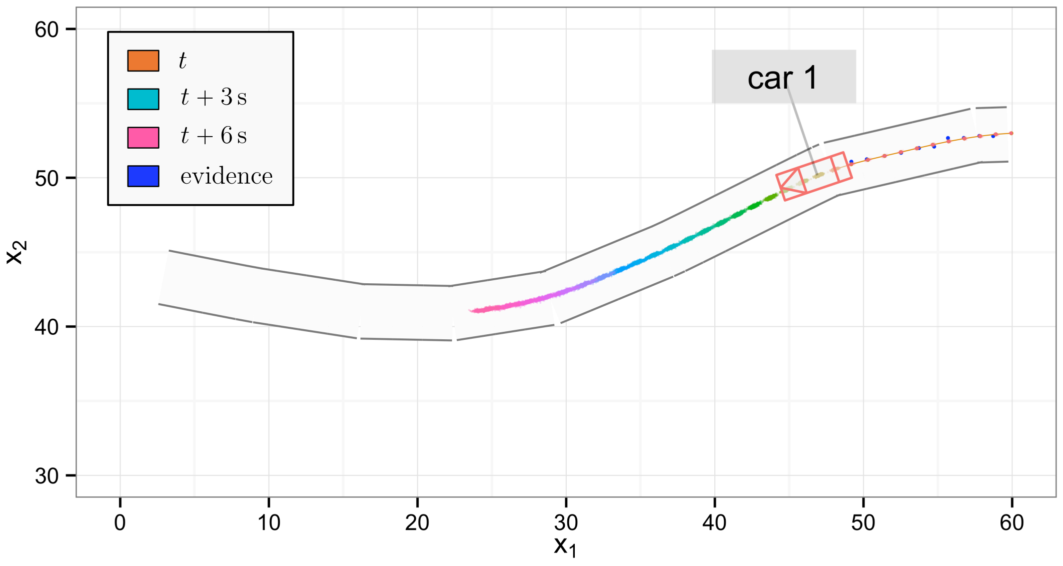
Figure 6.3: Sample episode of the lane following scenario. The numbers indicate the time steps.

Figure 6.4 and Figure 6.5 show examples of predictions generated by the two models. The prediction horizon covers 20 steps in the future, which equals a period of 6 seconds. The densities of the predicted positions are depicted in different colors to indicate the time. The orange dots behind the car show their smoothed position estimates for past time steps. The measurements are depicted in blue.

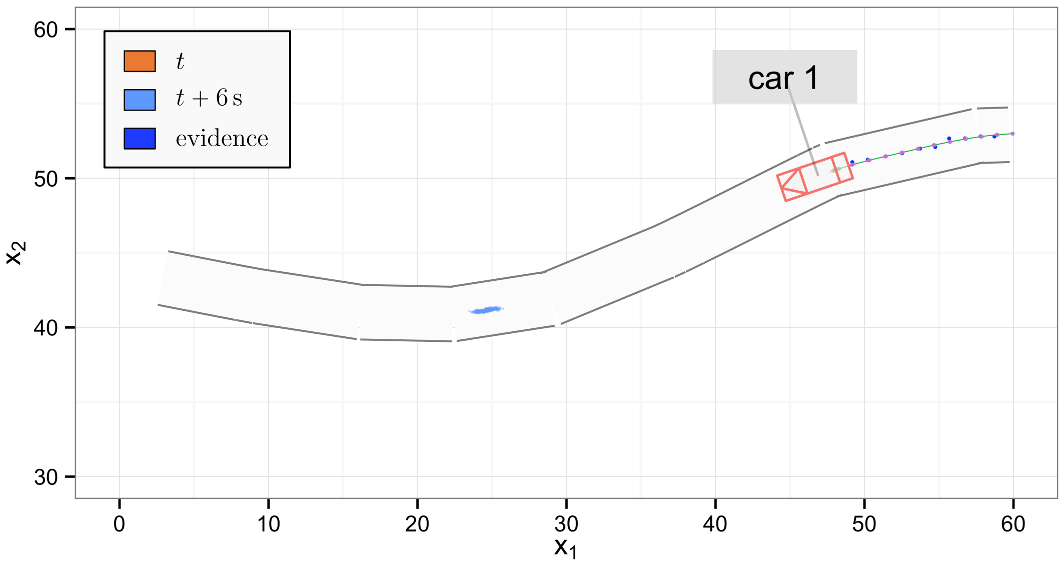
The prediction of the single track model (6.5) spreads in all directions since it uses no map data and therefore, has to consider all kinds of curvatures ahead. It is clear that no accurate prediction over longer time periods can be obtained from such a model. The learned model on the other hand is able to predict the actions well (6.4). This leads to predicted trajectories along the course of the road. The prediction uncertainty mainly spreads in the longitudinal direction to account for the different velocity profiles of drivers but is limited in lateral direction which reflects the behavior of drivers to mainly stay on lanes.

Table 6.1: Prediction MLL (higher is better).

model	MLL (Δt: 0.0 s)	MLL (Δt: 3.0 s)	MLL (Δt: 6.0 s)
learned model	1.280	0.362	-0.799
single track model	0.295	-6.537	NA

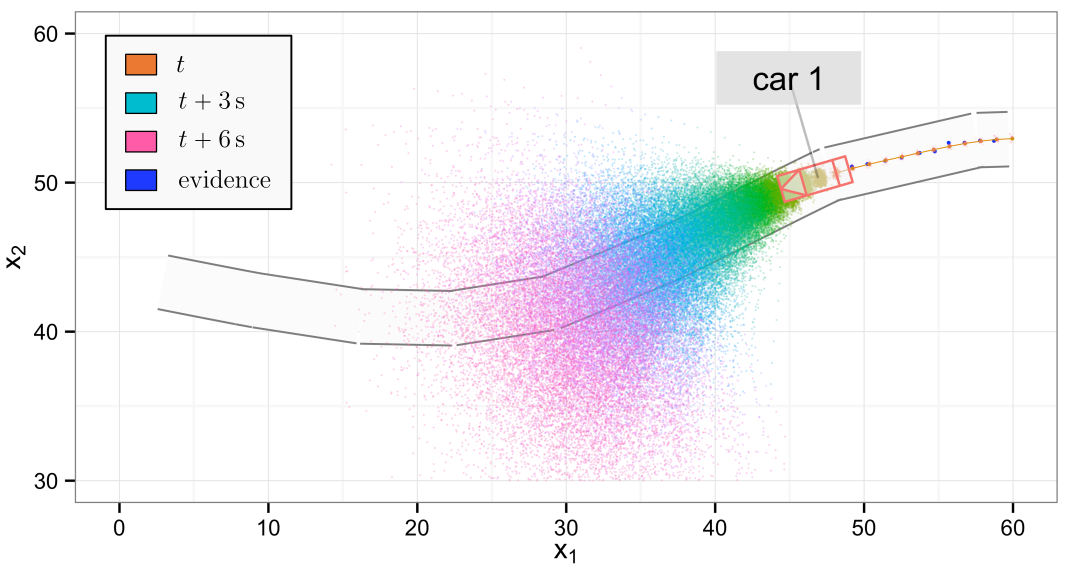


(a) Predicted trajectories of the car over 6 s.

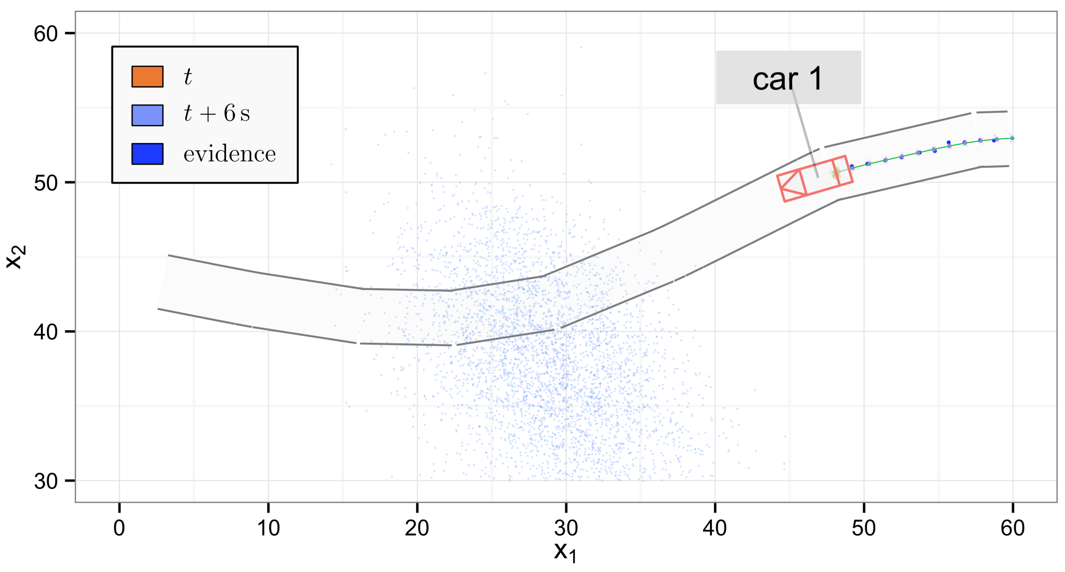


(b) Predicted future positions of the car 6 s ahead.

Figure 6.4: Learned model: Predicted trajectories of the car on the lane following track. Figure (a) shows the predicted trajectories over a period of 6 s. The trajectories follow the course of the road but vary in their velocity profile. Figure (b) shows only the prediction 6 s in the future.



(a) Predicted trajectories of the car over 6 s.



(b) Predicted future positions of the car 6 s ahead.

Figure 6.5: Single track model: Predicted trajectories of the car on the lane following track. Figure (a) shows the predicted trajectories over a period of 6 s. Since the prediction only relies on the motion history, the trajectories fan out in all directions to account for possible curves ahead. Figure (b) shows only the prediction 6 s in the future.

Table 6.2: Prediction RMSE of position (x_1, x_2) in meters (lower is better).

model	(x_1, x_2) RMSE (Δt: 0.0 s)	(x_1, x_2) RMSE (Δt: 3.0 s)	(x_1, x_2) RMSE (Δt: 6.0 s)
learned model	0.152	0.904	2.081
single track model	0.157	4.775	14.906

Table 6.3: Prediction RMSE of velocity v in seconds (lower is better).

model	v RMSE (Δt: 0.0 s)	v RMSE (Δt: 3.0 s)	v RMSE (Δt: 6.0 s)
learned model	0.143	0.457	0.527
single track model	0.108	0.784	1.111

Table 6.4: Prediction RMSE of heading ψ in rad (lower is better).

model	ψ RMSE (Δt: 0.0 s)	ψ RMSE (Δt: 3.0 s)	ψ RMSE (Δt: 6.0 s)
learned model	0.054	0.064	0.123
single track model	0.068	0.502	0.859

6.1.3 Intersection Scenario

In this scenario, the behavior of traffic participants at intersections is investigated. The intersection that is considered is a standard unsignaled intersection with 4 junctions. German traffic rules apply, i.e., drivers coming from the right have right-of-way over cars coming from the left (relative positions) and drivers that make a left turn on the intersection have to yield to other cars coming from the opposite direction. Figure 6.6 shows an overview of the intersection road map used for the evaluation and a sample episode of the two cars approaching the intersection from opposite directions.

The goal of this scenario is to evaluate whether the learning approach is able to learn the interaction patterns that occur at an intersection where right-of-way regulations lead to different behaviors. An interesting aspect is that for this experiment no right-of-way relations are provided in the context features. The system has to discover the underlying rules based on the motion observations only.

The models were learned from 60 observed episodes with an average length of 144 time steps ($\sim$43 s). An additional set of 10 episodes was used to evaluate the test set performance.

Results and analysis The results of the prediction accuracy evaluation for the two compared models is provided in the tables tables 6.5 to 6.8. As in the pre-

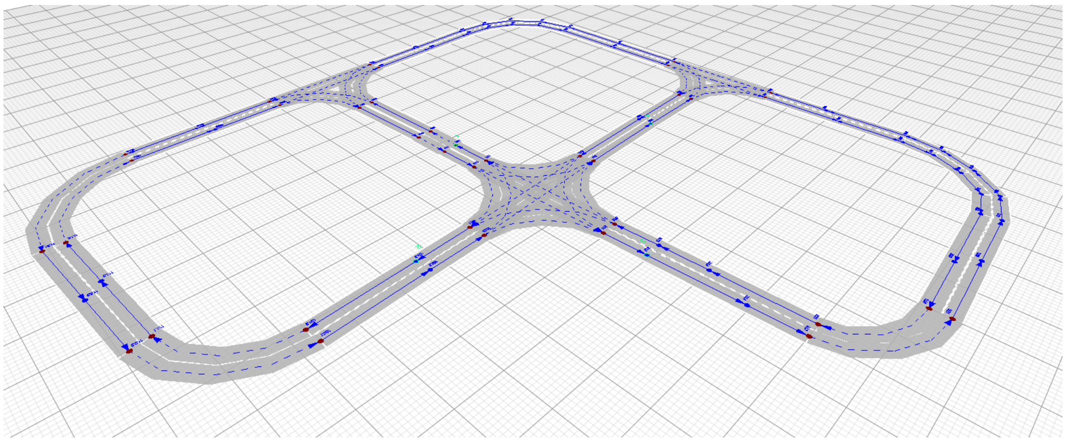


Figure 6.6: Overview of the intersection scenario. In this scenario, two cars approach the intersection from various directions and act according to German traffic rules.

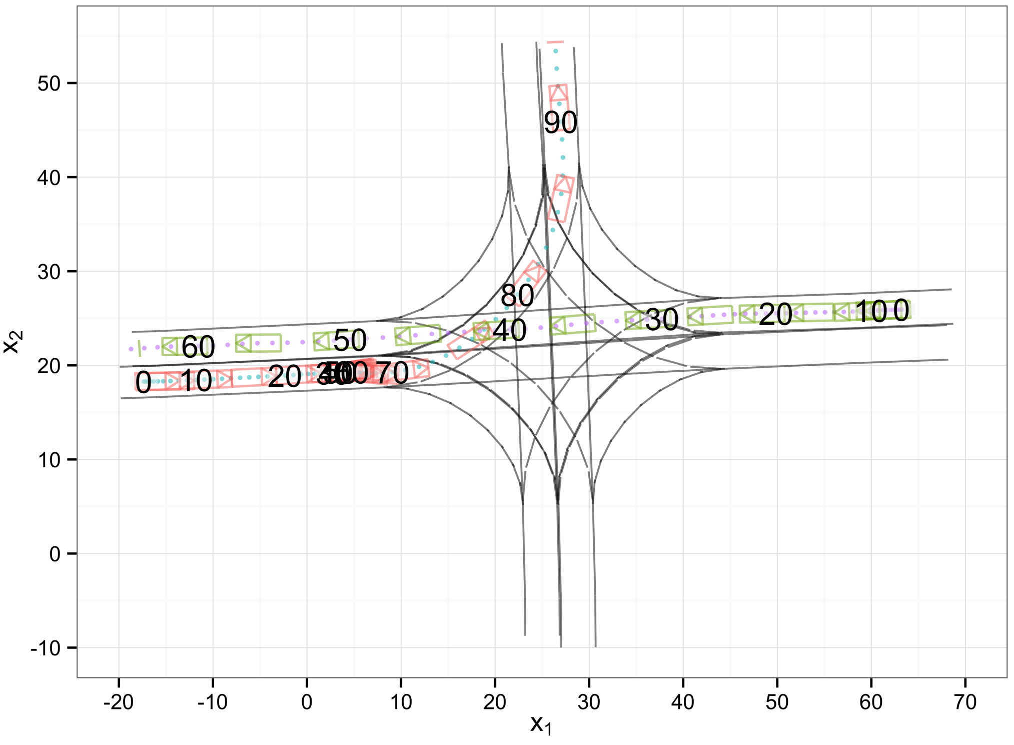


Figure 6.7: Sample episode of the intersection scenario where the car coming from left (red) yields to the car coming from the right. The numbers indicate the time steps.

vious example, the learned model yields a significantly better accuracy for the state estimation as well for the prediction. The differences become even more apparent in this scenario due to the situational dependency of the behavior patterns.

Table 6.5: Prediction MLL (higher is better).

model	MLL (Δt: 0.0 s)	MLL (Δt: 3.0 s)	MLL (Δt: 6.0 s)
learned model	1.931	-1.456	-3.655
single track model	0.577	-14.971	NA

Table 6.6: Prediction RMSE of position (x_1, x_2) in meters (lower is better).

model	(x_1, x_2) RMSE	(x_1, x_2) RMSE	(x_1, x_2) RMSE
	$(\Delta t$: 0.0 s)	$(\Delta t$: 3.0 s)	$(\Delta t$: 6.0 s)
learned model	0.142	1.662	6.327
single track model	0.161	4.664	14.209

Table 6.7: Prediction RMSE of velocity v in seconds (lower is better).

model	v RMSE	v RMSE	v RMSE
	$(\Delta t$: 0.0 s)	$(\Delta t$: 3.0 s)	$(\Delta t$: 6.0 s)
learned model	0.113	0.537	0.785
single track model	0.120	0.962	1.362

Table 6.8: Prediction RMSE of heading ψ in rad (lower is better).

model	ψ RMSE	ψ RMSE	ψ RMSE
	$(\Delta t$: 0.0 s)	$(\Delta t$: 3.0 s)	$(\Delta t$: 6.0 s)
learned model	0.053	0.213	0.432
single track model	0.072	0.487	0.797

Predictions can be seen in the following visualizations of exemplary anticipated situations. The following Figures (6.8 – 6.11) show predictions of different test episodes at different points in time. As before, the prediction horizon is set to 20 steps (6 s). The particle densities of the predicted states are depicted in different colors to indicate the time. Additionally, the predicted routes are visualized through highlighted lane boarders (green). Although the inference computes joint predictions for all cars, the figures do not show which prediction of one car belongs to which prediction of the other car for visual clarity.

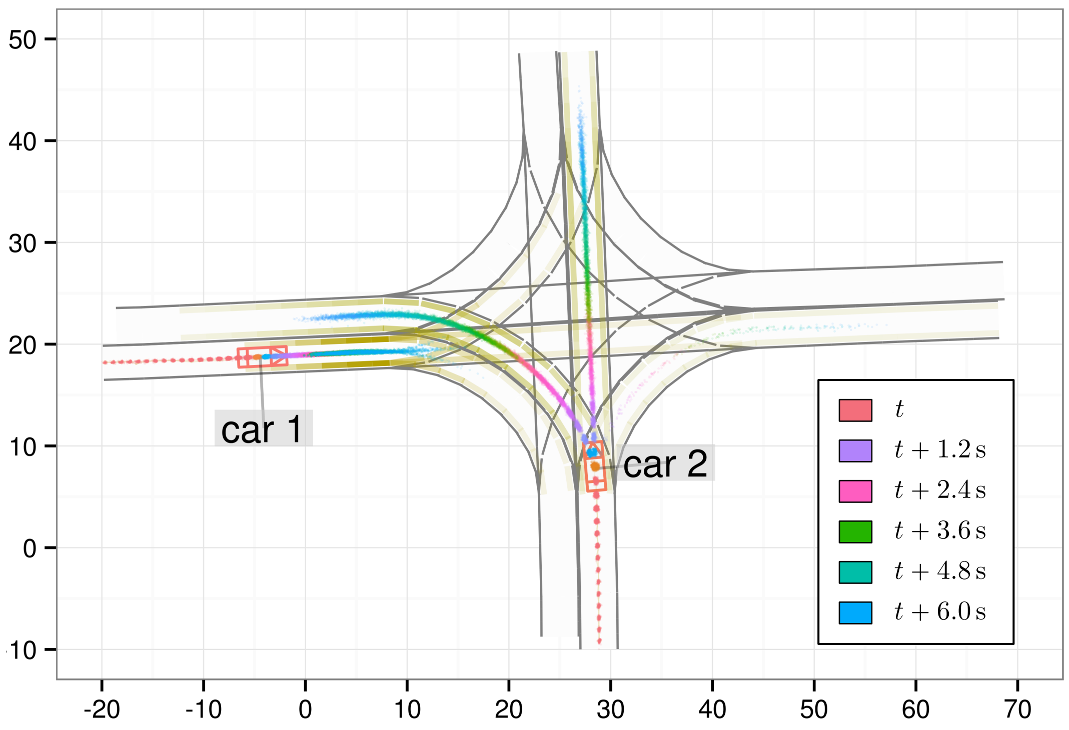


Figure 6.8: Prediction of a situation at an intersection over a period of 6 s with a policy model learned from traffic observations. Due to data-driven approach and consideration of contextual information the development of the situation is realistically predicted.

Figure 6.8 illustrates the prediction for a situation where a car (*car 2*) coming from the bottom has right of way over a car coming from the left (*car 1*). Several conclusion can be drawn from this example. Again, the learning algorithm successfully managed to learn the lane following behavior of drivers also in this more complex example. The second observation is that the drivers' different choices at the intersection lead to multiple modes in the predicted distributions. Since *car 2* has the right of way in this example, it is realistically predicted that *car 1* stops in front of the intersection while *car 2* takes one of the possible ways over the intersection without stopping. The observed driving behavior and the current heading of *car 2* together with its velocity profile leads the Bayesian model to the conclusion that not all modes are equally likely. At this point, the driver will rather make a left turn or drive straight than make a right turn.

The prediction that *car 1* is stopping at the intersection is only possible through the consideration of the whole situational context. It is necessary to take all other traffic participants and their roles into account to determine the right interpretation and anticipate consequentially that the car will most likely stop at the intersection. In addition, the model does not only conclude that *car 1* is going to stop but also provides an estimate of the distribution over where and how it is going to stop.

The influence of the situational context becomes even more evident in the example shown in Figure 6.9. In this case, only the predictions at $t + 6\,\mathrm{s}$ are plotted without the intermediate steps. Again, it can seen how the possible decisions of *car 2* lead to different clusters in the density of the predicted states. An interesting aspect is that the learner implicitly learned the effects of traffic rules on the actions of traffic participants just by observing traffic episodes.

In another example, we can see how the presented Bayesian approach with learned policy models employs temporal reasoning to make realistic predictions. In Figure 6.10, two cars approach the intersection from opposite directions. The first car coming from the left begins to slow down while the second car coming from the right keeps its velocity. The temporal reasoning integrated the observed changes in behavior over time and put them into relation with the overall situation. The conclusion of this reasoning process is that the hypotheses for *car 1* that it will make a left turn are the most likely ones. The other options became improbable over time since the observed deceleration did not match the expected behavior when driving straight across the intersection or making a right turn. Similar reasoning applied for the trajectory hypotheses of *car 2*. Since *car 2* would have to stop and yield in case of a left turn but does not exhibit a deceleration behavior, the corresponding hypotheses were ruled out.

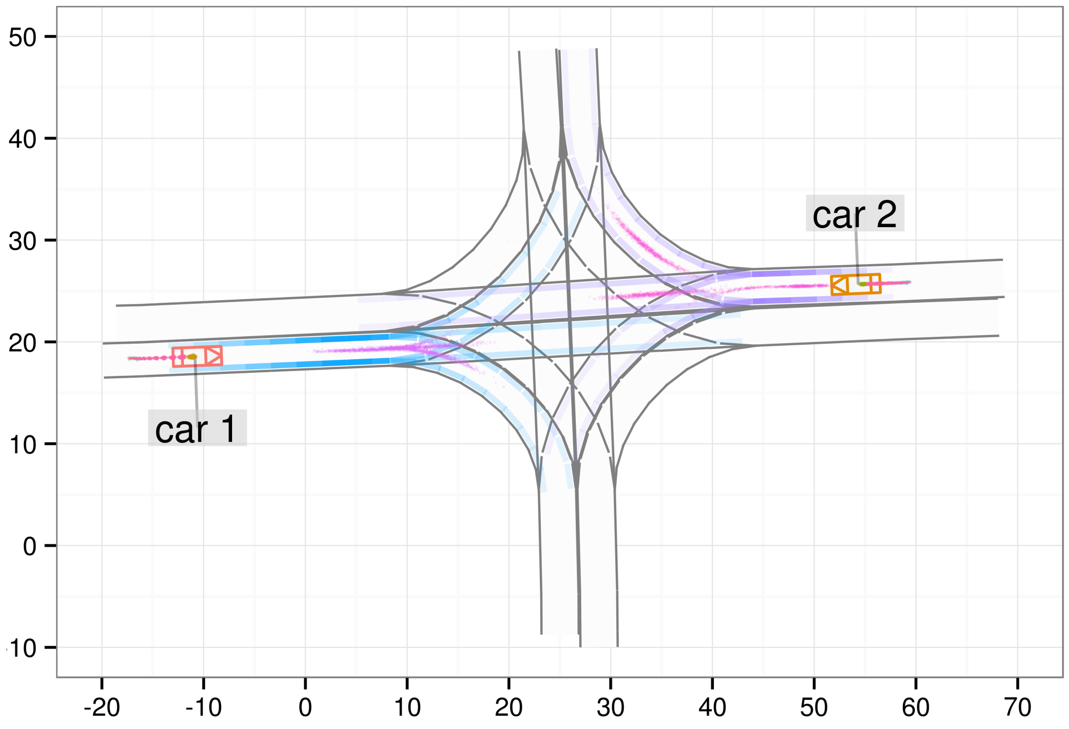


Figure 6.9: Prediction for two cars approaching the intersection from opposite directions. In this plot, only the predictions at $t + 6\,\mathrm{s}$ and not the intermediate steps are shown. One can see that the presented approach was able to learn traffic rules. As a result, the model is able to predict the different modes correctly.

Figure 6.11 shows predictions for several episodes. The episodes are typical examples that show the importance of considering the situational context when making predictions. Note, even if predictions with the same color (at the same time) overlap, it does not necessarily mean that a collision is predicted because they do not necessarily belong to the same joint prediction.

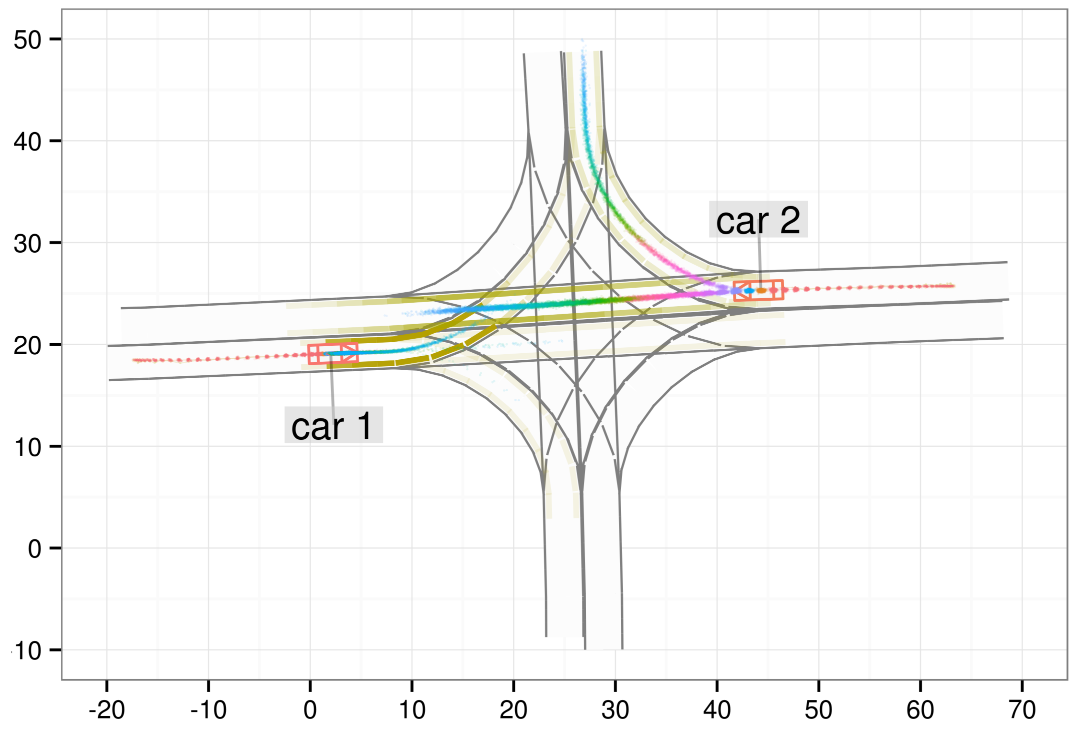
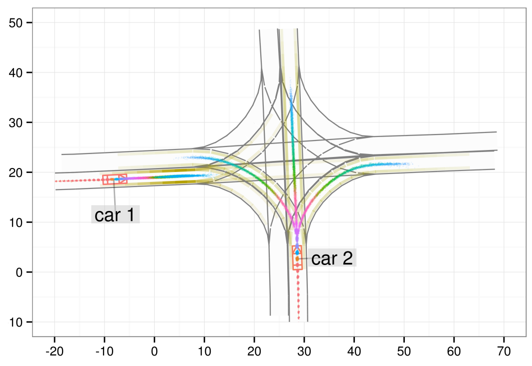
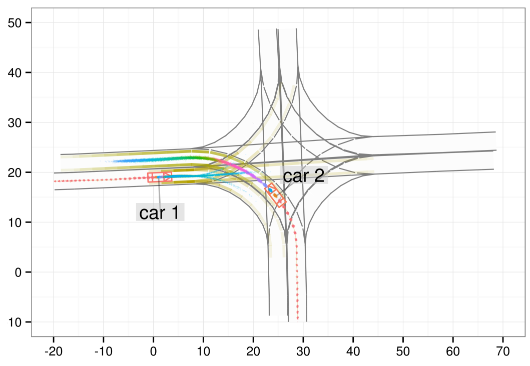


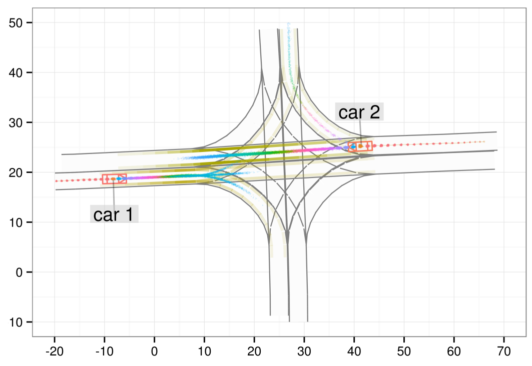
Figure 6.10: Example of temporal reasoning. The hypotheses for *car 1* that it will make a left turn are the most likely ones. The other options became improbable over time since the observed deceleration did not match the expected behavior for driving straight across the intersection or making a right turn.



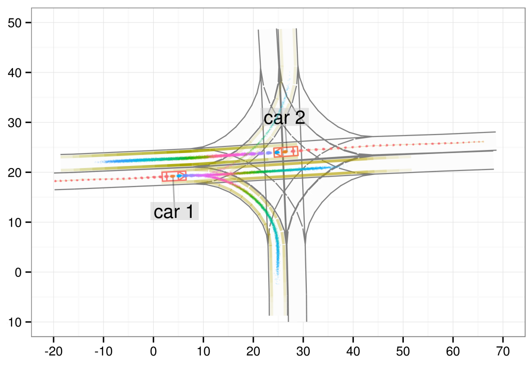
(a) Episode 1, t=25 (7.5 s).



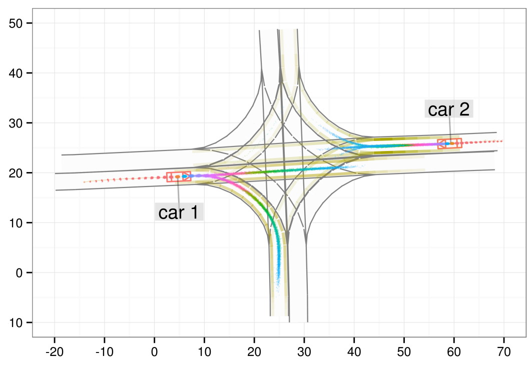
(b) Episode 1, t=35 (10.5 s).



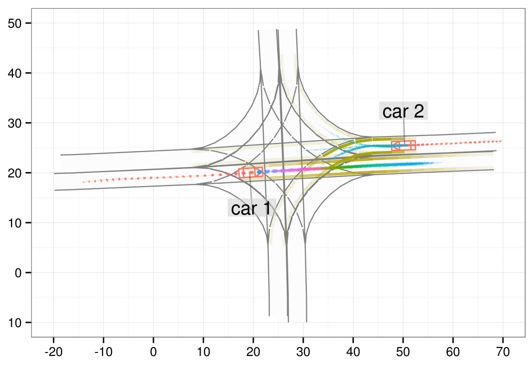
(c) Episode 2, t=20 (6 s).



(d) Episode 2, t=30 (9 s).



(e) Episode 3, t=15 (4.5 s).



(f) Episode 3, t=25 (7.5 s).

Figure 6.11: Snapshots of the prediction process for different episodes. The prediction horizon spans 6 s (20 steps). Notice the context dependency of predictions. These kind of context sensitive predictions are necessary for realistic risk assessments that go beyond regular time-to-collision estimates.

6.1.4 Roundabout Scenario

The roundabout scenario is used to examine the learning in a complex setting. The road network consists of a roundabout with roads connecting the entries and exists and a three-way intersection. Figure 6.12 gives an overview of the map and the roundabout. Three interacting cars are observed in this scenario. In the observed training examples, the cars always stop before they enter the roundabout.

The goal of this scenario is to evaluate whether the learning approach is able to learn the interaction patterns that occur at a roundabout.

The models were learned from 30 observed episodes with an average length of 600 time steps ($\sim 180\,\mathrm{s}$). An additional set of 10 episodes was used to evaluate the test set performance.

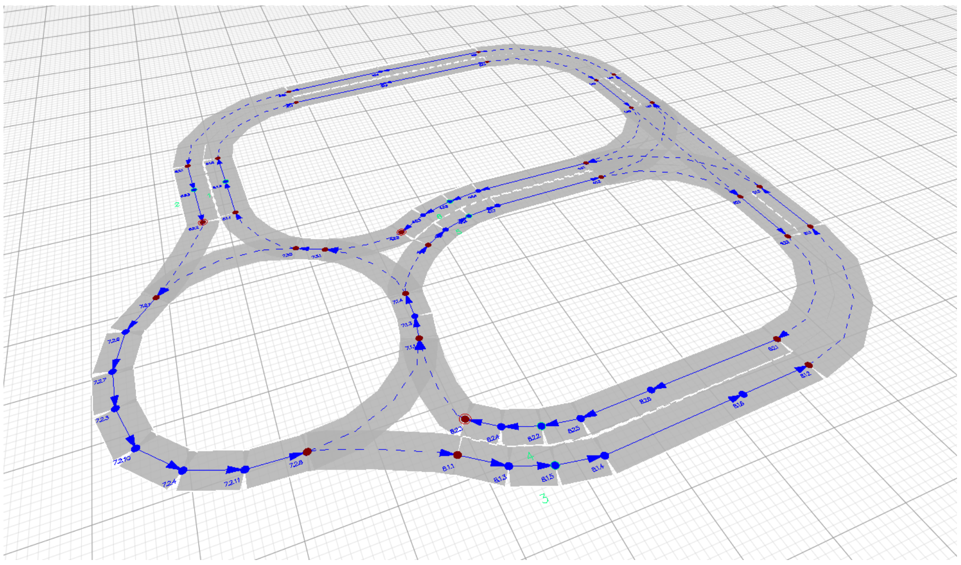
Results and analysis The results of the prediction accuracy evaluation for the two compared models is provided in the tables tables 6.9 to 6.12. As in the previous scenarios, the learned model yields a significantly better accuracy for the state estimation as well for the prediction. Due to the increased complexity of the road network configuration and context dependency of the behavior decisions of the drivers, the advantage of the learning approach over the comparison model becomes even more apparent.

Table 6.9: Prediction MLL (higher is better).

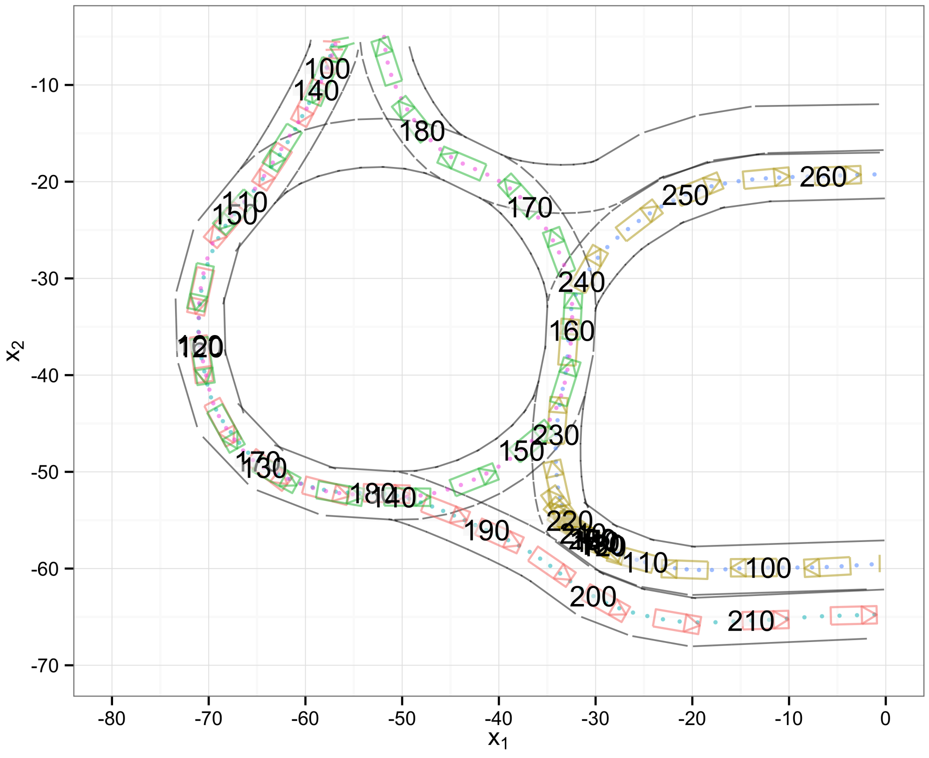
model	MLL (Δt: 0.0 s)	MLL (Δt: 3.0 s)	MLL (Δt: 6.0 s)
learned model	2.331	-6.006	NA
single track model	0.130	-21.913	NA

Table 6.10: Prediction RMSE of position (x_1, x_2) in meters (lower is better).

model	(x_1, x_2) RMSE (Δt: 0.0 s)	(x_1, x_2) RMSE (Δt: 3.0 s)	(x_1, x_2) RMSE (Δt: 6.0 s)
learned model	0.160	2.065	6.746
single track model	0.206	4.759	14.391



(a) Birds-eye perspective view of the roundabout scenario.



(b) Sample episode with 3 cars.

Figure 6.12: Overview of the roundabout scenario.

Table 6.11: Prediction RMSE of velocity v in seconds (lower is better).

model	v RMSE (Δt: 0.0 s)	v RMSE (Δt: 3.0 s)	v RMSE (Δt: 6.0 s)
learned model	0.128	0.990	1.512
single track model	0.094	1.236	1.819

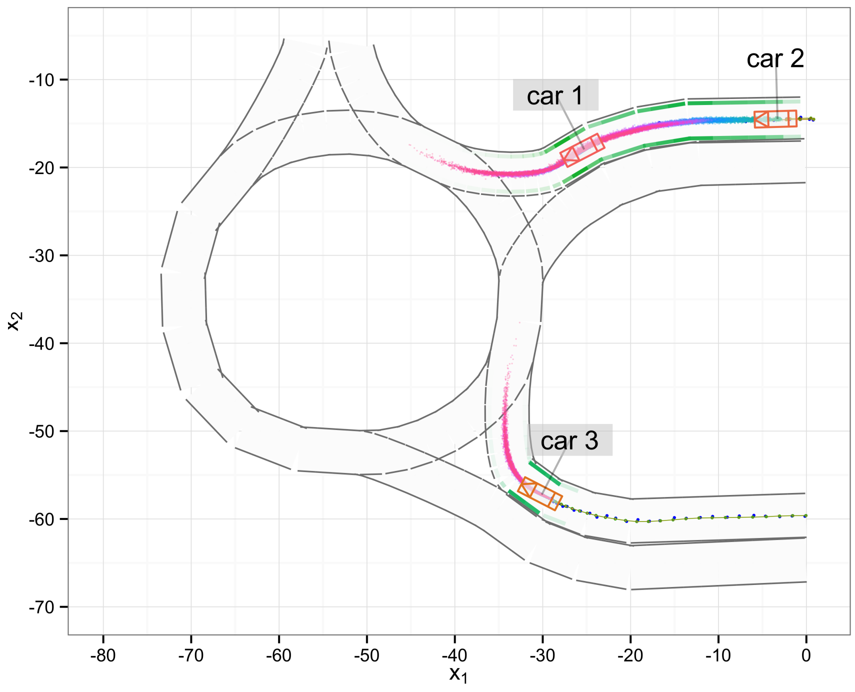
Table 6.12: Prediction RMSE of heading ψ in rad (lower is better).

model	ψ RMSE (Δt: 0.0 s)	ψ RMSE (Δt: 3.0 s)	ψ RMSE (Δt: 6.0 s)
learned model	0.060	0.218	0.487
single track model	0.085	0.562	0.940

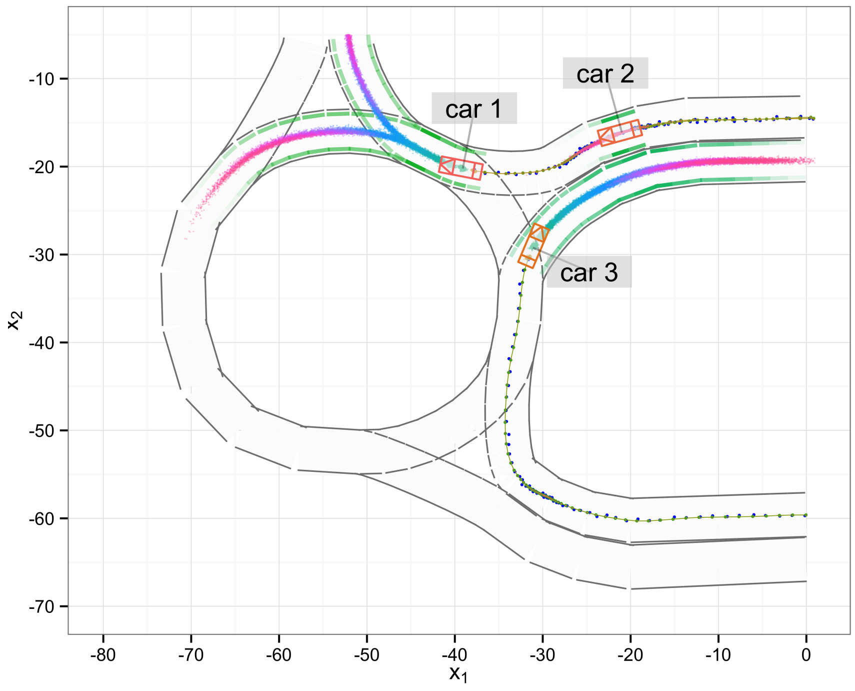
Figure 6.13 shows an example of the predicted behavior of three cars approaching and entering the roundabout. The prediction period is 6 s. In 6.13a, *car 1* and *car 3* are about to enter the roundabout while *car 2* follows *car 1*. Since *car 1* is driving slower ahead, *car 2* is predicted to slow down. 12.6 s later (6.13b), *car 3* is leaving the roundabout and *car 1* has entered it. Both choices for *car 1* of staying in the roundabout or leaving it at the next exit are predicted with equal probability since there are no indicators for one choice at this point.

Another episode is shown in Figure 6.14. It shows the prediction of a following behavior at the entry of the roundabout. At $t = 122$, *car 2* is predicted to keep a safety distance to *car 3*. Since *car 3* has no other road user ahead, it is predicted to accelerate and take one of the two possible routes with equal probability (6.14a). Figure 6.14b depicts the same situation but shows only the predicted positions 6 s ahead instead of all intermediate prediction steps. The ground truth of the future car positions is also depicted to show the accuracy of the predicted distributions.

Figure 6.15 shows the predictions of another episode at several time steps. In this episode, the context-dependency of the behavior prediction can be seen. At $t = 91$, *car 1* is in the roundabout while *car 2* is approaching the roundabout from the right (6.15a). At $t = 117$, *car 2* has slowed down and is predicted to most likely stop at the entry of the roundabout since up to this point it is unclear for *car 2* whether *car 1* will stay in the roundabout or leave it at the next exit (6.15b).

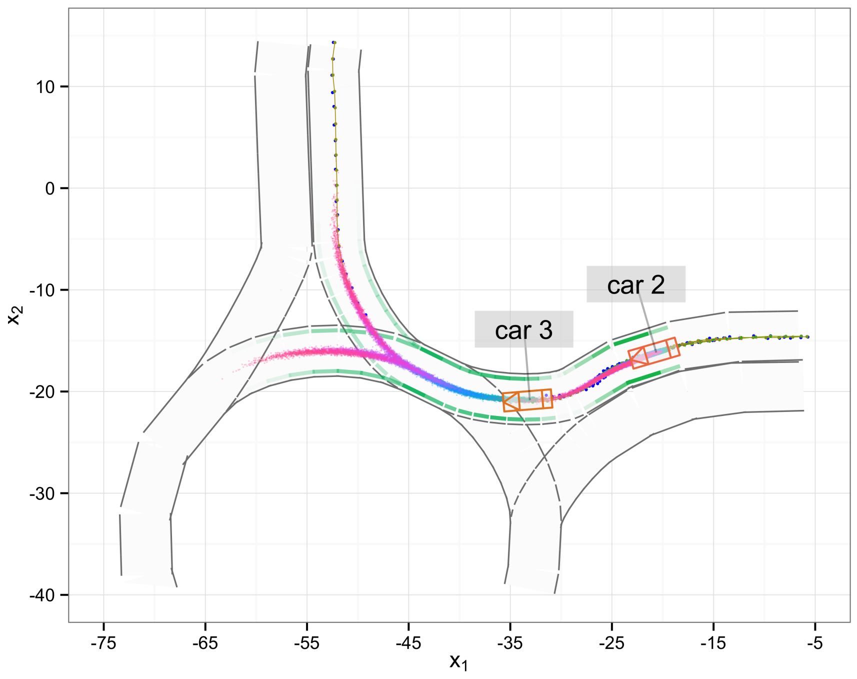


(a) Time step 80 (24.0 s).

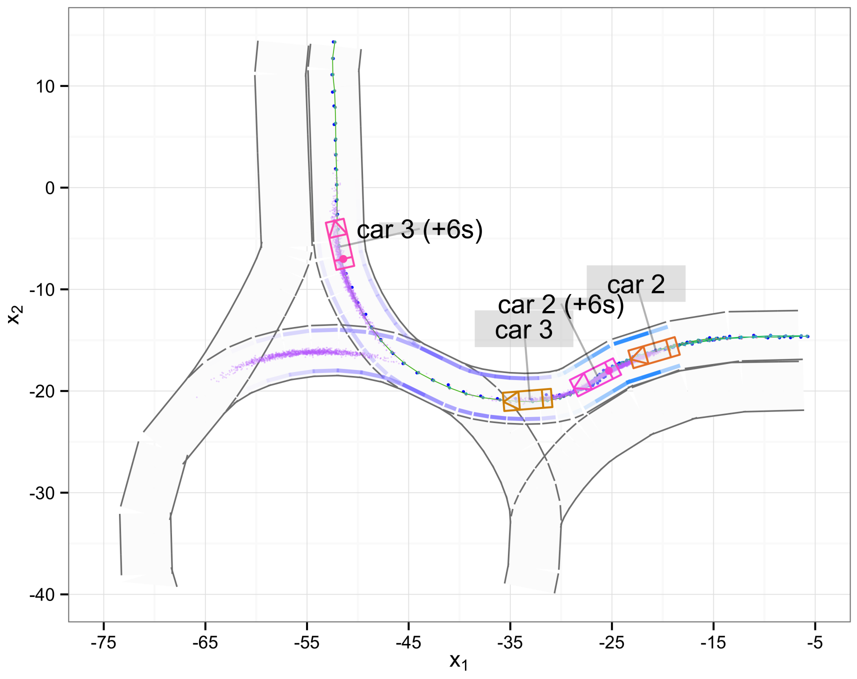


(b) Time step 122 (36, 6 s).

Figure 6.13: Predictions of three cars at a roundabout.



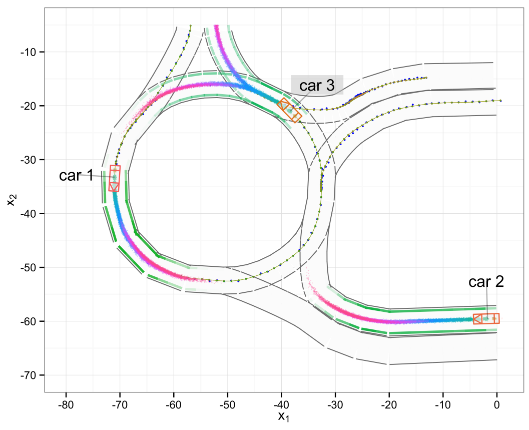
(a) Full trajectory predictions at $t = 122$ (36.6 s).



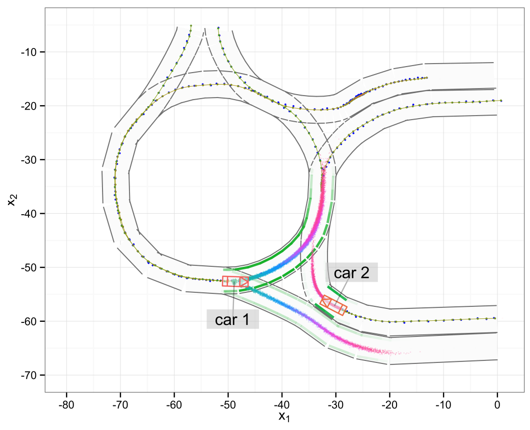
(b) Predictions 6 s ahead with ground truth at $t = 122$ (36.6 s).

Figure 6.14: Predictions of two cars at a roundabout with ground truth.

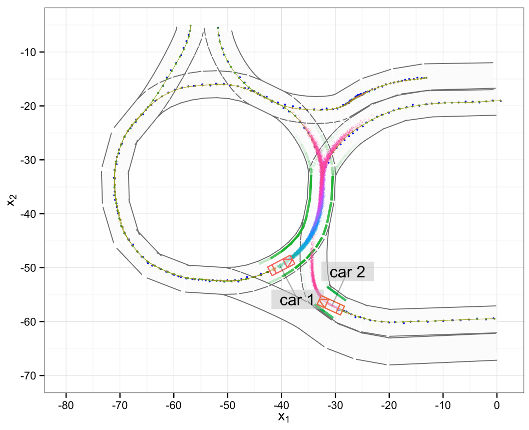
Note that at this point, the consideration of the mutual influences between road users is essential for deriving realistic predictions. At $t = 124$, *car 1* has decided to stay in the roundabout which causes *car 2* to be predicted to wait at the entry (6.15c). As *car 1* passes *car 2* and making the way free for *car 2*, it is predicted in the following time steps (140, 160, 177) that *car 2* accelerates and enters the roundabout. Notice the difference in the prediction probability of the different route choices of *car 1* and *car 2* at similar positions at $t = 140$ and $t = 177$. Due to their difference in orientation, it is more likely for *car 1* to take the exit in Figure 6.15d as it is more likely for *car 2* to stay in the roundabout in Figure 6.15f.



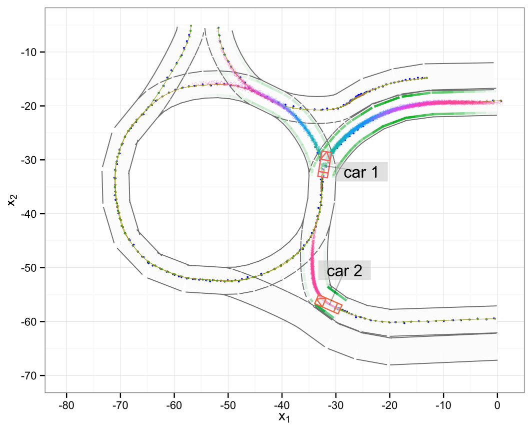
(a) Time step 91 (27.3 s).



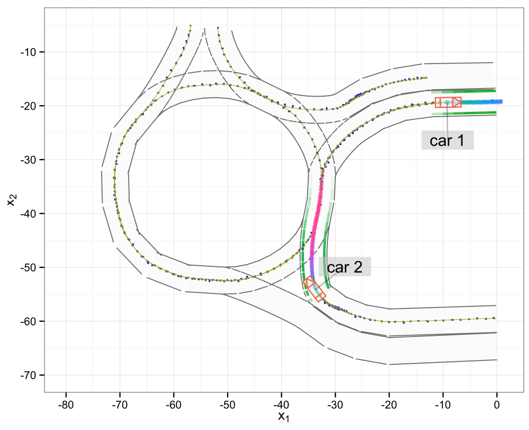
(b) Time step 117 (35.1 s).



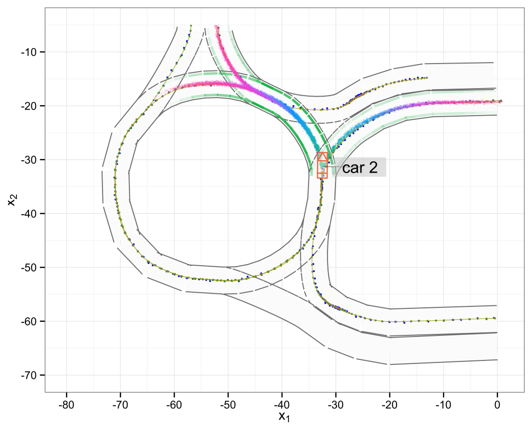
(c) Time step 124 (37.3 s).



(d) Time step 140 (42.0 s).



(e) Time step 160 (48.0 s).



(f) Time step 177 (53.1 s).

Figure 6.15: Trajectory and route predictions of three cars in a roundabout. When *car 1* is in the roundabout, it is correctly predicted that *car 2* will stop at the entry and then will wait until the way is clear.

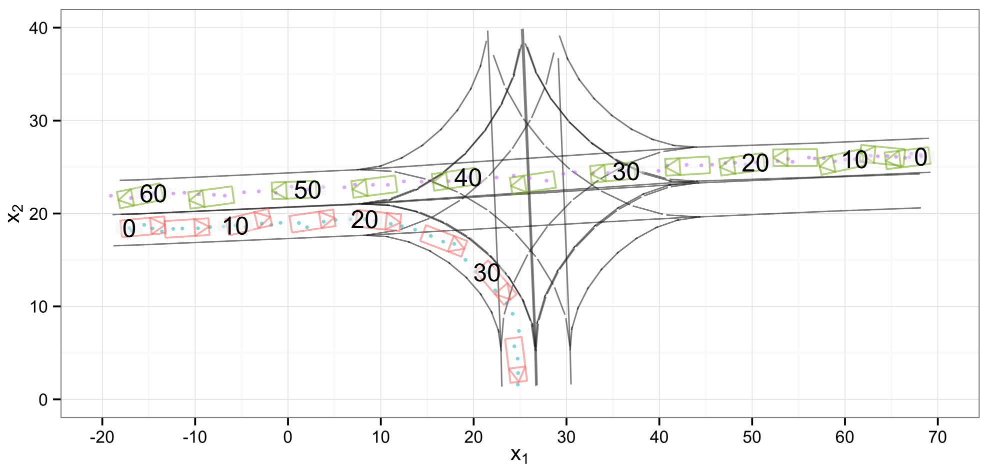
6.2 Influence of Noise

Traffic situations can only be observed with noise since sensors are imperfect. In this work several measures have been taken, such as explicitly modelling the observation noise and using an EM-based iterative learning approach, to be able to handle noise in the data. In this experiment, the influence of noise on the learning and prediction process is investigated. To evaluate the influence of noise, multiple models were trained based on data with increasing levels of noise. Table 6.13 shows the noise levels and the standard deviations for the white noise applied to the position, velocity and heading data. Figure 6.16 shows visualizations of an example episode with different noise levels. To ensure comparability, each model was trained with the same observed traffic episodes but with differing random noise. The data stemmed from observations in the intersection scenario (see Section 6.1.2). The models were learned from 60 observed episodes with an average length of 144 time steps ($\sim$43 s). An additional set of 10 episodes was used to evaluate the test set performance.

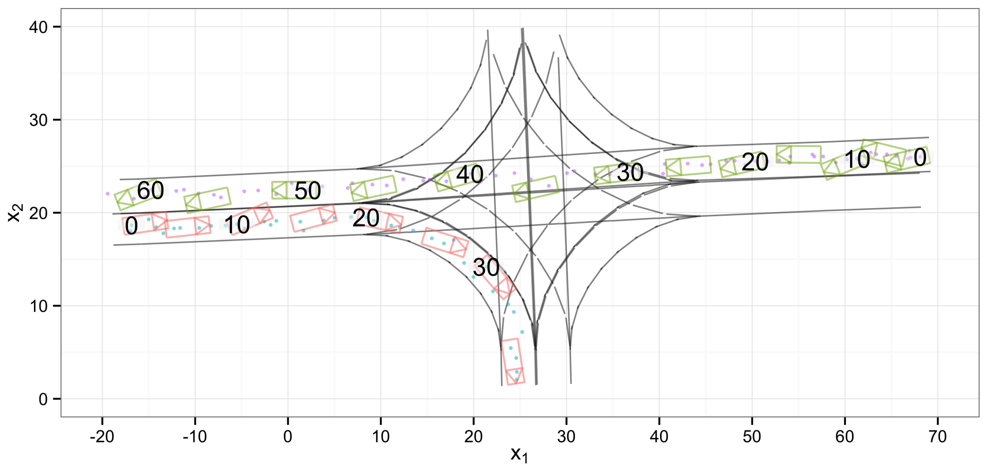
Table 6.13: Noise levels.

noise level	σ_{x_1}	σ_{x_2}	σ_v	σ_ψ
1	0.1 m	0.1 m	$0.1\,\frac{\mathrm{m}}{\mathrm{s}}$	0.05 rad
2	0.2 m	0.2 m	$0.2\,\frac{\mathrm{m}}{\mathrm{s}}$	0.10 rad
4	0.4 m	0.4 m	$0.4\,\frac{\mathrm{m}}{\mathrm{s}}$	0.20 rad
8	0.8 m	0.8 m	$0.8\,\frac{\mathrm{m}}{\mathrm{s}}$	0.40 rad

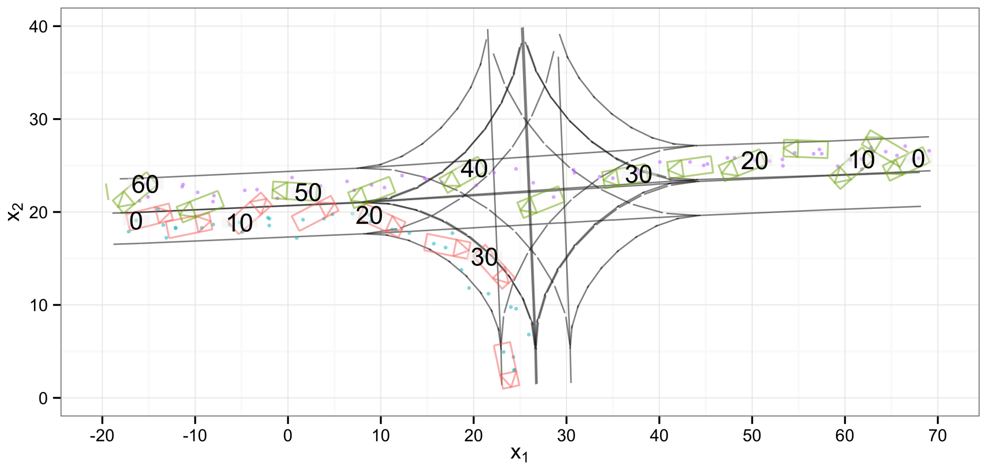
Results and analysis The noise in training data has a significant impact on the learned models. Table 6.14 reports the results for the prediction MLL. With increasing levels of noise the accuracy of predictions which are produced by the learned models decreases. Figure 6.17 depicts the MLL curve for the 6 s predictions. Due to the degradation of the signal-to-noise ratio with increasing noise in the data, it becomes more difficult for the learner to identify behavior patterns in the data. This implies that more data is needed for learning models of the same accuracy when the noise in the data increases. Figure 6.18 shows the resulting predictions of the learned models for an intersection traffic situation. As can be seen, the predictions get more uncertain and less accurate with



(a) Data with noise level 2.



(b) Data with noise level 4.



(c) Data with noise level 8.

Figure 6.16: Predictions over 6 s of the learned models for increasing levels of noise. With higher levels of noise the predictions get more uncertain and less accurate.

higher levels of noise. Even though the prediction accuracy decreases with increasing noise, the approach was still able to learn basic behavior patterns such as lane following at the highest investigated noise level. Models that are more uncertain in their predictions also have an impact on inference. With higher uncertainty present in state estimation, more samples (particles) are needed to obtain reliable Monte Carlo estimates (MacKay, 2003).

Table 6.14: Prediction MLL for increasing levels of noise (higher MLL is better).

model	noise level	MLL (Δt: $0s$)	MLL (Δt: $3s$)	MLL (Δt: $6s$)
learned model	1	1.931	-1.456	-3.655
learned model	2	0.544	-2.613	-4.858
learned model	4	-3.777	-6.101	-7.618
learned model	8	-7.124	-6.708	-15.884

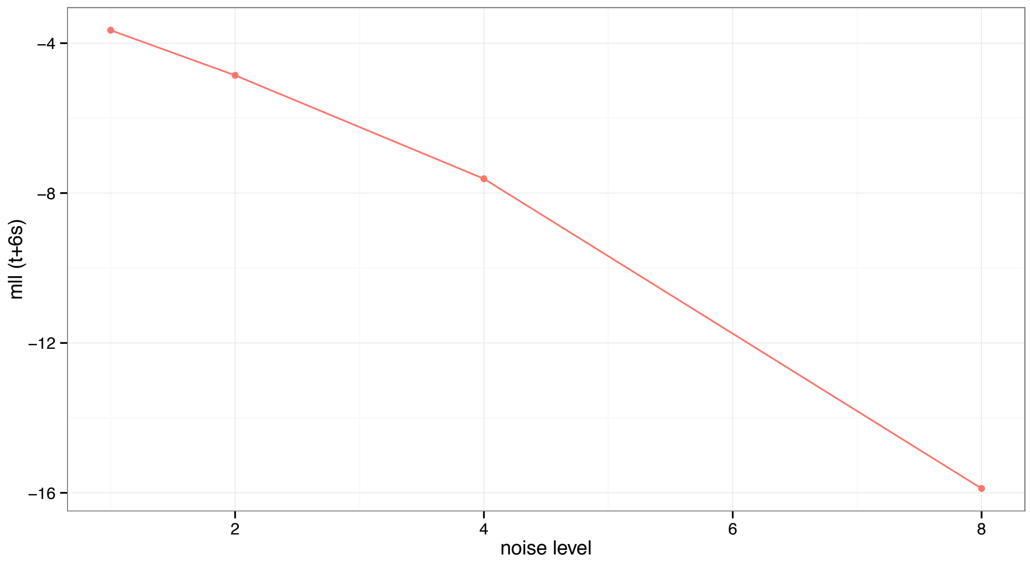
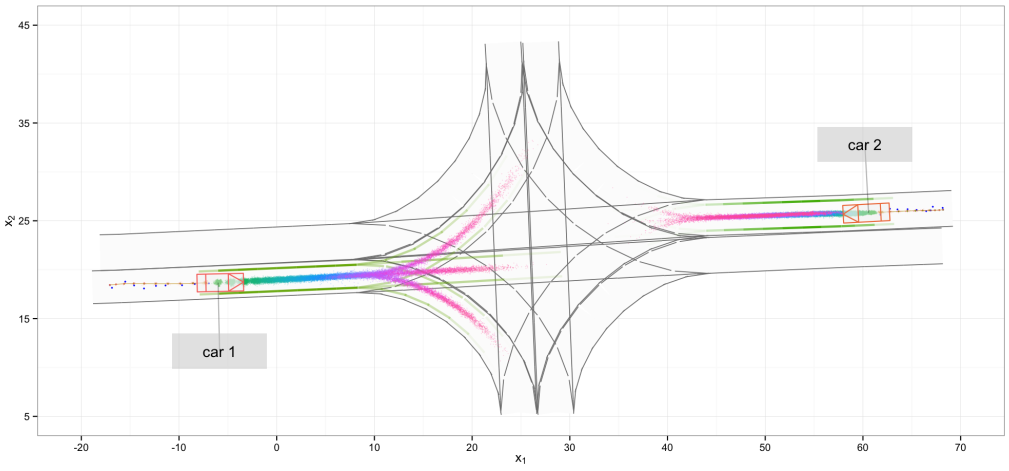
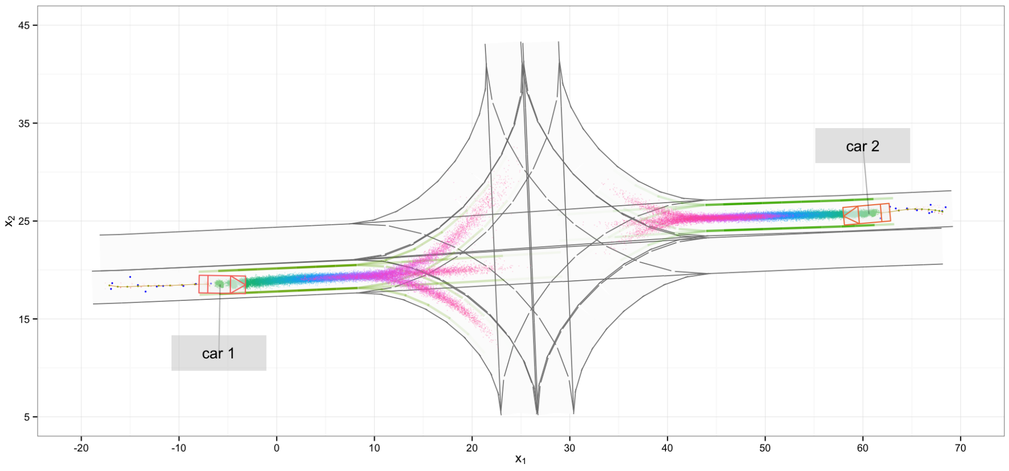


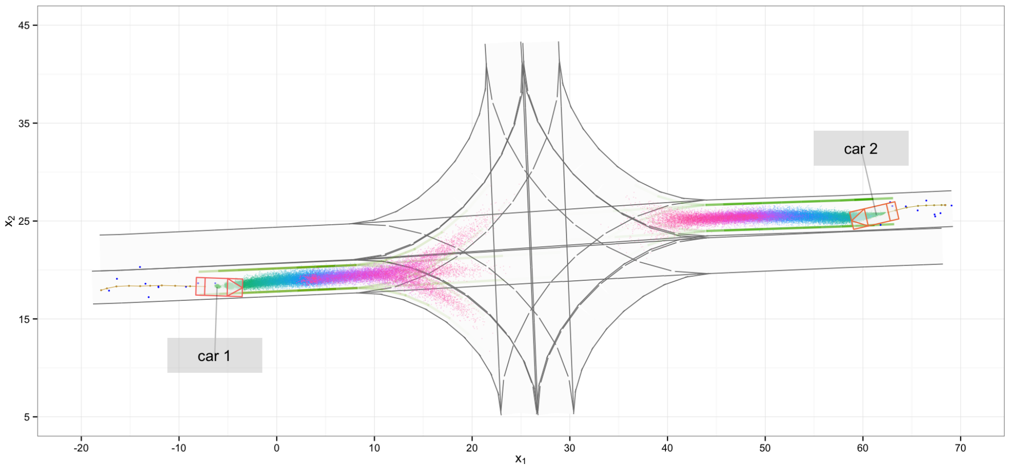
Figure 6.17: MLL of predictions $\Delta t = +6\,\text{s}$ for different levels of noise.



(a) Predictions of model learned at noise level 2.



(b) Predictions of model learned at noise level 4.



(c) Predictions of model learned at noise level 8.

Figure 6.18: Predictions over 6 s of the learned models for increasing levels of noise. With higher levels of noise the predictions get more uncertain and less accurate.

6.3 Summary and Conclusion

In this chapter, the properties and potentials of the presented learning approach to traffic prediction were investigated and analysed. The prediction accuracy was evaluated for different traffic scenarios, which exhibited different difficulties to the learner. It was shown that with the hierarchical Bayesian model and policy models learned from traffic observations realistic predictions over time periods of several seconds can be obtained. The learning approach successfully managed to identify the situational dependencies on the action choices of road users in the different episodes. The right of way rules are learned solely by observing traffic in the intersection scenario. The comparison to a standard Bayesian filter with a single track model underlined that the consideration of the mutual influences between traffic participants are crucial for making accurate predictions.

We also analyzed the influence of noise on the learning and prediction process. As expected, the signal-to-noise ratio has a strong impact on the learned models and the prediction accuracy. With increasing levels of noise the learning task becomes more and more difficult and increases the amount of data needed to obtain models of the same accuracy. A practical consequence of this finding is that it is worth investing in good sensors and processing technology for the development of learning cars that are able to learn behavior patterns quickly.

7 Discussion and Outlook

Traffic participants act in a highly dynamic and only partially observable environment. The behavior of traffic participants is highly coupled and uncertain. How a driver acts strongly depends on a situation and his intentions. All these properties make the prediction of traffic situations a challenging task. Being able to anticipate situation developments is a prerequisite for a lot of applications in the traffic domain such as the realization of self-driving cars, next generation advanced driver assistance systems and traffic surveillance systems.

In this thesis, we showed that the evolution of traffic situations can be accurately predicted by reasoning about the decision making of traffic participants and their interactions. By taking their perspectives, it is possible to draw conclusions about their goals, plans and actions and take their mutual influences into account. Modeling the dependencies between situations and behavior decisions is difficult due to the manifold of possible situations and the stochastic nature of human behavior. We therefore proposed learning these complex dependencies from traffic observations.

This approach was realized in form of a novel Bayesian model for estimating and predicting whole traffic situations. We derived a learning algorithm to obtain the policy models of traffic participants from incomplete data. To ensure the feasibility of the learning approach, we introduced novel machine learning methods for decision trees with improved generalization capabilities. The presented approach and methods have a wide range of applications beyond the traffic domain. In the following, we summarize the results covered by this work.

State estimation and prediction of traffic situations We presented a novel Bayesian model for describing the evolution of traffic situations as a stochastic process (Chapter 3). By resembling the decision making of road users on several abstraction levels, conclusions about their resulting behavior are drawn and used to predict the situation development.

In contrast to existing approaches, goals, plans and actions of traffic participants are simultaneously estimated in a unified probabilistic framework under consideration of context-dependent mutual influences. This allows fine-grained predictions on the level of continuous dynamics as well as long term predictions on the level of routes and goals. Compared to approaches that only consider one specific abstraction level, the hierarchical approach can yield more accurate predictions. The reason is that the flow of information between the different layers can improve the predictions at each level of detail. For calculating the state estimation and prediction posteriors, suitable sequential Monte-Carlo inference methods for the Bayesian model based on likelihood weighting were derived.

Applications such as decision making or motion planning for autonomous cars rely on the ability of anticipating future situation developments over a planning horizon of several seconds. They can thus profit from the improvement in prediction accuracy achieved by the presented approach over state-of-the-art methods. To concretize this aspect, we showed how the presented Bayesian model can be embedded as a process model into (partially observable) Markov decision processes to derive behavior decision or solve motion planning task in the traffic domain.

We evaluated the presented approach in several different traffic scenarios such as lane following, intersection and roundabout scenarios (Chapter 6). The experiments showed significant improvements in the accuracy of predictions over standard Bayesian filter approaches that do not consider situational context dependencies.

A direction for future research is the differentiation of traffic participants into various types such as cars, trucks, bicyclists and pedestrians. The use of individualized policy models would allow considering their type specific idiosyncrasies and further improve the prediction accuracy in complex traffic situations.

The complexity of considering the interactions between road users increases with their number. In future research, methods should be investigated that allow reducing the computational demand in situations with many road users involved. One way to achieve this is to consider only road users that are relevant for the individual decision making.

Policy model learning from observations The key components of the Bayesian model we presented in this thesis were the policy models. They describe the relationships between situations and the probable actions road users are going to conduct. In the proposed hierarchical setting, this is not only modeled on the level of actions such as accelerating or steering but also on the more abstract level of routes that road users are likely to choose and intermediate goals which they pursue. The resemblance of the decision making allows modeling the mutual influences of road users and their common behavior patterns.

Formulating the policy models manually is difficult and error-prone. In contrast to previous approaches, we therefore pursued a data-driven approach to learn the policy models of road users from traffic observations (Chapter 4). We used a combination of machine learning methods and domain-specific knowledge to learn the complex non-linear relationships between situations and decisions and to maximize generalization. The learning algorithm is based on a Monte Carlo expectation maximization scheme to make learning feasible in the incomplete data setting. The incomplete data is caused by the fact that important variables, for instance, the goals and plans of road users, cannot be observed.

We developed a non-parametric learning procedure based on random forests for learning the action policy model. The learning procedure considers input-dependent noise and uses efficient basis function estimators for performing the substeps of decision tree induction. The non-parametric learning technique adapts the model complexity to the data, which is important to minimize the risk of over- and underfitting.

In comparison to approaches that learn classifiers for behavior recognition, the presented approach does not require manually labeled data. This renders the learning cost-effective. In combination with non-parametric learning, it makes the learning process scalable. New experiences in form of situations that are observed for the first time can be easily integrated.

The generalization of the learning approach was leveraged by incorporating domain-specific knowledge in form of map data, motion models, expressive context features, traffic regulations and conditional independencies. This makes it possible to transfer the experiences to similar traffic situations, which has also been demonstrated in the evaluation of the approach.

In order to further improve the generalization of the policy models, future research can investigate to learn additional context features that describe situational aspects which are relevant for the decision making of road users. Deep learning methods are promising candidates for approaching the feature learning problem (Arel et al., 2010; Bengio, 2009). By introducing additional hidden variables that represent memory states with no predefined semantics, temporal features could be learned that integrate relevant information over time.

Another promising direction for future work is life-long and collaborative learning. Self-driving cars would benefit if they were able to learn from new experiences as they drive. An even larger potential has the exchange of new experiences between systems. This is expected to increase the learning rate massively by parallelization. More importantly, it would produce more robust models since even rarely occurring situations would be considered.

Machine learning methods As a general contribution to the field of machine learning, we derived a new class of decision trees called generalized decision trees, which generalize common tree models such as classic decision trees, hierarchical mixtures of experts and fuzzy decision trees (Chapter 5). In difference to classic decision trees, which use discontinuous split functions, generalized decision trees employ smooth split functions. This improves the representation capabilities. Additionally, it renders the tree function differentiable and enables gradient-based learning.

We derived a gradient-based learning algorithm that enables the simultaneous optimization of all split and basis functions of generalized decision trees. As shown in the evaluation, this improves generalization to unseen data because better split functions are discovered. This property becomes increasingly important in high-dimensional learning problems, where the parameter space of split functions cannot be searched exhaustively. Suboptimal split function parameter estimates of greedy induction methods can also be corrected with the presented methods by post-optimizing all tree parameters.

In difference to classic decision trees with hard splits, generalized decision trees have an increased complexity for function evaluation since all paths in a tree have to be evaluated. However, some functions can be represented more efficiently with generalized decision trees so that the smaller tree sizes overcompensate this property.

In order to preserve the efficiency of function evaluation, which is crucial for Monte Carlo inference, we developed a second learning algorithm for decision trees with hard splits. It uses the concepts of generalized decision trees in the framework of continuation methods. A newly developed parametric loss function allows to gradually optimize the parameters of the split and basis functions. As in the case of generalized decision trees, the generalization is improved by finding better split functions. Both presented learning methods achieved significantly improved performances on the tested data sets over state-of-the-art decision tree induction methods.

The developed algorithms are general machine learning methods for solving classification and regression tasks beyond the traffic domain. The incremental nature of the learning algorithm for generalized decision trees enables tackling learning problems for which large amounts of data are available.

The presented methods can be directly combined with incremental non-parametric tree learning techniques (see for example (Ikonomovska, 2012)). Future research should investigate the implications and the potential gains in generalization that could be achieved by such an approach.

7.1 Conclusion

It has been shown that highly accurate predictions of traffic situations can be made with learned models that resemble the decision making of traffic participants. The combination of machine learning methods with domain-specific knowledge and the consideration of interactions between traffic participants yields predictions of unmatched accuracy and is able to generalize to a broad set of traffic situations. The methods developed in this thesis open new possibilities in a number of applications. Especially the decision making of autonomous vehicles can benefit from these findings since it heavily relies on the ability to anticipate future developments. We believe that learning based approaches have a high potential beyond the topics addressed in this work.

A Appendix

A.1 Digital Maps

An important source of information for ADAS are digital maps. They provide topological and geometric information about the route network in a specific area and are often enhanced with additional information like points of interest or the current traffic density. While they are currently mostly used for navigation, we expect future ADAS systems to make heavy use of digital maps.

Digital maps for automotive applications come in different sizes and shapes with varying level of details. They are mostly represented as annotated graphs. Besides commercially available maps that are tailored for specific purposes, there also exist freely available maps. One of the most popular projects to provide free maps to is the OpenStreetMap (OSM) project (OpenStreetMap, 2013;

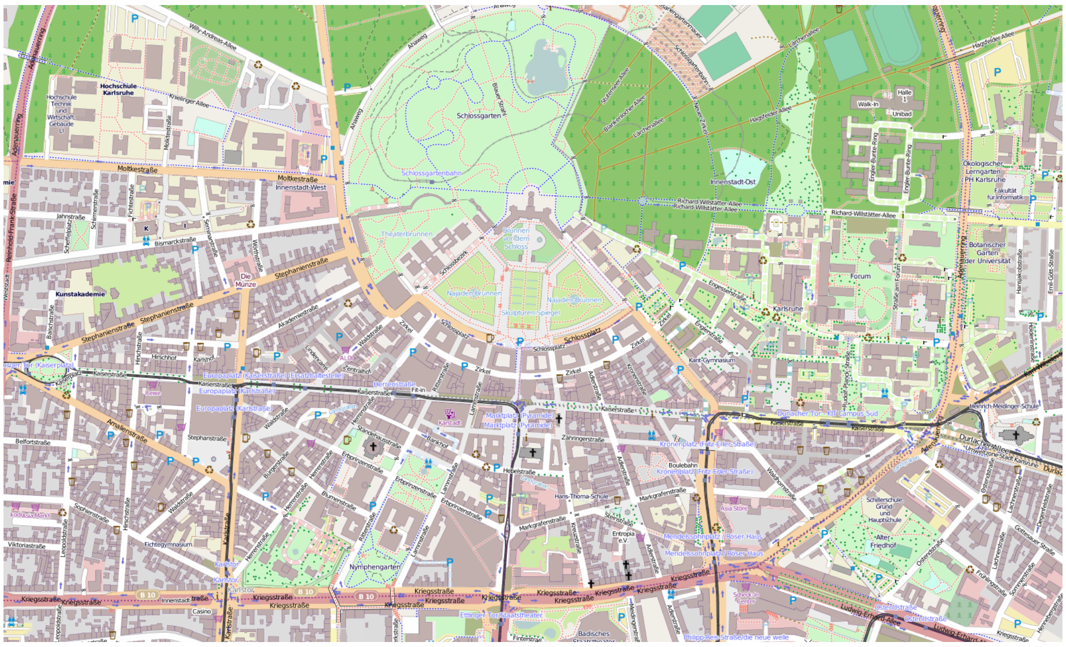
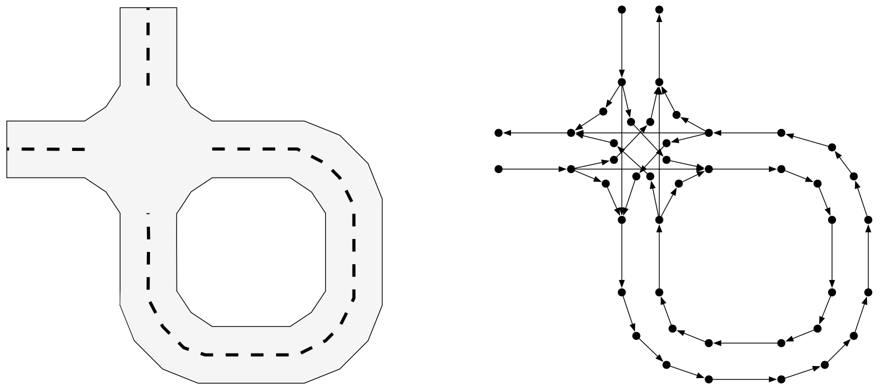


Figure A.1: Digital map of the central area of Karlsruhe from open street map (OpenStreetMap, 2013).



(a) Road network. (b) Route network graph.

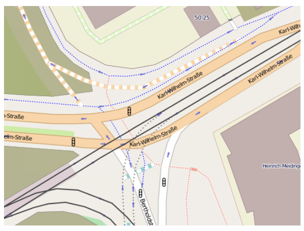
Figure A.2: Example of a road network and the corresponding graph representation.

Haklay and Weber, 2008; Neis et al., 2011). OSM is an open source project where the maps are extended and updated by a active user base all over the world. While the OSM standard allows in principle to specify highly precise lane geometries, the level of detail in practice often ends at the level of roads. Since the presented approach makes use of lane information, we build upon a graph representation that was first introduced in the DARPA Grand Challenge and then later extended for the DARPA Urban Challenge to be able to represent basic traffic scenarios including unsignalized intersections, highways and parking lots (Darpa, 2007).

The graph representation is very similar to the OSM representation. The vertices of the graph represent geospatial points in a global coordinate system. The edges of the graph define individual lane segments or boundaries of areas. Vertices as well as edges are tagged with additional attributes to define e.g. the width of a lane, speed limits or the type of lane markings. The direction of an edge indicates the intended driving direction. Edges are allowed to overlap which is necessary to describe the multiple ways in intersections. Figure A.2 gives an example of a simple intersection and the corresponding route network graph.

The annotated route network provides detailed information about the lane geometries and the relationship between lanes. This information and the conclusions that can be drawn from them are essential for a driver to navigate safely and according to the traffic rules. An artificial system can use graph search techniques to derive the right of way relations at an intersection or to detect if there

(a) Aerial image (City of Karlsruhe, 2010).


(b) OSM representation (OpenStreetMap, 2013).

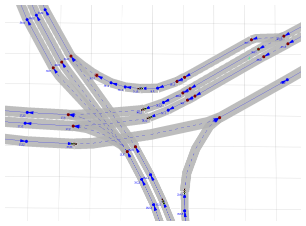

(c) Route network graph.

Figure A.3: Comparison of map representations of a part of the *Durlacher Tor* in Karlsruhe.

exists an opportunity to overtake a slower driving vehicle in front. By detecting overlapping regions, a system can derive possible areas of conflict (Schröder, 2009). Figure A.3 shows a region of the *Durlacher Tor* in Karlsruhe (Germany) and the corresponding route network in OSM and our graph representation.

A.2 Expected Utilities of Routes with Finite Planning Horizon

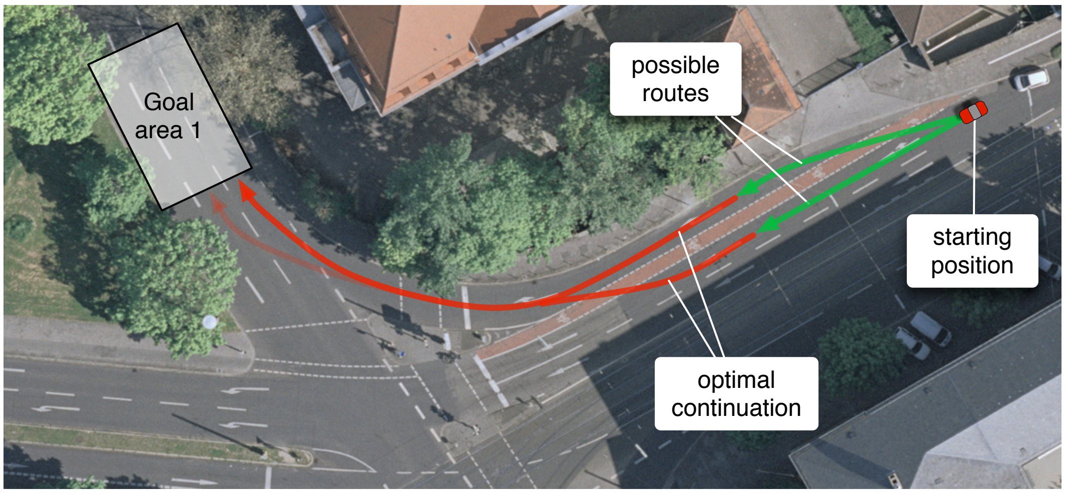


Figure A.4: Routes can end before the goal is reached, due to the finite planning horizon. In this case the utility of the missing part is estimated based on the utility of the optimal continuation of a route to the goal. (Aerial image provided by (City of Karlsruhe, 2010)).

Due to the finite planning horizon, the routes normally end before reaching the goal area. To make utilities of routes comparable, utilities need to be estimated for the whole route to the goal. Otherwise, it could happen that the utility of a route is very high but ends at a lane segment from where only paths to the goal with very low utilities exist. In such cases, the true utility of the whole route would be over- or underestimated. To estimate the utility of the missing part, we use the utility of the optimal continuation route $\hat{\pi}^R$ from the last segment of the route π^R to the goal (Figure A.4). The utility of $\hat{\pi}^R$ is defined analogously, but with the difference that the context is no longer considered in the reward function r

$$q_g(\pi^R) = \sum_{l=1}^{|\hat{\pi}^R|-1} r(seg(\hat{\pi}^R, l), seg(\hat{\pi}^R, l+1)) \,. \tag{A.1}$$

The function builds only upon the static properties of the transitions and lane segments since the situational context does not provide much information for the route continuation past the planning horizon due to the uncertain development. Hence it can be precalculated.

The total expected utility of a route policy therefore consists of two parts: The situation-dependent utility for the route until the planning horizon and the continuation of the route to the goal

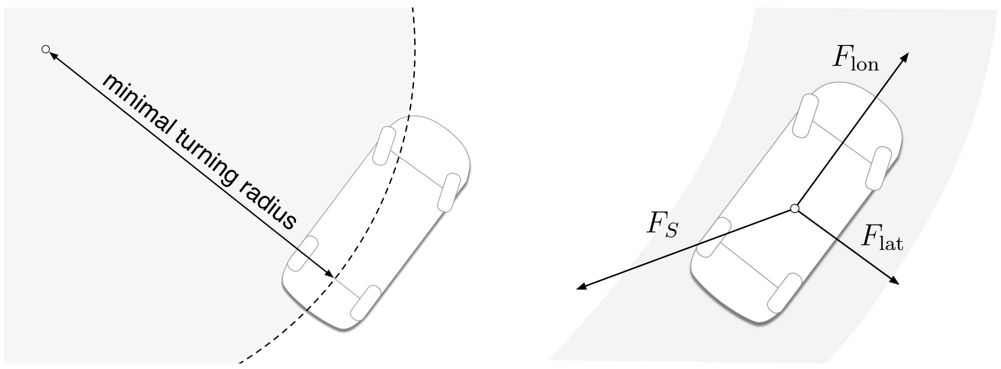
$$q(\pi^R, s^R) = q_s(\pi^R, s^R) + q_g(\pi^R)\,. \tag{A.2}$$

For a known road network, the optimal routes from each lane segment to a goal region can be efficiently calculated using dynamic programming with the well known Dijkstra algorithm (Dijkstra, 1959; Zhan and Noon, 1998). To apply Dijkstra's algorithm, the problem of finding the optimal route is cast into a shortest path graph search problem. By negating the rewards, the expected utility of a route is described in terms of costs. This turns the route with the highest utility into the route with lowest costs. With the restriction to reward functions that only allow negative rewards, the requirement of non-negative costs of Dijkstra's algorithm is met.

A.3 Physical Motion Constraints

The front wheels of a normal car can only be turned up to a certain limit, which results in a minimal turning radius r_{min} for a car (see Figure A.5a). Knowing the minimal turning radius, the limitation of the yaw rate ω_{turn} can be derived with the curvature of the corresponding turning circle

$$\frac{1}{r} = \frac{\dot{\psi}}{\dot{s}} = \frac{\omega}{v} \quad \rightsquigarrow \quad \omega_{\text{turn}} = \frac{v}{r_{\text{min}}} \,. \tag{A.3}$$



(a) Minimal turning radius. (b) Forces.

Figure A.5: The motion of a vehicle is limited through physical constraints.

A second important limitation stems from the dynamics of the car. If a car is accelerated strongly, the driver risks that the car looses traction and becomes harder or even impossible to control. To prevent this kind of behavior, drivers usually stay within the regions were the car can be controlled. This transition is reached when the forces applied to the car exceed the static friction between the wheels and the road surface. The forces stem from steering, which induces a lateral centrifugal force F_{lat}, and from accelerating or decelerating, which induces a longitudinal force F_{lon}. In order to ensure that the vehicle remains controllable, the sum of these perpendicular forces must stay below the static friction force F_S (see Figure A.5b), which can approximately be described by the Coulomb model of friction (Ruina and Pratap, 2008). The force F_S is given by

$$F_S = \mu_S F_N \overset{\text{flat surface}}{\approx} \mu_S F_G = \mu_S m g \tag{A.4}$$

with μ_S being the friction coefficient. F_S depends on the normal force F_N, i.e., the component of the gravity force $F_G = mg$ which is perpendicular to the road surface. The mass of the car is denoted by m and $g \approx 9.81\frac{\mathrm{m}}{\mathrm{s}^2}$ is the gravity constant. Assuming a flat road, it holds that $F_N = F_G$.

The lateral centrifugal force F_{lat} is

$$F_{\text{lat}} = m\frac{v^2}{r} = mv\omega\ . \tag{A.5}$$

The limitation of the yaw rate ω_{traction} is obtained through

$$\begin{aligned}\vec{F}_S &= \vec{F}_{\text{lat}} + \vec{F}_{\text{lon}}\\ \mu_S mg &= \sqrt{(mv\omega_{\text{traction}})^2 + (ma)^2}\\ \omega_{\text{traction}} &= \frac{\sqrt{(\mu_S g)^2 - a^2}}{v}\ .\end{aligned} \tag{A.6}$$

If both constraints are considered, the absolute limitation of the yaw rate for a given velocity and acceleration is

$$\omega_{\max} = \min(\omega_{\text{turn}}, \omega_{\text{traction}})\ . \tag{A.7}$$

Figure A.6 shows the resulting maximal yaw rate over the velocity for several accelerations ($\mu_S = 1$). For small velocities $\omega_{\max}$ is dominated by ω_{turn} while at higher velocities, ω_{traction} is the dominating factor.

The acceleration and deceleration of a car is also restricted through the car physics. These bounds result from the powertrain and braking system of the car and environmental properties such as the road surface type and weather conditions. We subsume these limitations by constants $a_{\max}$ and $a_{\min}$, which express conservative thresholds for acceleration and deceleration for an average car and average conditions.

The constrained action policy $\bar{\pi}^A$ of an action policy π^A is obtained by limiting the controls a and ω according to the listed constraints

$$\bar{\pi}^A = \begin{bmatrix} \bar{a} & = & \max(\min(a, a_{\max}), a_{\min}) \\ \bar{\omega} & = & \max(\min(\omega, \omega_{\max}), -\omega_{\max}) \end{bmatrix}. \tag{A.8}$$

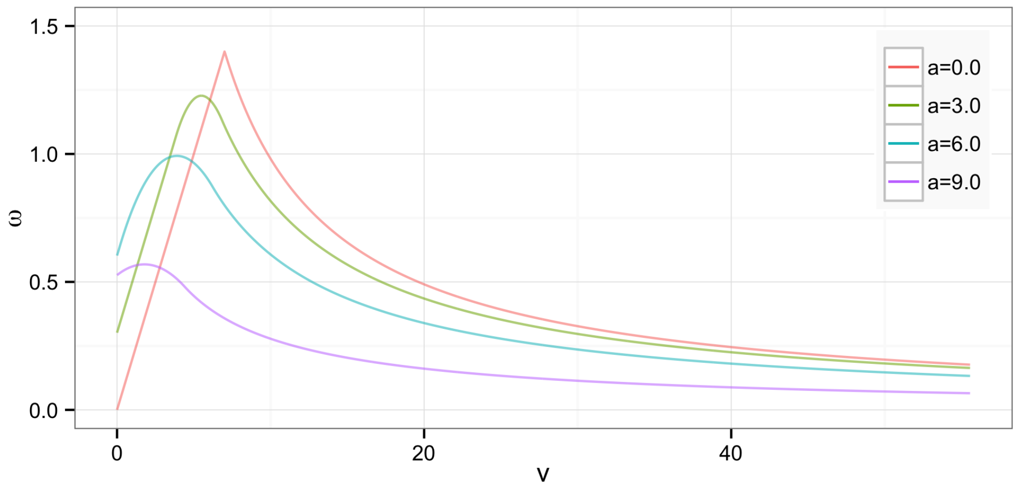


Figure A.6: Plot showing the maximal change in orientation ω in 1 s depending on velocity v and longitudinal acceleration a. The change in orientation is constrained by the minimal turning radius and the traction of the car ($\mu_S = 1$).

The constraints are incorporated in the Bayesian model by using the constrained action policy $\bar{\pi}^A$ derived from π^A with A.8 as input for the motion model. To prevent density estimation over redundant actions that lie outside the valid set, the constrained values are also used for learning the action policy model. This way the learner concentrates on finding a distribution over the valid set of actions.

A.4 Gradient-Based Learning

The task of function estimation in the supervised learning case consists in finding the function that produces the best y given any x. The quality of a function can be assessed by a cost function $C(\theta)$ called the expected cost function (Tsypkin, 1973)

$$C(\theta) = \mathop{\mathbb{E}}_{z} L(z,\theta) = \int L(z,\theta)dP(z)\,. \tag{A.9}$$

The goal is to find the function $F_\theta \in \mathscr{F}$ identified by parameter vector θ that minimizes $C(\theta)$: $\theta^* = \operatorname{argmin}_\theta C(\theta)$. The scalar loss function $L(z,\theta)$ measures the performance of F_θ given sample $z = (x, y)$. Since the ground truth distribution $P(z)$ is unknown, the expected cost function cannot be optimized directly. It is however possible to approximate $P(z)$ through a finite sample set $\mathscr{S}$, thereby obtaining an approximation of the expected costs, namely the empirical cost function

$$C_{\mathscr{S}}(\theta) = \mathop{\mathbb{E}}_{z\in\mathscr{S}} L(z,\theta) = \frac{1}{N}\sum_{i=1}^{N} L(z_i,\theta)\,. \tag{A.10}$$

If the samples $\mathscr{S}$ are drawn independently from $P(z)$, optimizing $C_{\mathscr{S}}$ asymptotically optimizes C. The empirical cost can therefore serve as an estimator for the expected cost, if the training set is large enough (Vapnik, 2006).

For most classes of non-linear models the optimum of $C_{\mathscr{S}}(\theta)$ cannot be analytically estimated, but gradient-based learning can be used to find a local optimum.

A batch gradient descent algorithm iteratively estimates the optimal parameters θ following the update formula

$$\theta_{t+1} = \theta_t - \gamma_t \frac{\partial C_{\mathscr{S}}(\theta)}{\partial\theta} = \theta_t - \gamma_t \frac{1}{N}\sum_{i=1}^{N} \frac{\partial L(z_i,\theta)}{\partial\theta}\,, \tag{A.11}$$

where the learning rate γ_t is a positive number. Many variants of (A.11) and the following on-line version (A.12) have been defined, like replacing γ_t with an appropriately chosen positive definite symmetric matrix for scaling the gradient

(LeCun et al., 1998; Martens and Sutskever, 2012) to make use of second order information.

An on-line gradient descent algorithm performs the updates on individual samples $z_i \sim P(z)$

$$\theta_{t+1} = \theta_t - \gamma_t \frac{\partial L(z_i, \theta)}{\partial \theta} \,. \tag{A.12}$$

The replacement of the summation prevents the need to store samples and to sweep over the entire training set at each cycle. These properties make the learning rule applicable for large or even infinite data sets. Le Cun et al. (Bottou and LeCun, 2004) provide theoretical evidence that a suitably designed on-line learning algorithm asymptotically outperforms any batch learning algorithm in the long run. The line of argument follows the idea that an on-line learning algorithm makes more efficient use of the information provided by the training examples than a batch learning algorithm.

The term that defines the optimization procedure is the gradient of the loss function regardless of using the batch or on-line variant. In order to apply a gradient-based learning method the loss function has to be differentiable.

A broad class of loss functions has the form

$$L_F(z, \theta) = \tilde{L}(F_\theta(x), y) \,. \tag{A.13}$$

The squared loss function $L_{MSE}(z, \theta) = \frac{1}{2}(F_\theta(x) - y)^2$ is a popular instance of this class. Applying the chain rule to L_F yields the gradient

$$\frac{\partial L_F(z, \theta)}{\partial \theta} = \frac{\partial \tilde{L}(F_\theta(x), y)}{\partial F_\theta(x)} \frac{\partial F_\theta(x)}{\partial \theta} \,. \tag{A.14}$$

This class of loss functions is differentiable if both the partial derivatives of $\tilde{L}$ and F_θ exist.

A.5 Partial Derivatives of a Artificial Neural Network

Artificial neural networks are compositions of multivariate functions that constitute the layers of a feed forward network (Rojas, 1996). These functions, also called layer functions, are often chosen to be compositions of linear functions and some sort of sigmoid function, like the logistic function as in the case of a multilayer perceptron. The overall function of an ANN with a linear output layer and n hidden layers is defined recursively as

$$f(x) = A^{(o)} f^{(n)}(x) - b^{(o)} \quad : \quad \mathbb{R}^m \mapsto \mathbb{R} \tag{A.15}$$

$$f^{(i)}(x) = \phi(g^{(i)}(x)) \quad : \quad \mathbb{R}^m \mapsto \mathbb{R}^{m_i} \tag{A.16}$$

$$g^{(i)}(x) = A^{(i)} f^{(i-1)}(x) - b^{(i)} \quad : \quad \mathbb{R}^m \mapsto \mathbb{R}^{m_i} \tag{A.17}$$

$$f^{(0)}(x) = x \quad : \quad x \in \mathbb{R}^m \tag{A.18}$$

with $f^{(i)}$ indicating the subnet function up to the i'th layer and $\phi(x)$ being a vector function that applies a sigmoid function to all inputs. The overall parameters of the ANN constitute of the parameters of all layers $\theta_{ANN} = \left(\theta^{(1)}, \ldots, \theta^{(n)}, \theta^{(o)}\right)$ with each set of parameters consisting of the linear function parameters $\theta^{(i)} \equiv A^{(i)}, b^{(i)}$.

The partial derivatives of the ANN with respect to the network parameters are

$$\frac{\partial f(x)}{\partial \theta_{ANN}} = \begin{pmatrix} \frac{\partial f(x)}{\partial \theta^{(o)}} & \frac{\partial f(x)}{\partial \theta^{(n)}} & \frac{\partial f(x)}{\partial \theta^{(n-1)}} & \dots & \frac{\partial f(x)}{\partial \theta^{(1)}} \end{pmatrix} \tag{A.19}$$

$$\frac{\partial f(x)}{\partial \theta^{(i)}} = \frac{\partial f(x)}{\partial f^{(i)}(x)} \frac{\partial f^{(i)}(x)}{\partial \theta^{(i)}} \tag{A.20}$$

$$\frac{\partial f(x)}{\partial f^{(i)}(x)} = \frac{\partial f(x)}{\partial f^{(n)}(x)} \frac{\partial f^{(n)}(x)}{\partial f^{(n-1)}(x)} \dots \frac{\partial f^{(i+1)}(x)}{\partial f^{(i)}(x)} \tag{A.21}$$

$$\frac{\partial f^{(i+1)}(x)}{\partial f^{(i)}(x)} = \frac{\partial f^{(i+1)}(x)}{\partial g^{(i+1)}(x)} \frac{\partial g^{(i+1)}(x)}{\partial f^{(i)}(x)} \tag{A.22}$$

$$\frac{\partial f^{(i)}(x)}{\partial g^{(i)}(x)} = \frac{\partial \phi(g^{(i)}(x))}{\partial g^{(i)}(x)} \tag{A.23}$$

$$\frac{\partial g^{(i+1)}(x)}{\partial f^{(i)}(x)} = \frac{\partial A^{(i+1)} f^{(i)}(x) - b^{(i+1)}}{\partial f^{(i)}(x)} = A^{(i+1)} \tag{A.24}$$

$$\frac{\partial f^{(i)}(x)}{\partial \theta^{(i)}} = \frac{\partial f^{(i)}(x)}{\partial g^{(i)}(x)} \frac{\partial g^{(i)}(x)}{\partial \theta^{(i)}} \tag{A.25}$$

$$\frac{\partial g^{(i)}(x)}{\partial \theta^{(i)}} = \begin{pmatrix} \frac{\partial g^{(i)}(x)}{\partial A^{(i)}} & \frac{\partial g^{(i)}(x)}{\partial b^{(i)}} \end{pmatrix} = \begin{pmatrix} f^{(i-1)}(x)^T & -1 \\ \vdots & \vdots \\ f^{(i-1)}(x)^T & -1 \end{pmatrix} \tag{A.26}$$

$$\frac{\partial f(x)}{\partial \theta^{(o)}} = \begin{pmatrix} \frac{\partial f(x)}{\partial A^{(o)}} & \frac{\partial f(x)}{\partial b^{(o)}} \end{pmatrix} = \begin{pmatrix} f^{(n)}(x)^T & -1 \\ \vdots & \vdots \\ f^{(n)}(x)^T & -1 \end{pmatrix} . \tag{A.27}$$

The derivatives can be calculated efficiently with back-propagation in order to optimize a function with gradient-based learning (LeCun et al., 1998).

Acronyms

ACC adaptive cruise control.

ADAS advanced driver assistance systems.

AHMM abstract hidden Markov model.

ANN artificial neural network.

BOF Bayesian occupancy filter.

DARPA Defense Advanced Research Projects Agency.

DBN dynamic Bayesian network.

E-step expectation step.

EM expectation maximization.

FDT fuzzy decision tree.

GDT generalized decision tree.

GEM generalized expectation maximization.

GPS Global Positioning System.

HME hierarchical mixture of experts.

HMM hidden Markov model.

IS-MCEM importance sampling Monte Carlo expectation maximization.

LW likelihood weighting.

M-step	maximization step.
MAP	maximum a posteriori.
MCEM	Monte Carlo expectation maximization.
MCMC	Markov chain Monte Carlo.
MDP	Markov decision process.
ML	maximum likelihood.
MLL	mean log likelihood.
OSM	OpenStreetMap.
POMDP	partially observable Markov decision process.
RMSE	root mean squared error.
SLAM	simultaneous localization and mapping.
SMC	sequential Monte Carlo.
SVM	support vector machine.

Glossary

action policy Π^A, π^A Mapping from situations to actions.

episode A sequence of situations describing the ongoing change in the world from the perspective of an observer.

goal policy Π^G, π^G Mapping from situations to goals.

goal policy termination state T^G, t^G indicates termination of current goal policy.

goal regions $\mathscr{G}$ The set of all goal regions derived from the road network.

lane segments $\mathscr{L}$ The set of all lane segments in the road network. Lanes are divided into several segments since they are approximated by rectangular shapes.

lane segments of goal area $\mathscr{L}_{\pi^G}$ The set of all lane segments that are part of the route π^G.

lane segments of route $\mathscr{L}_{\pi^R}$ The set of all lane segments that are part of the route π^R.

measurements Z, z Measurements of traffic participants.

route policy Π^R, π^R Mapping from situations to routes.

route policy termination state T^R, t^R indicates termination of current route policy.

routes $\mathscr{R}$ The set of all routes consists of all possible ways from each lane segment of the road network to each lane segment of each goal region.

routes to goal $\mathscr{R}_{c^L,\pi^G}$ The set of all routes that begin at lane segment c^L and can reach the goal area π^G.

situation A situation comprises the state of the world at a point in time. We restrict the state to aspects that are normally considered by a traffic participant for navigating in traffic.

situational context C, c describes the situation from the perspective of a traffic participant through relations.

traffic participants' states X, x A state x of a single traffic participant is described by his position $\left(x_1, x_2\right)$, heading ψ (orientation) and velocity v in a global world frame.

List of Figures

List of Tables

Bibliography

Juan David Adarve, Mathias Perrollaz, Alexandros Makris, and Christian Laugier. Computing occupancy grids from multiple sensors using linear opinion pools. In Proc. of *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4074–4079. IEEE, 2012.

G. Agamennoni, J. I. Nieto, and E. M. Nebot. Estimation of multivehicle dynamics by considering contextual information. *IEEE Transactions on Robotics*, 28 (4):855 – 870, 2012.

Andreas Alin, Martin V. Butz, and Jannik Fritsch. Incorporating environmental knowledge into Bayesian filtering using attractor functions. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 476–481. IEEE, 2012.

Andreas Alin, Jannik Fritsch, and Martin V. Butz. Improved tracking and behavior anticipation by combining street map information with Bayesian-filtering. In Proc. of *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2235–2242. IEEE, 2013.

E.L. Allgower and K. Georg. *Introduction to numerical continuation methods*, volume 45. Society for Industrial Mathematics, 2003.

Georges S. Aoude, Vishnu R. Desaraju, Lauren H. Stephens, and Jonathan P. How. Behavior classification algorithms at intersections and validation using naturalistic data. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 601–606, 2011.

Itamar Arel, Derek C. Rose, and Thomas P. Karnowski. Deep machine learning–a new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, 5(4):13–18, 2010.

AutoNOMOS Labs. Autonomous car navigates the streets of berlin. Press Release, 2011. URL `http://www.fu-berlin.de/en/presse/informationen/fup/2011/fup_11_291`. [accessed 2015-07-15].

Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, Stephen Cacciola, Patrick Currier, Aaron Dalton, Jesse Farmer, Jesse Hurdus, Shawn Kimmel, Peter King, Andrew Taylor, David Van Covern, and Mike Webster. Odin: Team VictorTango's entry in the DARPA Urban Challenge. *Journal of Field Robotics*, 25(8):467–492, 2008.

Haoyu Bai, David Hsu, Wee Sun Lee, and Vien A. Ngo. Monte Carlo value iteration for continuous-state POMDPs. In *Algorithmic foundations of robotics IX*, pages 175–191. Springer, 2011.

Haoyu Bai, David Hsu, and Wee Sun Lee. Integrated perception and planning in the continuous space: A POMDP approach. In Proc. of *Robotics: Science and Systems Conference (RSS)*, 2013.

Y. Bar-Shalom. Multitarget-multisensor tracking: Applications and advances. Volume III. *Norwood, MA: Artech House*, 2000.

Parag H. Batavia. *Driver-adaptive lane departure warning systems*. PhD thesis, Carnegie Mellon University, 1999.

Thomas Batz, Kym Watson, and Jürgen Beyerer. Recognition of dangerous situations within a cooperative group of vehicles. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 907–912, 2009.

Mathias Bauer. Integrating probabilistic reasoning into plan recognition. In Proc. of *European Conference on Artificial Intelligence (ECAI)*, pages 620–620, 1994.

Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6, 1957.

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Ake Björck. *Numerical methods for least squares problems.* Siam, 1996.

S. Blackman and R. Popoli. *Design and analysis of modern tracking systems.* Norwood, MA: Artech House, 1999.

Jonathan Bohren, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart, Paul Vernaza, Jason Derenick, John Spletzer, and Brian Satterfield. Little Ben: The Ben Franklin Racing team's entry in the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):598–614, 2008.

Sarah Bonnin, Franz Kummert, and Jens Schmüdderich. A generic concept of a system for predicting driving behaviors. In Proc. of *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1803–1808, 2012.

James G. Booth and James P. Hobert. Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(1):265–285, 1999.

L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17:9, 1998.

Leon Bottou and Yann LeCun. Large scale online learning. *Advances in Neural Information Processing Systems (NIPS)*, 16:217, 2004.

Sebastian Brechtel, Tobias Gindele, Jan Vogelgesang, and Rüdiger Dillmann. Probabilistisches Belegtheitsfilter zur Schätzung dynamischer Umgebungen unter Verwendung multipler Bewegungsmodelle. In Proc. of *21. Fachgespräch Autonome Mobile Systeme*, pages 49–56, Karlsruhe, 2009.

Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Recursive importance sampling for efficient grid-based occupancy filtering in dynamic environments. In Proc. of *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3932–3938, Anchorage, AL, USA, 2010.

Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Probabilistic MDP-behavior planning for cars. In Proc. of *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1537–1542, Washington DC, USA, 2011.

Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Solving continuous POMDPs: Value iteration with incremental learning of an efficient space representation. In Proc. of *International Conference on Machine Learning (ICML)*, volume 28, pages 370–378, Atlanta, GA, USA, 2013.

Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Probabilistic Decision-making under Uncertainty for Autonomous Driving Using Continuous POMDPs. In Proc. of *IEEE International Conference on Intelligent Transportation Systems*, pages 392–399, Qingdao, China, 2014.

L. Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

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

C.E. Brodley and P.E. Utgoff. Multivariate decision trees. *Machine Learning*, 19 (1):45–77, 1995.

Martin Buehler, Karl Iagnemma, and Sanjiv Singh. The 2005 DARPA Grand Challenge. *Springer Tracts in Advanced Robotics*, 36(5):1–43, 2007.

Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA Urban Challenge: autonomous vehicles in city traffic*, volume 56. Springer, 2009.

Hung Hai Bui, Svetha Venkatesh, and Geoff West. Policy recognition in the abstract hidden Markov model. *Journal of Artificial Intelligence Research*, 17: 451–499, 2002.

Eduardo F. Camacho and Carlos Bordons Alba. *Model predictive control*. Springer, 2013.

Guy Campion and Woojin Chung. Wheeled robots. *Handbook of Robotics*, pages 391–410, 2008.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In Proc. of *International Conference on Machine Learning (ICML)*, pages 161–168. ACM, 2006.

Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In Proc. of *International Conference on Machine Learning (ICML)*, pages 96–103. ACM, 2008.

Gilles Celeux and Jean Diebolt. The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational Statistics Quarterly*, 2(1):73–82, 1985.

Eugene Charniak and Robert P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, 1993.

Guang Chen, Feihu Zhang, Daniel Clarke, and Alois Knoll. Learning to track multi-target online by boosting and scene layout. In Proc. of *International Conference on Machine Learning and Applications (ICMLA)*, volume 1, pages 197–202. IEEE, 2013.

Wongun Choi and Silvio Savarese. A unified framework for multi-target tracking and collective activity recognition. In *Computer Vision–ECCV 2012*, pages 215–230. Springer, 2012.

City of Karlsruhe. Imagery of Karlsruhe. ©Stadt Karlsruhe, 2010.

C. Coue, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessiere. Bayesian occupancy filtering for multitarget tracking: An automotive application. *The International Journal of Robotics Research*, 25(1):19, 2006.

I.J. Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

I. Dagli and D. Reichardt. Motivation-based approach to behavior prediction. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, volume 38, 2002.

I. Dagli, M. Brost, and G. Breuel. Action recognition and prediction for driver assistance systems using dynamic belief networks. *Lecture Notes in Computer Science*, pages 179–194, 2003.

Darpa. Urban Challenge - route network definition file (RNDF) and mission data file (MDF) formats. Technical report, Defense Advanced Research Projects Agency, 2007.

F. Daum. Multitarget-multisensor tracking: Principles and techniques. *IEEE Aerospace and Electronic Systems Magazine*, 11(2):41, 1996.

Bernard Delyon, Marc Lavielle, and Eric Moulines. Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics*, pages 94–128, 1999.

A. Demcenko, M. Tamosiunaite, A. Vidugiriene, and A. Saudargiene. Vehicle's steering signal predictions using neural networks. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 1181–1186. IEEE, 2008.

Andriejus Demčenko, Minija Tamošiunaite, Aušra Vidugiriene, and Leonas Jakevičius. Lane marker parameters for vehicle's steering signal prediction. *WSEAS Transactions on Systems*, 8(2):251–260, 2009.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

Ernst D. Dickmanns and A. Zapp. Autonomous high speed road vehicle guidance by computer vision. In Proc. of *World Congress of the International Federation of Automatic Control*, volume 1, pages 221—226, 1988.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

Anup Doshi and Mohan M. Trivedi. Tactical driver behavior prediction and intent inference: A review. In Proc. of *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1892–1897. IEEE, 2011.

Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

Arnaud Doucet, Nando De Freitas, Neil Gordon, et al. *Sequential Monte Carlo methods in practice*, volume 1. Springer New York, 2001.

Andreas Eidehall and Lars Petersson. Statistical threat assessment for general road scenes using Monte Carlo sampling. *Intelligent Transportation Systems, IEEE Transactions on*, 9(1):137–147, 2008.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

Floriana Esposito, Donato Malerba, Giovanni Semeraro, and J Kay. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.

Daniel Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Eno Center for Transportation. Washington, DC*, 2013. URL `https://www.enotrans.org/publications`. [accessed 2015-07-15].

Andreas Festag, Roberto Baldessari, Wenhui Zhang, Long Le, Amardeo Sarma, and Masatoshi Fukukawa. Car-2-x communication for safety and infotainment in europe. *NEC Technical Journal*, 3(1):21–26, 2008.

J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The batmobile: Towards a Bayesian automated taxi. In Proc. of *International Joint Conference on Artificial Intelligence*, volume 14, pages 1878–1885, 1995.

D. Fox. KLD-sampling: Adaptive particle filters and mobile robot localization. *Advances in Neural Information Processing Systems (NIPS)*, 2001.

Robert Fung and Kuo-Chu Chang. Weighting and integrating evidence for stochastic simulation in Bayesian networks. In Proc. of *Uncertainty in artificial intelligence*, volume 5, pages 209–219, 1989.

FZI Research Center for Information Technology at Karlsruhe Institute of Technology. Fully automatic – following the tracks of Bertha Benz. Press Release, 2013. URL `http://www.kit.edu/visit/pi_2013_13901.php`. [accessed 2015-07-15].

J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining high-speed data streams. In Proc. of *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 523–528. ACM, 2003.

Hervé Gauvrit, J.P. Le Cadre, and Claude Jauffret. A formulation of multitarget tracking as an incomplete data problem. *IEEE Transactions on Aerospace and Electronic Systems*, 33(4):1242–1257, 1997.

Walter R. Gilks, Sylvia Richardson, and David J. Spiegelhalter. *Markov chain Monte Carlo in practice*, volume 2. CRC press, 1996.

Tobias Gindele, Daniel Jagszent, Benjamin Pitzer, and Rüdiger Dillmann. Design of the planner of team AnnieWAY's autonomous vehicle used in the DARPA Urban Challenge 2007. In Proc. of *IEEE Intelligent Vehicles Symposium*, pages 1131–1136, Eindhoven, Niederlande, 2008.

Tobias Gindele, Sebastian Brechtel, Joachim Schröder, and Rüdiger Dillmann. Bayesian occupancy grid filter for dynamic environments using prior map knowledge. In Proc. of *IEEE Intelligent Vehicles Symposium*, pages 669–676, Xi'an, China, 2009.

Tobias Gindele, Sebastian Brechtel, and Rüdiger Dillmann. A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments. In Proc. of *IEEE International Conference on Intelligent Transportation Systems*, pages 1625–1631, Madeira, Portugal, 2010.

Tobias Gindele, Sebastian Brechtel, and Rüdiger Dillmann. Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intelligent Transportation Systems Magazine*, 7(1):69–79, 2015.

Robert P. Goldman, Christopher W. Geib, and Christopher A. Miller. A new model of plan recognition. In Proc. of *Conference on Uncertainty in Artificial Intelligence*, pages 245–254, 1999.

Erico Guizzo. How Google's self-driving car works. *IEEE Spectrum*, Oct. 2013. URL `http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works#`. [accessed 2015-07-15].

H. Guo and W. Hsu. A survey of algorithms for real-time Bayesian network inference. In Proc. of *joint AAAI-02/KDD-02/UAI-02 workshop on Real-Time Decision Support and Diagnosis Systems*, Edmonton, Alberta, Canada, 2002.

Dirk Hähnel, Wolfram Burgard, and Sebastian Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27, 2003.

Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations*. Springer, 1993.

Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

John C. Hayward. Near-miss determination through use of a scale of danger. *Highway Research Record*, 1972.

Christoph Hermes, C Wohler, Konrad Schenk, and Franz Kummert. Long-term vehicle motion prediction. In Proc. of *Intelligent Vehicles Symposium*, pages 652–657. IEEE, 2009.

T. Hester and P. Stone. Negative information and line observations for Monte Carlo localization. In Proc. of *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2764–2769. IEEE, 2008.

J. Hoffman, M. Spranger, D. Gohring, and M. Jungel. Making use of what you don't see: negative information in Markov localization. In Proc. of *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2947–2952. IEEE, 2005.

Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(3):334–352, 2004.

Marcus J. Huber, Edmund H. Durfee, and Michael P. Wellman. The automated mapping of plans for plan recognition. In Proc. of *International Conference on Uncertainty in Artificial Intelligence*, pages 344–351, 1994.

T. Hulnhagen, Ingo Dengler, Andreas Tamke, Thao Dang, and Gabi Breuel. Maneuver recognition using probabilistic finite-state machines and fuzzy logic. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 65–70. IEEE, 2010.

L. Hyafil and R.L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

Joseph G. Ibrahim, Ming-Hui Chen, and Stuart R. Lipsitz. Monte Carlo EM for missing covariates in parametric regression models. *Biometrics*, 55(2):591–596, 2004.

Elena Ikonomovska. *Algorithms for Learning Regression Trees and Ensembles on Evolving Data Streams*. PhD thesis, Jozef Stefan International Postgraduate School, Ljubljana, Slovenia, October 2012.

M.I. Jordan. A statistical approach to decision tree modeling. In Proc. of *Conference on Computational Learning Theory*, pages 13–20. ACM, 1994.

M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2):181–214, 1994.

Bridle JS. *Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition*, pages 227–236. Springer, Berlin, Germany, 1989.

E. Käfer, Christoph Hermes, C. Wöhler, Helge Ritter, and Franz Kummert. Recognition of situation classes at road intersections. In Proc. of *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3960–3965. IEEE, 2010.

Sören Kammel, Julius Ziegler, Benjamin Pitzer, Moritz Werling, Tobias Gindele, Daniel Jagzent, Joachim Schröder, Michael Thuy, Matthias Goebl, Felix von Hundelshausen, Oliver Pink, Christian Frese, and Christoph Stiller. Team AnnieWAY's autonomous system for the DARPA Urban Challenge 2007. *International Journal of Field Robotics Research*, 25(9):615–639, 2008.

Dietmar Kasper, Galia Weidl, Thao Dang, Gabi Breuel, Andreas Tamke, and Wolfgang Rosenstiel. Object-oriented Bayesian networks for detection of lane change maneuvers. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 673–678. IEEE, 2011.

Henry Kautz and James F. Allen. Generalized plan recognition. In Proc. of *National Conference on Artificial Intelligence*, volume 1, pages 32–37. Philadelphia, PA, 1986.

Will Knight. Driverless cars are further away than you think. *MIT Technology Review Magazine*, 116(6), Nov./Dec. 2013. URL `http://www.technologyreview.com/featuredstory/520431/driverless-cars-are-further-away-than-you-think/`. [accessed 2015-07-15].

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques.* The MIT Press, 2009.

Tomasz Kucner, Jari Saarinen, Martin Magnusson, and Achim J. Lilienthal. Conditional transition maps: Learning motion patterns in dynamic environments. In Proc. of *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

Puneet Kumar, Mathias Perrollaz, Stéphanie Lefèvre, and Christian Laugier. Learning-based approach for online lane change intention prediction. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, 2013.

H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In Proc. of *Robotics: Science and Systems (RSS)*, 2008.

Gerhard Kurz, Igor Gilitschenski, Simon Julier, and Uwe D. Hanebeck. Recursive estimation of orientation based on the Bingham distribution. In Proc. of *International Conference on Information Fusion*, pages 1487–1494. IEEE, 2013.

Christian Laugier, Igor E. Paromtchik, Mathias Perrollaz, Mao Yong, John-David Yoder, Christopher Tay, Kamel Mekhnacha, and Amaury Nègre. Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety. *IEEE Intelligent Transportation Systems Magazine*, 3(4):4–19, 2011.

Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. *Neural networks: Tricks of the trade*, pages 546–546, 1998.

S. Lefèvre, J. Ibanez-Guzman, and C. Laugier. Context-based estimation of driver intent at road intersections. In Proc. of *IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems*, 2011.

Richard A. Levine and George Casella. Implementations of the Monte Carlo EM algorithm. *Journal of Computational and Graphical Statistics*, 10(3):422–439, 2001.

Martin Liebner, Christian Ruhhammer, Felix Klanner, and Christoph Stiller. Generic driver intent inference based on parametric models. In Proc. of *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 268–275, 2013.

Roderick J.A. Little and Donald B. Rubin. *Statistical analysis with missing data*. Probability and Statistics. Wiley New York, 2002.

D.J.C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

James Martens. Deep learning via hessian-free optimization. In Proc. of *International Conference on Machine Learning (ICML)*, pages 735–742, 2010.

James Martens and Ilya Sutskever. Training deep and recurrent networks with hessian-free optimization. In *Neural Networks: Tricks of the Trade*, pages 479–535. Springer, 2012.

Frank P. McKenna. The human factor in driving accidents. An overview of approaches and problems. *Ergonomics*, pages 867–877, 2010.

Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

Kamel Mekhnacha, Yong Mao, David Raulo, and Christian Laugier. The "fast clustering-tracking" algorithm in the Bayesian occupancy filter framework. In Proc. of *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 238–245, Aug. 2008.

Xiao-Li Meng and David Van Dyk. The EM algorithm—an old folk-song sung to a fast new tune. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(3):511–567, 1997.

D. Meyer-Delius, C. Plagemann, G. Von Wichert, W. Feiten, G. Lawitzky, and W. Burgard. A probabilistic relational model for characterizing situations in dynamic multi-agent systems. In Proc. of *Annual Conference of the Gesellschaft für Klassifikation: Data Analysis, Machine Learning and Applications*, pages 269–276. Springer, 2008.

D. Meyer-Delius, C. Plagemann, and W. Burgard. Probabilistic situation recognition for vehicular traffic scenarios. In Proc. of *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4161–4166, 2009.

Isaac Miller, Mark Campbell, Dan Huttenlocher, Frank-Robert Kline, Aaron Nathan, Sergei Lupashin, Jason Catlin, Brian Schimpf, Pete Moran, Noah Zych, Ephrahim Garcia, Mike Kurdziel, and Hikaru Fujishima. Team Cornell's Skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics*, 25(8):493–527, 2008.

John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243, 1989.

Manfred Mitschke and Henning Wallentowitz. *Dynamik der Kraftfahrzeuge*, volume 4. Springer, 1972.

Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

J. Moody. Fast learning in multi-resolution hierarchies. In Proc. of *Advances in Neural Information Processing Systems (NIPS)*, pages 29–39, 1989.

Julien Moras, Véronique Cherfaoui, and Philippe Bonnifait. Credibilist occupancy grids for vehicle perception in dynamic environments. In Proc. of *IEEE International Conference on Robotics and Automation (ICRA)*, pages 84–89. IEEE, 2011.

J.J. Moré and Z. Wu. Global continuation for distance geometry problems. *SIAM Journal on Optimization*, 7(3):814–836, 1997.

Brendan Morris, Anup Doshi, and Mohan Trivedi. Lane change intent prediction for driver assistance: On-road design and evaluation. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 895–901. IEEE, 2011.

K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, 2002.

K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

S.K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

S.K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

Radford M. Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

Pascal Neis, Dennis Zielstra, and Alexander Zipf. The street network evolution of crowdsourced maps: OpenStreetMap in Germany 2007–2011. *Future Internet*, 4(1):1–21, 2011.

Dennis Nienhüser. *Kontextsensitive Erkennung und Interpretation fahrrelevanter statischer Verkehrselemente*. PhD thesis, Karlsruhe Institue of Technology, 2014. URL `http://nbn-resolving.org/urn:nbn:de:swb:90-379135`.

T. Nothdurft, P. Hecker, S. Ohl, F. Saust, M. Maurer, A. Reschka, and J.R. Bohmer. Stadtpilot: First fully autonomous test drives in urban traffic. In Proc. of *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 919–924. IEEE, 2011.

C. Olaru and L. Wehenkel. A complete fuzzy decision tree technique. *Fuzzy sets and systems*, 138(2):221–254, 2003.

Nuria Oliver, Eric Horvitz, and Ashutosh Garg. Layered representations for human activity recognition. In Proc. of *IEEE International Conference on Multimodal Interfaces*, pages 3–8. IEEE, 2002.

Openstreetmap, 2013. URL `http://www.openstreetmap.org`. [accessed 2013-10-14].

Michaël Garcia Ortiz, Jannik Fritsch, Franz Kummert, and Alexander Gepperth. Behavior prediction at multiple time-scales in inner-city scenarios. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 1068–1073. IEEE, 2011a.

Michaël Garcia Ortiz, Jens Schmüdderich, Franz Kummert, and Alexander Gepperth. Situation-specific learning for ego-vehicle behavior prediction systems. In Proc. of *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1237–1242. IEEE, 2011b.

Sarah Osentoski, Victoria Manfredi, and Sridhar Mahadevan. Learning hierarchical models of activity. In Proc. of *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 891–896. IEEE, 2004.

Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. *Advances in Neural Information Processing Systems (NIPS)*, pages 1043–1049, 1998.

Donald Patterson, Lin Liao, Dieter Fox, and Henry Kautz. Inferring high-level behavior from low-level sensors. In Proc. of *Ubiquitous Computing (UbiComp)*, pages 73–89. Springer, 2003.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In Proc. of *IEEE International Conference on Computer Vision*, pages 261–268. IEEE, 2009.

Dominik Petrich, Thao Dang, Dietmar Kasper, Gabi Breuel, and Christoph Stiller. Map-based long term motion prediction for vehicles in traffic environments. In Proc. of *IEEE International Conference on Intelligent Transportation Systems*, pages 2166–2172, 2013.

Eleni Petridou and Maria Moustaki. Human factors in the causation of road traffic crashes. *European Journal of Epidemiology*, 16(9):819–826, 2000.

Josep M. Porta, Nikos Vlassis, Matthijs T.J. Spaan, and Pascal Poupart. Point-based value iteration for continuous POMDPs. *The Journal of Machine Learning Research*, 7:2329–2367, 2006.

D. Potts. Incremental learning of linear model trees. In Proc. of *International Conference on Machine Learning (ICML)*, page 84. ACM, 2004.

Pascal Poupart, Kee-Eung Kim, and Dongho Kim. Closing the gap: Improved bounds on optimal POMDP solutions. In Proc. of *International Conference on Automated Planning and Scheduling (ICAPS)*, 2011.

Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*, volume 414. John Wiley & Sons, 2009.

David V. Pynadath and Michael P. Wellman. Accounting for context in plan recognition, with application to traffic monitoring. In Proc. of *Conference on Uncertainty in Artificial Intelligence*, pages 472–481, 1995.

David V. Pynadath and Michael P. Wellman. Probabilistic state-dependent grammars for plan recognition. In Proc. of *Conference on Uncertainty in Artificial Intelligence*, pages 507–514, 2000.

J.R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

J.R. Quinlan. Learning with continuous classes. In Proc. of *Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore, 1992.

Fred W. Rauskolb, Kai Berger, Christian Lipski, Marcus Magnor, Karsten Cornelsen, Jan Effertz, Thomas Form, Fabian Graefe, Sebastian Ohl, Walter Schumacher, Jörn-Marten Wille, Peter Hecker, Tobias Nothdurft, Michael Doering, Kai Homeier, Johannes Morgenroth, Lars Wolf, Christian Basarke, Christian Berger, Tim Gülke, Felix Klose, and Bernhard Rumpe. Caroline: An autonomously driving vehicle for urban environments. *Journal of Field Robotics*, 25(9):674–724, 2008.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods.* Springer Texts in Statistics. Springer, New York, 2 edition, 2004.

Raúl Rojas. *Neutral Networks: A Systematic Introduction.* Springer, 1996.

J.K. Rosenblatt. DAMN: A distributed architecture for mobile navigation. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2):339–360, 1997.

Andy L. Ruina and Rudra Pratap. *Introduction to Statics and Dynamics.* Preprint for Oxford University Press, 2008.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

S.J. Russell, P. Norvig, J.F. Canny, J. Malik, and D.D. Edwards. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Englewood Cliffs, NJ, 1995.

Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial intelligence*, 5(2):115–135, 1974.

S.R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):660–674, 1991.

Joachim Schröder. *Adaptive Verhaltensentscheidung und Bahnplanung für kognitive Automobile*. PhD thesis, Universität Karlsruhe (TH), 2009.

I.K. Sethi. Entropy nets: From decision trees to neural networks. *Proceedings of the IEEE*, 78(10):1605–1613, 1990.

S. Shah and P.S. Sastry. New algorithms for learning and pruning oblique decision trees. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 29(4):494–505, 1999.

Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

David Shepardson. GM says self-driving vehicles not ready in 'foreseeable future'; Nissan vows by 2020. Detroit News, 2013. URL `http://www.detroitnews.com/article/20131119/AUTO01/311190072#ixzz2lC5duWxC`. [accessed 2013-11-22].

Matthijs T.J. Spaan and Nikos A. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 24:195–220, 2005.

J.E. Stromberg, J. Zrida, and A. Isaksson. Neural trees-using neural nets in a tree classifier structure. In Proc. of *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 137–140. IEEE, 1991.

Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P Goldman. *Plan, Activity, and Intent Recognition: Theory and Practice*. Newnes, 2014.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.

MengKeat Christopher Tay, Kamel Mekhnacha, Cheng Chen, Manuel Yguel, and Christian Laugier. An efficient formulation of the Bayesian occupation filter for target tracking in dynamic environments. *International Journal of Vehicle Autonomous Systems*, 6(1):155–171, 2008.

S. Thrun. A probabilistic on-line mapping algorithm for teams of mobile robots. *The International Journal of Robotics Research*, 20(5):335–363, 2001.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. MIT press, Cambridge, Massachusetts, USA, 2005.

Sebastian Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.

Quan Tran and Jonas Firl. A probabilistic discriminative approach for situation recognition in traffic scenarios. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 147–152. IEEE, 2012.

Manolis Tsogas, Xun Dai, George Thomaidis, Panagiotis Lytrivis, and Angelos Amditis. Detection of maneuvers using evidence theory. In Proc. of *IEEE Intelligent Vehicles Symposium*, pages 126–131. IEEE, 2008.

Ya Tsypkin. *Foundations of the Theory of Learning Systems*, volume 101. Academic Press, New York, 1973.

Simon Ulbrich and Markus Maurer. Probabilistic online POMDP decision making for lane changes in fully automated driving. In Proc. of *IEEE International Conference on Intelligent Transportation Systems*, pages 2063–2070, 2013.

Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William ldquoRedrdquo Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. Autonomous driving in urban environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

Jur van den Berg, Sachin Patil, and Ron Alterovitz. Efficient approximate value iteration for continuous Gaussian POMDPs. In Proc. of *Conference on Artificial Intelligence (AAAI)*, 2012.

V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 2000.

V.N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, New York, 2006.

Andrei Vatavu, Radu Danescu, and Sergiu Nedevschi. Environment perception using dynamic polylines and particle based occupancy grids. In Proc. of *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 239–244. IEEE, 2011.

Viz-Lab. PROUD – Car Test 2013. Webpage, 2013. URL `http://vislab.it/proud/`. [accessed 2015-07-15].

Andreas von Eichhorn, Moritz Werling, Peter Zahn, and Dieter Schramm. Maneuver prediction at intersections using cost-to-go gradients. In Proc. of *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 112–117, 2013.

Greg C.G. Wei and Martin A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990.

P.M. Williams. Using neural networks to model conditional multivariate densities. *Neural Computation*, 8(4):843–854, 1996.

Hermann Winner, Bernd Danner, and Joachim Steinle. Adaptive cruise control. In *Handbuch Fahrerassistenzsysteme*, pages 478–521. Springer, 2009.

C.F. Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

W. Yao, H. Zhao, F. Davoine, and H. Zha. Learning lane change trajectories from on-road driving data. In Proc. of *IEEE Intelligent Vehicles Symposium (IV)*, pages 885–890. IEEE, 2012.

Jingang Yi, Luis Alvarez, and Roberto Horowitz. Adaptive emergency braking control with underestimation of friction coefficient. *IEEE Transactions on Control Systems Technology*, 10(3):381–392, 2002.

C. Yuan and M.J. Druzdzel. Importance sampling algorithms for Bayesian networks: Principles and performance. *Mathematical and Computer Modelling*, 43(9-10):1189–1207, 2006.

F. Benjamin Zhan and Charles E. Noon. Shortest path algorithms: an evaluation using real road networks. *Transportation Science*, 32(1):65–73, 1998.

Jianwei Zhang and Bernd Rössler. Situation analysis and adaptive risk assessment for intersection safety systems in advanced assisted driving. In Proc. of *Fachgespräch Autonome Mobile Systeme*, pages 249–258, 2009.

Enlu Zhou, Michael C. Fu, and Steven I. Marcus. Solving continuous-state POMDPs via density projection. *IEEE Transactions on Automatic Control*, 55 (5):1101–1116, 2010.