Signal phase and timing prediction for intelligent transportation systems

Valeriy Khakhutskyy
Signal phase and timing prediction for intelligent transportation systems

Vorhergabe von Signalphasen und Signalzeiten für intelligente Verkehrssysteme

Author: Valeriy Khakhutskyy
Supervisor: Prof. Dr. Javier Esparza
Advisors: Prof. Dr. Thomas A. Runkler, Dr. Justinian Rosca
Date: November 15, 2011
I assure the single handed composition of this master’s thesis only supported by declared resources.

Acknowledgments

This study was carried out at Siemens Corporate Research, Princeton, NJ, in the department of Intelligent Systems & Control.

It is a pleasure to thank those who made this thesis possible. I would like to express my gratitude to my advisers, Dr. Thomas Runkler and Dr. Justinian Rosca, for the time and knowledge they shared with me. I am grateful to my supervisor, Dr. Javier Esparza, for his valuable feedback. Finally, I am indebted to my family and my friends for their spiritual support. However, I am the only one to blame for errors in this thesis.
Abstract

This work studies the methods for signal phase and timing prediction and their application in Intelligent Transportation Systems (ITS). For a given movement direction, we want to be able to predict the traffic light signal at every moment of time for the next few seconds to several minutes. The ITS applications can advise a driver about the optimal speed and future traffic light changes making the driving safer as well as more fuel-efficient and environmentally friendly.

The problem of traffic signal prediction is not trivial since traffic light controllers are programmed to react on the immediate traffic load using complex algorithms. We approach the problem by decomposing it into a number of simpler tasks and applying machine learning techniques for prediction.

The evaluation of the developed methods using a real-world dataset shows that they satisfy the requirements of many ITS applications in terms of accuracy and prediction horizon. They also can adapt to the new traffic conditions and can be extended with new sensor technologies.
Zusammenfassung

Diese Arbeit untersucht die Methoden zur Vorhersage von Signalphasen und Signalzeiten einer Lichtsignalanlage (LSA) für die Anwendung in intelligenten Verkehrssystemen (IVS). Für eine gegebene Bewegungsrichtung wollen wir in der Lage sein das Ampelsignal zu jedem Zeitpunkt für die nächsten Sekunden der Minuten vorherzusagen. Die IVS-Anwendungen können den Fahrer dann über die optimale Fahrgeschwindigkeit informieren. Somit kann ein Fahrer das Abbremsen und Wiederbeschleunigen auf Kreuzungen vermeiden, was zu einer sparsamen und umweltfreundlichen Fahrweise führt.

Das Problem der Signalvorhersage ist nicht trivial, weil die LSA darauf programmiert sind auf die unmittelbare Verkehrslage mit Hilfe von komplexen Steuerungsalgorithmen zu reagieren. Wir zerlegen das Problem in eine Reihe von einfachen Aufgaben und lösen diese mit Techniken des maschinellen Lernens.

Die Evaluierung der entwickelten Methoden unter Verwendung eines realen Datensatzes zeigt, dass diese den Anforderungen vieler IVS-Anwendungen im Bezug auf Genauigkeit und Prognosehorizont genügen. Weiterhin können die entwickelten Methoden sich an die neuen Verkehrssituationen und Bedingungen anpassen, sowie durch neue Sensortechnologien erweitert werden.
# Contents

- Acknowledgements vii
- Abstract ix
- List of Figures xv
- List of Tables xvi

1. **Introduction** 1
   - 1.1. Motivation 1
   - 1.2. Applications 3
   - 1.3. Challenges 7
   - 1.4. Scope and limits of the work 8
   - 1.5. Structure overview 8

2. **Background Information** 11
   - 2.1. Introduction of the terminology 11
   - 2.2. Fundamental principles and architecture of intersections 13
   - 2.3. Controller modes 16

3. **Related Work** 21

4. **Dataset Description** 23

   - 5.1. Pre-processing and feature extraction 27
   - 5.2. Motivation for the choice of machine learning methods 31
   - 5.3. k-nearest neighbors 31
   - 5.4. Prediction with feed-forward neural networks 32
   - 5.5. Decision trees 37
   - 5.6. Ensemble methods 40

6. **Phase Length Prediction** 41
   - 6.1. Model 41
   - 6.2. Evaluation 42

7. **Phase Activity Prediction** 45
   - 7.1. Model 46
   - 7.2. Evaluation 47

8. **Next Phase Prediction** 51
   - 8.1. Model 51
   - 8.2. Evaluation 52

9. **Traffic Flow Prediction** 55
   - 9.1. Models 55
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2. Evaluation</td>
<td>59</td>
</tr>
<tr>
<td><strong>10. Components Integration and Interaction</strong></td>
<td>61</td>
</tr>
<tr>
<td>10.1. Integration in Green Light Optimized Speed Advisory application</td>
<td>61</td>
</tr>
<tr>
<td>10.2. Integration in Realistic Traffic Control in Simulations application</td>
<td>63</td>
</tr>
<tr>
<td>10.3. Extending prediction horizon</td>
<td>63</td>
</tr>
<tr>
<td><strong>11. Conclusions and Future Work</strong></td>
<td>67</td>
</tr>
<tr>
<td><strong>Appendix</strong></td>
<td>71</td>
</tr>
<tr>
<td>A. Dynamic Filtering for Estimation of Transition Probabilities</td>
<td>71</td>
</tr>
<tr>
<td><strong>Bibliography</strong></td>
<td>73</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Decomposition of the traffic signal prediction problem</td>
</tr>
<tr>
<td>1.2</td>
<td>Vehicle speed estimation using signal prediction functions</td>
</tr>
<tr>
<td>1.3</td>
<td>Indecision zone</td>
</tr>
<tr>
<td>2.1</td>
<td>Vehicular and pedestrian movement directions</td>
</tr>
<tr>
<td>2.2</td>
<td>Cycle with two phases</td>
</tr>
<tr>
<td>2.3</td>
<td>Standard ring-and-barrier diagram</td>
</tr>
<tr>
<td>2.4</td>
<td>Architecture of a traffic signal system</td>
</tr>
<tr>
<td>2.5</td>
<td>Location of multiple detectors along the arterial road</td>
</tr>
<tr>
<td>2.6</td>
<td>Time-space diagram of the coordination mode</td>
</tr>
<tr>
<td>2.7</td>
<td>Application of passage time in the actuated mode</td>
</tr>
<tr>
<td>4.1</td>
<td>Map of the Mowry Avenue, Fremont, CA</td>
</tr>
<tr>
<td>4.2</td>
<td>Scheme of the intersection Mowry Ave and Blacow Rd, Fremont, CA</td>
</tr>
<tr>
<td>5.1</td>
<td>Data mining pipeline for traffic signal prediction</td>
</tr>
<tr>
<td>5.2</td>
<td>Illustration of windowing method</td>
</tr>
<tr>
<td>5.3</td>
<td>Basic steps of the windowing algorithm</td>
</tr>
<tr>
<td>5.4</td>
<td>Feed-forward neural network</td>
</tr>
<tr>
<td>5.5</td>
<td>Cluster representation of an FFNN</td>
</tr>
<tr>
<td>5.6</td>
<td>Schematic representation of a dynamical system</td>
</tr>
<tr>
<td>5.7</td>
<td>Time-delay recurrent neural network</td>
</tr>
<tr>
<td>5.8</td>
<td>Finite unfolding in time using shared weights</td>
</tr>
<tr>
<td>5.9</td>
<td>Generalized finite unfolding in time</td>
</tr>
<tr>
<td>5.10</td>
<td>Dataset partitioning and corresponding decision tree</td>
</tr>
<tr>
<td>6.1</td>
<td>Length of the phases 2 and 3 at different times of the day</td>
</tr>
<tr>
<td>7.1</td>
<td>Phase activity prediction scheme</td>
</tr>
<tr>
<td>7.2</td>
<td>Phase length prediction using classification</td>
</tr>
<tr>
<td>7.3</td>
<td>Prediction accuracy for different $\delta$</td>
</tr>
<tr>
<td>7.4</td>
<td>Prediction accuracy using neural networks for $\delta = 1$ and $\delta = 2$</td>
</tr>
<tr>
<td>7.5</td>
<td>Prediction accuracy using decision trees for $\delta = 1$ and $\delta = 2$</td>
</tr>
<tr>
<td>7.6</td>
<td>Phase activity prediction confidence</td>
</tr>
<tr>
<td>8.1</td>
<td>Histogram of misclassifications</td>
</tr>
<tr>
<td>9.1</td>
<td>Diagram of two intersections</td>
</tr>
<tr>
<td>9.2</td>
<td>Markov model describing the transition of vehicles</td>
</tr>
<tr>
<td>9.3</td>
<td>Recurrent neural network for traffic flow prediction</td>
</tr>
<tr>
<td>10.1</td>
<td>GLOSA integration flowchart</td>
</tr>
<tr>
<td>10.2</td>
<td>Output formatting for GLOSA application</td>
</tr>
<tr>
<td>10.3</td>
<td>RTCS integration flowchart</td>
</tr>
<tr>
<td>10.4</td>
<td>Multistage prediction process</td>
</tr>
<tr>
<td>10.5</td>
<td>RNN with overshooting</td>
</tr>
</tbody>
</table>
List of Tables

1.1. Car intelligence – controller intelligence taxonomy ........................................ 2
1.2. Functional requirements for ITS applications ............................................... 7

2.1. Characteristics and application fields of different control strategies .................. 16

4.1. Description of the fields in the original dataset ............................................. 25

5.1. Assignment of features to problems .............................................................. 30
5.2. Confidence levels and corresponding $\chi^2$ critical values ............................... 40

6.1. Phase length prediction results ................................................................. 43

7.1. Phase activity prediction results for selected applications ............................ 48

8.1. Next phase prediction results ................................................................. 52
1. Introduction

Present traffic control technology can be greatly enhanced to aid the reduction of energy consumption and air pollution by vehicles in major cities. The goal of this thesis is to formalize and solve specific problems of traffic signal prediction that would improve this technology. For a given movement direction, we want to be able to predict the traffic light signal at every moment in the time interval between few seconds and several minutes.

Since traffic light controllers are programmed to react on the immediate traffic load using complex algorithms, the solution for this problem is not trivial. The absolutely precise prediction of the traffic signal is impossible due to stochastic processes in vehicle movements. Because of the human factor, vehicles are determined to move in unpredictable way.

Instead of trying to create the universal predictor, we approach the problem from the application-centric perspective. We investigate the ITS applications that can profit from the traffic signal prediction and establish the requirements to prediction horizon and accuracy of the predictors.

Additionally, we decompose our complex problem into several smaller problems that can be solved using machine learning techniques. Figure 1.1 illustrates this decomposition. The problems 1–3 correspond to classical supervised learning problems: regression, binary classification, or multi-label classification. The problem 4 is a regression problem that requires advanced algorithms. We present two methods for the solution of the problem 4 using probabilistic approach and recurrent neural network.

![Figure 1.1: Decomposition of the traffic signal prediction problem into a number of simpler sub-problems. Arrows represent the direction of information flow.](image)

This application-centric problem decomposition approach together with the development of the feature extraction method represent the major contribution of this thesis.

1.1. Motivation

The invention of the car in the late 19th century marked the beginning of the new era of personal mobility. With the development of the automobile industry people gained a simple and affordable way to move over long distances, work far away from their homes, or move out...
Table 1.1.: Amount of available information for different categories of vehicles and infrastructure. We focus on the bottom right quadrant with traffic adaptive controllers and complex driver assistance applications in vehicles

<table>
<thead>
<tr>
<th>Controller Intelligence</th>
<th>Less</th>
<th>More</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car Intelligence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less</td>
<td>Fixed schedule, no information for driver</td>
<td>Traffic-dependent (actuated) signal control, no information for driver</td>
</tr>
<tr>
<td>More</td>
<td>Fixed schedule, information about signal length, speed advice</td>
<td>Variable schedule, signal length prediction, speed advice</td>
</tr>
</tbody>
</table>

of the cities into suburbs. This development, however, has also some negative effects with it. With the growing number of vehicles, air pollution and energy consumption by cars have become a severe problem of the 21th century.

According to statistics in present studies, vehicles are responsible for 16% [1] of the total CO₂ emission and 26% [2] of the total energy consumption in the EU. For the United States these numbers are 20% and 12% correspondingly [3]. The severity of the problem was recognized by the national and international governmental organizations and the first measures to prevent further escalation are already being taken [4].

Car accidents are another serious problem in the world today. In 2010, in Germany 288 297 traffic accidents with personal damages were registered, and 3 648 victims in these accidents were lethally injured [5].

Intelligent Transportation Systems (ITS) is a response to the problems resulting from the increasing use of vehicles. It aims to make the traffic circulation safer, cleaner, and more comfortable by applying information and communication technologies to the transport infrastructure and vehicles. There are numerous examples of ITS applications. Automatic road enforcement has already been helping authorities for years to prevent speeding and red light violations. Computer programs are helping users to find the optimal way to their destination using private or public transport. Other applications are in development or active research [1].

How can we support engineers in their development of ITS applications that would help to reduce air pollution and energy consumption by vehicles or to decrease the amount of accidents due to speeding and red light violation? Let us focus on one scenario: vehicle passing a regulated intersection. What happens if we can make this vehicle or the intersection itself “smarter”? Table 1.1 summarizes different options in a 2x2 matrix. A still very common situation is described in the upper left quadrant: no intelligence is exploited neither in the vehicle nor in the traffic controller. A traffic controller uses a fixed timing schedule and drivers have no information to predict the signal changes [1].

The status quo is described in the upper right quadrant. The traffic signal schedule depends on the current traffic situation (see Section 2.3), but drivers are not informed about the signal timing. These systems try to maximize throughput and minimize delay and so offer suboptimal solutions for the immediate situation. However, there is some space for improvement, since uninformed drivers still perform unnecessary acceleration and deceleration maneuvers.

The two bottom quadrants describe the current trend, where more and more sophisticated driver’s assistance systems find their way into vehicles. Whether with traffic actuated signal

1The major universities, like MIT, companies, like Siemens, and scientific societies, like IEEE, have their ITS branches.

2Also in the situations, where the traffic light has the same schedule for years and the drivers passing it at the same time every day know the exact schedule, this can work remarkably good.
control (bottom right quadrant) or not (bottom left quadrant), ITS applications can provide drivers with information about future traffic schedule and advise them about the speed required to pass intersections on green. The study conducted by AUDI AG and TUM in Ingolstadt, Germany, has shown that application of “intelligent” traffic controlling strategies together with speed advisory modules resulted in the reduction of CO₂-emission by 15% and fuel consumption by 17% [6].

We focus our interest on the bottom two quadrants. Using intelligent algorithms and communication with infrastructure, we want to enable vehicles to advise about the speed a driver should hold to pass the intersection in the green wave.

The following section continues the motivation for the study of traffic signal and traffic flow prediction models by introducing promising applications for ITS and traffic simulations.

1.2. Applications

The prediction methods developed in this thesis can be used in a number of ITS applications that may serve traffic participants and traffic engineers. This chapter reviews relevant applications for traffic signal prediction systems. The list of applications was adopted from the work of Koukoumidis et al. [7]. We extended this list and reconsidered the main requirements. The section presents a taxonomy of applications from the point of view of the required prediction horizon and accuracy.

1.2.1. Green Light Optimized Speed Advisory

A Green Light Optimized Speed Advisory (GLOSA) system estimates the optimal speed the driver should retain in order to pass the following intersections on green light. The ability to avoid unnecessary acceleration and deceleration in intersections with traffic lights not only helps to reduce the waiting and travel time, as well as driver’s frustration from waiting, but also helps to reduce air pollution and energy consumption by over 15% [6, 7].

In order to estimate the optimal speed, a GLOSA system requires the current location of the vehicle, the location of the stop line in the intersection, the residual time until the movement direction the vehicle follows remains active or becomes active, and the length of the queue in front of the stop line. The first and the second pieces of information can be retrieved from a GPS receiver and cartographical data [8]. One can estimate the length of the queue using information about the number and positions of sample vehicles [9]. Finally, the traffic signal predictor gives the remaining time.

Figure 1.2 shows the time-space diagram for a road with four intersections and the values of the green light prediction functions \( f_1, \ldots, f_4 \) for these intersections. The functions give the probability for green light depending on time in the future. Using the information from these functions, the GLOSA module in the vehicle can estimate the constant velocity (the slope of the red arrow) the driver has to maintain to pass all four intersections on green light.

In order to estimate the requirements to prediction horizon and accuracy, we assume that an average vehicle is traveling in the city with 45 km/h (approx. 12.5 m/s). The interval of 250 m is enough to safely and efficiently adjust the speed if the circumstances should require this. In this case the minimal prediction horizon should be 20 seconds. The prediction error should be less than 10% of the green time to avoid wasting it. For the green time of 50 seconds on the main directions in our dataset, it would lead to the required accuracy of 5 seconds.

1.2.2. Traffic Signal-Adaptive Navigation

The idea of Traffic Signal-Adaptive Navigation (TSAN) is related to that of GLOSA: computation of a route that is optimal not only regarding distance and travel time, but also regarding
Figure 1.2: Vehicle speed estimation using signal prediction functions. Functions $f_1, \ldots, f_4$ give the probability for green light on the intersections. The slope of the red arrow corresponds to the constant velocity required in order to pass the intersections in green wave.
1.2. Applications

waiting time. Depending on the route, the time spent in waiting at an intersection can be as much as several minutes [10]. Using the traffic signal schedule estimated from a traffic signal predictor, the greedy graph search algorithm for route computation can be easily extended to estimate the time of different detours with regard to the signal schedules [11] and to make suggestions for routes based on the total travel time (i.e. travel and waiting time together).

The prediction horizon for TSAN depends on the network structure. Koukomidis et al. suggest that the lead-up of 5 block or about 115 seconds for a vehicle driving 50 km/h, and a prediction error of less than 20% of average green time should provide sufficient boundaries for efficient routing. A higher prediction error would make TSAN to suggest unnecessary detours.

1.2.3. Red Light Duration Advisory

The Red Light Duration Advisory (RLDA) system aims to inform a driver about the estimated remaining red light time. Using this information, a driver can decide to turn off the engine to reduce energy consumption and air pollution. According to a study conducted by American Society of Mechanical Engineers, the amount of fuel required to restart the engine is equal to the amount spent during 6 seconds idling with AC on. Savings from turning off the engine can accumulate to as much as 25 cents per day [12].

The prediction accuracy for RLDA needs to be better than 5 seconds so that turning off the engine would actually safe fuel. The prediction horizon should be at least 20% of the average red light time, which is approximately 10 seconds.

1.2.4. Imminent Red Light Advisory

Imminent Red Light Advisory (IRLA) informs the driver about the remaining green time so that she can make a decision not to accelerate and impair safety. The operation of IRLA is similar to GLOSA. Therefore, we consider the requirements to IRLA to be the same as to GLOSA. However, since the safety is paramount to the comfort, even higher accuracy may be required.

1.2.5. Red Light Violation Advisory

Additionally to IRLA, Red Light Violation Advisory (RLVA) system warns the driver about the danger of red light violation if the vehicle is approaching at high speed an intersection that is about to turn red. This application helps to avoid red light violations or an unnecessary emergency breaking. The prediction horizon of the application is just a few seconds, however the requirements to the accuracy should also be higher. In our opinion, the allowance of the accuracy of 5 seconds suggested by Koukomidis et al. would hurt the confidence into the system and make drivers to ignore the advices. We suggest that an accuracy of less then 2 seconds is more appropriate.

1.2.6. Realistic Traffic Control in Simulations

Traffic simulation is an important topic in intelligent transportation research and industry. A number of open-source systems for traffic simulations offers the tools for realistic large-scale simulations on macroscopic and mesoscopic scales [13][14]. Currently, this open-source systems offer only simplified traffic control algorithms that do not react on the traffic flow in the same way the real modern traffic controller do. Realistic Traffic Control in Simulations (RTCS) offers the real-world traffic controller emulators that behave in exactly the same way as the real traffic controller in the simulated areas.

Using machine learning techniques, RTCS extracts the algorithm logic from the historical logs of traffic light changes in the area of interest in a completely automatic fashion, and it is
able to adapt to the changes in traffic controller firmware without human interference, when log data from the traffic controllers is provided.

The prediction horizon and accuracy requirements depend on the temporal resolution of the simulations. For a second-by-second simulation they both are equal to 1 second.

1.2.7. Safe Traffic Phase Transition

In the scope of this thesis, we develop prediction algorithms for very short term traffic flow prediction. This information is not only actively used by traffic signal prediction algorithms as shown in Chapters 6, 8, and 7, but it is also valuable for traffic engineers. It allows one to schedule the change of the traffic lights during the time interval when no vehicles will arrive from the active direction. This would minimize the number of drivers who would want to accelerate on yellow light. Safe Traffic Phase Transition (STPT) system would increase the safety of intersections and satisfaction of drivers.

In conversation with Siemens traffic engineers we established that the prediction horizon of interest should be at least 10 seconds. The accuracy of the prediction for vehicle arrival times should be at least 3 seconds which is equal to the beginning of the indecision zone (see Figure 1.3). If a driver is less than 3 seconds away from the intersection, very probably he will be going to pass it event if the traffic light switches to yellow.\footnote{Chang et al. found that 85\% of drivers stopped if they were more than 3 seconds from the stop line, regardless of their speed.\cite{15}}

Figure 1.3.: Illustration of an indecision zone in an intersection. If the traffic light turns yellow, drivers will most probably pass if they are less than 3 seconds away from the intersection, and they will most probably stop if they are more than 5.5 seconds away from the intersection. The behavior of drivers in the indecision zone is unpredictable.

1.2.8. Application Categorization

Table 1.2 summarizes the prediction horizon and accuracy requirements of the different applications described above. We can divide these applications into three categories. The first category consists of the application that require very accurate predictions in the next 10 seconds or less: RTCS, RLVA, and TSPT. The applications that tolerate the prediction error of 5 seconds or lower for the prediction horizon between 10 and 20 seconds make up the second...
1.3. Challenges

Table 1.2.: Categorization of functional requirements for ITS applications. Time horizon denotes the time interval in the future for which predictions should be available. Accuracy denotes the maximal prediction error allowed.

<table>
<thead>
<tr>
<th>Application</th>
<th>Time horizon [s]</th>
<th>Accuracy [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCS</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RLVA</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>TSPT</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>RLDA</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>GLOSA</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>IRLA</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>TSAN</td>
<td>115</td>
<td>10</td>
</tr>
</tbody>
</table>

category: RLDA, GLOSA, and IRLA. The last category contains only one application, TSAN, that requires the traffic signal prediction for the next couple of minutes, but it is satisfied with less precise forecasts, as long as the prediction error is not greater than 10 seconds.

Table 1.2 presents the functional requirements to our predictors, e.g. we expect a predictor to work properly in the productive environment if for a given time horizon it can forecast with the prediction error equal or less to those defined in the “Accuracy” column.

1.3. Challenges

Developing a traffic signal prediction system, we have to bear in mind that it has to operate in a very heterogeneous environment. The environment imposes additional nonfunctional requirements to the prediction system, besides prediction horizon and accuracy described above.

Because traffic light controllers are of many types and have a long life cycle, a signal prediction system should be able to handle a variety of different traffic controller hardware and software systems without need to make big adjustments in the infrastructure. For example, it should be feasible to extend the system with an additional transmitter for information broadcasting, but we can not expect that old traffic controllers will be replaced just according to the demands of our ITS applications.

The same also concerns traffic detectors. We can not expect the availability of state of the art traffic detectors with speed and occupancy time information, recognition of individual vehicles, etc. The most widely used kind of detector today—single-loop traffic detector—offers only rudimentary information (see Section 2.2.2). This is the minimum amount of information about traffic flow we can use. However, as the sensor and communication technologies are being permanently developed, the simple integration of new sources of information into the models should also be possible (extendability).

Moreover, for a given traffic controller the prediction for different modes of operation is necessary. In the long term the traffic environment often changes, e.g., an opening of a mall or a park house would change traffic patterns and accordingly the traffic actuated signal schedule needs to change. Therefore, a traffic signal prediction system needs to be able to adapt to new traffic conditions (adaptability).

Finally, the manual work for deployment of the prediction system should be minimal, as even 30 man-minutes for the setup of a system for every traffic controller can lead man-years for the deployment in the dimensions of the whole city. This can be a significant obstacle to integration of such systems.
1. Introduction

1.4. Scope and limits of the work

In this work we primarily concentrate on the study of the traffic signal prediction problem and the areas of its application. Unlike other studies that come from the traffic engineering and electrical engineering domains, in this computer science project we ignore the technical details of car-to-infrastructure communication, but establish the functional (Section 1.2) and non-functional (Section 1.3) requirements, and then we design a solution that would satisfy them.

We do not focus on the comparison of a large number of machine learning methods. Where it was feasible and intelligible, we compared parametric methods (neural networks) with non-parametric (k-nearest neighbors) and nonmetric (decision trees). The machine learning theory offers a large number of other methods for regression and classification: kernel regression, support vector machines, conditional random fields, hidden Markov models, just to name a few. We deliberately rejected the idea of comparison of the numerous methods with exhaustive parameter search, as this time consuming activity would significantly reduce the scope of problems we could investigate, while the conclusions would be only applicable for a concrete dataset and the generalization of any claim would be questionable.

There is no doubt that the proper problem formulation and a good set of features can improve the prediction accuracy much more than, for example, the substitution of neural networks by SVN. Therefore, we focused our attention on the last.

Furthermore, we do not investigate the cases where the normal traffic signal schedule is distorted due to preemptive calls from a fire truck or an emergency vehicle. The cases of preemptive calls occur seldom enough to be treated as outliers at first, e.g. in the available dataset, we did not have any single case of a preemptive call. In future, one could study the algorithms for preemptive calls handling as a special case of the traffic signal prediction problem.

This project was conducted in cooperation with Siemens Corporate Research and BMW Group. Since the domain specific knowledge of Siemens traffic engineers and the dataset we used for evaluation origin from the United States, in the thesis we studied and cited many US-specific regulations and manuals. “Richtlinien für Lichtsignalanlagen” describes the traffic actuated signal control rules and guidelines in German transportation system [16]. Since the fundamental principles of the traffic actuated control in both countries are similar, we assume that the results would hold if the methods would be evaluated on the data from Germany.

1.5. Structure overview

This thesis is organized as follows. Chapters 2-4 represent an introductory part. Chapter 2 imparts the domain specific knowledge required for understanding of this work. Chapter 3 reviews the related work. Chapter 4 presents the dataset used for evaluations. Chapter 5 concludes the introductory part of this thesis with the descriptions of the machine learning methods used in the thesis.

Chapters 6-10 represent the main contribution. Since predictions required for the applications of interest are complex, we pursue the approach of dividing it into several simpler sub-problems, solving them separately and combining the resulting predictors into an overall solution. The prediction of the length of the green time can be approached as a regression problem (described in Chapter 6) or as a classification problem in discretized time (described in Chapter 7). Moreover, the generalized problem of signal timing prediction requires the estimation of the next green direction (called phase), because the assumption that phases have a constant order falls short under real-world circumstances. We describe the methods for the prediction of the next active direction in Chapter 8. Since modern traffic controllers are programmed to react to the current traffic situation, the accuracy of the prediction models heavily
depends on the availability of the traffic data predictions. Therefore, this thesis would not be complete without a discussion of the methods for very short term traffic flow prediction, which the reader can find in Chapter 9. Overall, Chapters 6-9 contain the formal specification of the problems, the description of the models that use the methods outlined in Chapter 5, and the evaluation of the approaches on the test dataset. The contribution part of the thesis concludes with the description of integration of the developed predictors into a solution of the complex problems in Chapter 10.

The final chapter gives the summary and discussion of the results. It also proposes directions for future work.
1. Introduction
2. Background Information

The design of intersections plays an important role to ensure the safety and high capacity on the roads. The signalized intersections, which means the intersections with traffic light controllers, for years work as an important instrument of safe traffic management. This safety comes to the price of controller delays, although, the modern algorithms for traffic control attempt to maximize the total traffic flow through the intersections acquiring the information about their environment from a system of vehicle detectors.

This chapter describes the fundamental principles of traffic signal controlling as well as explains the terminology used in the following chapters of the thesis.

2.1. Introduction of the terminology

In this section we introduce the terminology of the traffic engineering domain used throughout this thesis. Some terms, i.e. “call” or “phase”, are overloaded and overused in the literature, which unavoidably leads to ambiguity and misunderstanding. In the thesis, we try to provide consistent terminology free from overloading. The reader is warned, however, that in the literature also other terms for the same definitions may be used.

Consider the intersection diagram illustrated in Figure 2.1. The arrows show the driving directions regulated by traffic light. A right turn does not require a dedicated traffic light and is usually merged with a corresponding straight forward direction. Left turn lanes may have designated traffic lights that permit and sometimes prohibit the turns. The driving directions are usually numbered so that the left turns have odd numbers, and straight forward directions have even numbers. It is a common practice to assign numbers 2 and 6 to the directions along the arterial road.

Pedestrian crossings are also numbered according to the movement directions, with which they can be performed concurrently. So in Figure 2.1 the pedestrian crossing number 4 is on the western leg of the intersection.

Important in the scope of this thesis is the definition of signal phase—the right-of-way, yellow change, and red clearance intervals in a cycle that are assigned to an independent traffic movement or a combination of traffic movements [17]. In other words, phase is a period of time when the movement in one or several directions is allowed. It is commonly used to associate phases with the numbers of movement directions they allow.

The sequence of all phases is called cycle. Figure 2.2 presents a time diagram of a cycle with two phases. Each phase consists of the green, yellow, and all-red (or clearance red) intervals. All-red is a short period of time when all movement directions receive red signal, which allows the last vehicles to clear the intersection.

It is a common practice to group the phases in a continuous loop called ring. The phases of the conflicting movement directions are separated from each other either by time intervals, when they are arranged in the same ring, or by barriers, when they are arranged in different rings.

Figure 2.3 illustrates the ring-and-barrier diagram of a standard intersection configuration used in this thesis. The intersections we consider in this thesis have two rings: ring 1 with phases 1, 2, 3, and 4; and ring 2 with phases 5, 6, 7, and 8. The phases 1, 2, 5, and 6 are located behind the barrier 1. In this case the phases 1 and 2 can operate concurrently to the phases 5 and 6. Behind the barrier 2 we have phases 3, 4, 7, and 8. Again, the phases 3 and 4 can operate...
2. Background Information

Figure 2.1.: Vehicular and pedestrian movement directions on an intersection. Left turns have odd numbers, straight forward movements and right turns have even numbers. Pedestrian movements have the number of their next parallel straight forward vehicle movements.

Figure 2.2.: Cycle with two phases. The combination of green, yellow (Yel.) and all-red (AR) times builds a phase. The sequence of all conflicting phases builds a cycle.
2.2. Fundamental principles and architecture of intersections

Objectives of the signal timing design

The Manual on Uniform Traffic Control Devices (MUTCD) defines a traffic control signal as any highway traffic signal that directs the traffic to stop and proceed alternatively [17]. Since there are numerous traffic types: motorvehicles, bicycles, pedestrians, public transport, ridden and herded animals, railway trains, etc, traffic signals play an important role in the road network design. According to the study by National Transportations Operation Coalition, there are more than 272,000 traffic signals in the United States [18].

The Signal Timing Manual defines following objectives for the design and timing of traffic signals [10]:

1. Provide for the orderly and efficient movement of people.
2. Effectively maximize the volume of movements served at the intersection.
3. Reduce the frequency and severity of certain types of crashes.
4. Provide appropriate levels of accessibility for pedestrians and side street traffic.

Obviously, the main goals of the traffic signal design, i.e. safety and efficiency, can conflict in many cases. It appears that in some cases safety has to be sacrificed to meet the increasing demands for higher traffic capacity on the highways.

The design and timing of the intersection traffic signals is the subject of governmental regulations, of needs and demands of local authorities, as well as of experience and judgment of traffic engineers.

Figure 2.3.: Standard ring-and-barrier diagram. Phases are separated in two rings. The phases in different rings behind a barrier, e.g. phases 1 and 6, can be activated concurrently, but the last two phases behind a barrier, e.g. phases 2 and 6, should end at the same time.
2. Background Information

2.2.1. Traffic signal controllers: components and architecture

Depending on the available infrastructure, a traffic signal system can consist of a large number of different components. Figure 2.4 illustrates the four main components of a traffic signal system [10]:

1. **Traffic light displays** are the only part of the signal system a driver or a pedestrian usually see. They help to regulate traffic by displaying permissive or prohibitive signals.

2. **Vehicle detectors**, usually in form of induction loops placed under the road surface, serve to register the traffic demand. Every time a vehicle is passing the loop it generates an impulse transmitted to the local controller.

3. A **local controller** operates the displays according to the configuration defined by traffic engineer and to the information from vehicle detectors. Local controllers implement the signal timing schedules and strategies described below in this chapter.

4. A **traffic control center** is responsible for collecting the operational data from the local controllers, it performs the system monitoring, synchronization, and allows to alter the timing parameters and strategies.

2.2.2. Detectors

Detectors are used by signal controllers to register the user’s demand. Every time a pedestrian is pushing a signal request button or a vehicle is passing a detector, the special signal, *call*, is sent to the controller. The Traffic Detector Handbook [19] describes the variety of vehicle detection technologies as well as discusses their strengths and weaknesses. In this section we
give only a short introduction to the detection technologies required for understanding of the thesis and refer to the Handbook for further information.

The vehicle detectors serve the fulfillment of safe and efficient controlling strategies. On the one hand, they identify vehicle presence on the active phas. Thus, they help to avoid the situation, when a driver is in the indecision zone while the traffic light turns yellow, and to ensure a safe phase termination for high-speed movements. On the other hand, detectors give the information needed to serve the vehicles queues at the stop line and those progressed from upstream traffic signals, detectors also help to identify gaps in traffic flow, where the phase may be ended.

Depending on the detector’s purpose, technology, and intersection approach speed, detectors are usually located at the stop line or on the upstream as Figure 2.5 shows. Stop bar detection is used to clear the queues and the multiple upstream detectors are used to safely terminate the phase.

![Figure 2.5: Multiple detectors along the arterial road. There is a number of detectors along the road: advanced, left turns and stop-line detectors.](image)

The most widely used technology for vehicle detection is an induction loop. An insulated, electrically conducting loop with permanently circulated electric current is located under the surface of the road. When a metal vehicle passes close to this wire loop, the inductance of the loop changes and the traffic light controller begins to register the call. Besides the standard loop detectors, a number of alternative technologies was invented and tested, including video, microwave, etc. But until now, they have not found wide-ranging acceptance.

The loop detectors can operate in one of two modes: pulse or presence. In the pulse mode, the detector sends a 0.1 or 0.15 seconds long “on” signal to the controller indicating the passage of a vehicle (point detection). In the presence mode, the occupancy is measured, the call starts with the arrival of the vehicle and ends when the vehicle leaves the detection zone.

The detector in pulse mode is able only to count the approaching vehicles, while the detector in presence mode can also provide the information about the approaching speed. However, the estimation of the speed using only one loop has to be based on the assumption of the average vehicles length and so tends to be unreliable. For the estimation of speed, the information from a sequence of loop detectors has to be used.
Table 2.1.: Characteristics and application fields of different actuated control strategies (from [10]).

<table>
<thead>
<tr>
<th>Operation strategy</th>
<th>Semi-acted</th>
<th>Fully-acted</th>
<th>Coordinated-acted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Length?</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Conditions Where Applicable</td>
<td>Where defaulting to one movement is desirable, major road is posted less than 40 mph and cross road carries light traffic demand</td>
<td>Where detection is provided on all approaches, isolated locations where posted speed is greater than 40 mph</td>
<td>Arterial roads where traffic is heavy and adjacent intersections are nearby</td>
</tr>
<tr>
<td>Application areas</td>
<td>Highway operations</td>
<td>Locations without nearby signals; rural, high speed locations; intersection of two arterials</td>
<td>Suburban arterial</td>
</tr>
<tr>
<td>Key Benefits</td>
<td>Lower cost for highway maintenance</td>
<td>Responsive to changing traffic patterns, efficient allocation of green time, reduced delay and improved safety</td>
<td>Lower arterial delay, potential reduction in delay for the system depending on the settings</td>
</tr>
</tbody>
</table>

2.3. Controller modes

Depending on the hardware availability, traffic load, and distance between intersections, a traffic engineer can set the controller to operate in pre-timed or actuated mode. We give only a brief overview of the pre-timed mode to make this description complete, but we are not interested in the development of prediction methods for this operation mode. In actuated mode the controller can implement fully-actuated, semi-actuated, or coordinated-actuated strategy. The understanding of the principles of actuation and coordination is essential for the development of a traffic signal prediction system.

2.3.1. Pre-timed control

This operating mode has a fixed schedule. It does not respond to the current traffic demand, although, using different schedules for am/pm peaks and the rest of the day, it can reflect the general patterns in the traffic flow. Since requirements to the infrastructure are minimal, pre-timed mode is used on intersections where controllers are old or where vehicle detectors are not available. Nevertheless, pre-timed mode allows the phases synchronization between the controllers in adjacent intersections in order to create a “green wave” in a way similar to coordination described below.

Since pre-timed mode can be seen as the subclass of coordinated-actuated strategy with constant demand on all phases, we do not discuss the prediction methods for this mode separately. However, because of its determinism, the prediction of the fixed schedule is trivial given the historical data and does not require the application of machine learning methods.
2.3. Controller modes

2.3.2. Actuated control

Actuated signal control mode determines the length of a phase depending on the current traffic demand. Depending on the available infrastructure, the location of the intersections, traffic patterns, and the time of the day, a traffic engineer can program the controller to implement one of three controlling strategies: fully-actuated, semi-actuated, or coordinated-actuated. Table 2.1 gives the overview of these modes and this section discusses them in more details.

In the later chapters we focus on fully-actuated and coordinated-actuated strategies. For the sake of simplicity we will refer to coordinated-actuated strategy as to coordinated mode, and to fully-actuated strategy as to actuated mode.

Fully-actuated strategy

In the fully-actuated (or free) control mode the timing for all four directions on an intersection is computed according to the detector information. Therefore, it is often used in isolated intersections with unstable traffic patterns or during the night periods, when the traffic volume is low.

Since this control strategy is highly responsive to the traffic demand and changes in the traffic patterns, it is able to reduce delays compared to pre-timed control. It allows the cycle time to be efficiently allocated on the cycle-by-cycle basis and to skip individual phases, if no demand is present.

Semi-actuated strategy

In the semi-actuated control mode the detectors are placed only on the minor movement directions of the intersection. The major-road phases are not actuated, which means that the traffic light dwells in these phases until the demand from minor movement phases is registered.

Semi-actuated control takes place at intersections that are a part of a coordinated-actuated arterial street system (see below). But unlike coordinated-actuated mode, semi-actuated control does not have fixed cycle length and explicit synchronization of phases. It also can serve on isolated intersections with a low-speed major load and lighter crossroads. But it can fail in the case of continuous demand on the phases associated with one or more minor movements.

Coordinated-actuated strategy

Coordinated-actuated control mode is used to synchronize a sequence of intersections in order to create a “green wave”—a situation at which a platoon of vehicles can pass all intersection at the green light. Usually, coordination takes place during the time with higher traffic density on the arterial streets.

Figure 2.6 illustrates the work of the coordinated intersections on a time-space diagram. A traffic engineer selects the fixed cycle length and the offset between the start of the cycles in a way to maximize the traffic flow in the green wave. In Figure 2.6 the traffic flow is proportional to the distance between two parallel arrows.

Most of the time, it is enough to coordinate only the arterial road of the intersection. In this case we speak of an coordinated-actuated system. The phases on the secondary road are served according to the demand and the unused time is added at the end of the coordinated phase. Though, for example, if there is no demand for a phase, it can be left out, the total length of the cycles measured between the ends of the phase 2 is always fixed. Since the synchronization of the intersections is made in order to create a green wave along the corresponding movement

---

1This way of coordination is called permissive mode. It is one of the most generally used methods, which was also used in our dataset. For the overview of other available methods we refer to the literature (e.g. [20]).
2. Background Information

Figure 2.6.: Time-space diagram of the coordination mode. Blue arrows denote the first and the last vehicle in platoon that can pass intersections $C_1, C_2, C_3$ in green wave. The distance between two parallel arrows corresponds to the size of the platoon.

directions, it is ensured that the coordinated phases serve at least the planned time or even more.

Usually, traffic engineers use three different configurations of cycle length and offset in order to respond on traffic demand at am/pm peaks as well as in remaining time. This configurations are called dials, and a controller program usually contains between three and six of them.

**Principle of signal actuation**

The principle of signal actuation is illustrated in Figure 2.7. Using the information from the detectors, controller selects the next phase in the ring with the registered recall from a vehicle or pedestrian detector. Once selected, the phase is served at least the minimum green amount of time. After the minimum green time was served, the controller continues to serve green as long as the detectors continue to register arriving vehicles. If there is no demand, the controller waits for a time interval defined by the parameter passage time and, if no other vehicles pass the intersection, it gaps out and selects the next phase to serve.

In the coordinated-actuated mode, the controller can terminate the phase even in presence of demand if the length of the phase exceeds the value of the maximum green parameter. For the semi-actuated and fully-actuated control strategies, there is no limits for the length of individual phases. If there is no demand from any direction in the ring, in the semi-actuated mode the controller activates the straight forward phases along the arterial road using an artificially placed recall.

\[\text{Sometimes in practice a more complex setting with two different maximum green timers can occur. We deliberately avoid the complex rules for the sake of clarity. In the case study we have the simple case with only one maximum green parameter.}\]
Figure 2.7.: Application of passage time in the actuated mode. After serving minimum green time, traffic controller extends the phase after detecting four vehicles. After the last vehicle passed, traffic controller gaps out before maximum green time was achieved.
2. Background Information
3. Related Work

The last three decades have been rich in research in the areas of traffic signal and traffic flow prediction. The analytical model for estimating the phase and cycle lengths depending on the intersection properties and traffic situation was described in Highway Capacity Manual \[21\] and refined by a number of studies \[22–26\]. This research is primarily directed towards developing formulas for estimation of the optimal phase and cycle lengths that would increase the performance of isolated and coordinated intersections depending on their geometry, infrastructure and traffic demand.

The resulting analytical models are not directly applicable to our problem since they are employed at the design stage of the traffic controller configuration and are used to establish the average values for signal timing. In contrast, we want to estimate the values at the particular moment of time or in the very near future. These studies can offer valuable ideas for features that we can use in our algorithms, although the estimation of the variables used in the analytical models is often not feasible in the real world. Just one example: the estimation of the capacity rate requires knowledge about headaway distances that can be measured with occupancy detectors, but not with the most commonly used pulse single-loop detectors (see Section 2.2.2 for the description of the detector technologies).

Traffic prediction research is directed towards forecasting of relevant traffic variables, e.g. volume, occupancy, and speed, depending on the traffic conditions and historical data. It was the subject of active research in the past decades (see the survey by Vlahogianni et al. \[27\] for a comprehensive overview). A lot of ideas were studied including as challenging as forecasting of relevant traffic variables with single-loop detectors only \[28, 29\].

The time resolution of the short term traffic prediction usually ranges between 5 and 20 minutes \[27\]. In fact, the Highway Capacity Manual indicates that 15 minutes intervals are the best for prediction, since the traffic flow exhibits strong fluctuations if the intervals are shorter \[30\]. Unfortunately, we need to predict the traffic arrivals with the time resolution of several seconds.

To the best of our knowledge, only two studies, by Head \[31\] and Chang and Su \[32\], were conducted for the very short term predictions, although, those were carried out with simulated and not with real data. Head used uplink detectors and traffic flow model for arrival time estimations. The model assumes the knowledge of the approach speeds and link flow speeds of the vehicles that can be obtained from occupancy detectors, but in our case only the information from the pulse single-loop detectors is available. Chang and Su utilize neural network models for prediction of the intersection queue length evolution, integrating the data from adjacent intersections. In this sense, their approach resembles the study of traffic flow prediction using spatial-temporal unfolding conducted in the scope of this thesis.

The adaptive traffic control is an area of research that develops models which are able to adjust the signal timing to the current traffic situation by optimizing the given objective function, e.g. by reducing time delay \[33–35\]. Systems like RHODES \[36\] and OPAC \[37\] apply traffic flow models to predict vehicle arrivals at intersections. Using peer-to-peer interchange of traffic volumes between intersections, collected from upstream and stop-line detectors, RHODES is trying to forecast the impact of traffic arriving for up to 1 minute upstream and adjust the signal timing and phase sequences. OPAC is a congestion control strategy that attempts to maximize throughput by adjusting signal timing parameters using predictive optimization with a rolling horizon with measured and modeled traffic demands. These approaches for traffic control use the idea of traffic flow passing that also found its place in our work.
3. Related Work

In the recent years the research of traffic signal prediction for ITS applications became more active. In 2009 the Institute for Traffic Engineering (Institut für Verkehrswochen) at the Technische Universität München in cooperation with Audi conducted the research project TRAVOLUTION that investigated the chances for the application of car-infrastructure communication in the traffic system of Ingolstadt [38]. In the first part of the project, the online-optimization of network-wide traffic signal control was performed using evolutionary optimization algorithm GALOP. In the second part, the application of GLOSA and RLDA was tested using traffic light controller to car communication. The authors of the paper note that hidden Markov models were used to predict the probability of signal change in the future. Braun et al. report that they succeeded to predict phases changes for between 10 and 60 seconds in the future with an accuracy of less than 1 second in most of the cases. Although we think that this exceptional accuracy is only possible in the cases with coordinated mode and stable traffic pattern. The hidden Markov model described in complimentary paper [39] requires the fixed cycle length and the enumeration of all possible states and phase transitions described by an expert for every second of the cycle. These limitations can be a significant obstacles to integration of such prediction systems for a large number of traffic controllers with different configurations and operational modes.

The recent study by Koukoumidis et al. describes the smartphone application SignalGuru developed to recognize the phase changes from the video sequences and to use car-to-car communication for aggregation and estimation of the signal timing schedule for GLOSA [7]. The authors describe the algorithm for predicting phase lengths with remarkable accuracy between 1 and 2 seconds using the historical data about the length of previous phases and cycles obtained from the traffic controller history in Singapore. The dataset used in the paper would be a good benchmark for our traffic signal prediction techniques, but, unfortunately, we could not acquire it to compare the results. Using the data described in Chapter 4, we obtained the results similar to those reported by the authors if only the data from coordinated mode were used (average absolute error of 1.9 s–1.2 s reported by Koukoumidis et al. compared to 1.4 s reported in Section 6.2).

Though the last three decades have been rich in research in the areas of traffic signal and traffic flow prediction, most of the results are not related to the applications discussed in this thesis. Only two works by Koukoumidis et al. and Braun et al. suggest the solutions for similar problems.
4. Dataset Description

From our collaboration partner BMW Group we obtained a dataset with log data from traffic light controllers in 6 adjacent intersections between Farwell Drive and Fremont Boulevard along Mowry Avenue in Fremont, CA, USA (see Figure 4.1). The data were collected between August 19, 2010 and August 27, 2010 so that it contains the traffic behavior for workdays and weekends, but not for the holidays. The dataset contains 4,493,365 entries in total.

Unfortunately, we did not receive the exact description of the measurement process so that we had to substitute some of the information by educated assumptions. So, usually, the measurements were taken about 60 times per minute, but the time between the measurements is not equidistant. The measurements contain gaps that probably were caused by the interruptions due to that the communication with the control center. We removed the data that contained gaps from the dataset. Most of the gaps, however, occurred during the night and so were not critical.

Figure 4.1.: Map of the Mowry Avenue, Fremont, CA, where the data for evaluation of the prediction methods were taken.

Figure 4.2 illustrates the intersection of Mowry Avenue and Blacow Road, which we used for the evaluation of the prediction methods from Chapters 6 to 8. All directions have numbers between 1 and 8 and the number of arrows corresponds to the number of lanes in each direction. The directions along the arterial road Mowry Ave have number 2 and 6.

Table 4 summarizes the main fields of the dataset. The field “vehicleActuation” that carries the information from the vehicle detectors was truncated to the first 8 bits, even though the intersections have between 14 and 18 vehicle detectors. Therefore, the detailed information about the traffic flow on the arterial streets was lost. Instead, we used the field “vehicleCall” that contains the summarized information from all vehicle detectors in one movement direction (e.g. advanced and stop line detectors on all lanes) as well as recalls.
Figure 4.2.: Scheme of the intersection Mowry Ave and Blacow Rd, Fremont, CA, used for evaluation of the methods in Chapters 6 to 8.
Table 4.1.: Description of the fields in the original dataset used for evaluation of the prediction methods.

<table>
<thead>
<tr>
<th>Field</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>varchar(32)</td>
<td>IP address of the traffic light</td>
</tr>
<tr>
<td>timeStamp</td>
<td>varchar(32)</td>
<td>The time in UTC when the live data from the light was recorded. Format: YY.MM.DD:HH:MM:SS:mmm (where m = milliseconds)</td>
</tr>
<tr>
<td>coordinationInControl</td>
<td>boolean</td>
<td>Marks if the light is running in free mode (false) or if the coordination is in control (true)</td>
</tr>
<tr>
<td>dialIndexes</td>
<td>smallint unsigned</td>
<td>The dial index (1, 2, or 3) of the configuration for coordinated mode or NULL for actuated mode</td>
</tr>
<tr>
<td>cycleTime</td>
<td>smallint unsigned</td>
<td>Time in the current cycle in seconds</td>
</tr>
<tr>
<td>phaseOnStatus</td>
<td>binary(8)</td>
<td>Binary vector indicating if the phase is active for each of 8 phases</td>
</tr>
<tr>
<td>ring1Status</td>
<td>smallint unsigned</td>
<td>Encoded status of the traffic light of right 1, e.g. green, yellow, or red</td>
</tr>
<tr>
<td>ring2Status</td>
<td>smallint unsigned</td>
<td>Encoded status of the traffic light of right 2, e.g. green, yellow, or red</td>
</tr>
<tr>
<td>vehicleCall</td>
<td>binary(8)</td>
<td>Binary vector indicating if the vehicle call (from detector or recall) is active for each of 8 phases</td>
</tr>
<tr>
<td>pedCall</td>
<td>binary(8)</td>
<td>Binary vector indicating if the pedestrian call is active for each of 8 phases</td>
</tr>
<tr>
<td>vehicleActuation</td>
<td>binary(8)</td>
<td>Binary vector indicating if the vehicle detector is registering a car, the input is truncated to the first 8 bits</td>
</tr>
</tbody>
</table>

Figure 5.1 shows the data mining pipeline for traffic signal prediction adapted from the book by Niemann [40, p. 26]. The raw data obtained from the traffic controller’s log contains duplicate entries and gaps that can hinder the performance of the learning algorithms. Therefore, in the pre-processing step the data is cleaned from duplicates, rediscretized into equidistant intervals of one second and the gaps in the records are identified and filled if the information about the values in the gaps is available.

After the pre-processing step we have a sparse set of binary values. If we apply our models to such inputs, the results will be disappointing. Therefore, in the feature extraction step, the data is aggregated and more condensed information (the features) is extracted. Moreover, the windowing method segments the time series data into a set of patterns with historical data and target values that can be used by machine learning prediction paradigms.

According to the available features and the problem statement, we create mathematical models with a number of parameters that are calculated to fit the models to observed data using machine learning techniques, e.g. k-nearest neighbors, neural networks, or decision trees.

Now we can evaluate the resulting predictor by looking at the learned values for the free parameters and/or by estimating its generalization performance on a before unseen dataset. If we are satisfied with the results, we can use the predictor in traffic signal applications.

The following sections describe in detail how we applied the individual steps for traffic signal prediction.

Figure 5.1.: Data mining pipeline for traffic signal prediction (from [40 p. 26]).

5.1. Pre-processing and feature extraction

5.1.1. Windowing

The data originally recorded from the traffic signal controller is not suitable for training. The measurements are irregular and contain gaps, the information is sparse and noisy. The first step is, therefore, to prepare it for the machine learning methods by rediscretizing the measurements, removing artifacts, aggregating sparse information, and rearranging, the data in a suitable form.

Based on knowledge that the measurements were taken as close as possible to the second-by-second basis, we rounded the milliseconds in timestamps to the nearest second. Obviously, this rediscretization method has its drawbacks since it can produce the duplicate entries, when two measurements point to the same timestamp, or small gaps, when the time between two
measurements is two seconds. However, the trial on the data shows that this occurs relatively rare. We take care of these gaps and duplicates in our windowing algorithm.

The windowing method transforms a time-series into a set of data patterns as illustrated in Figure 5.3: for the time \( t \) the previous outputs \( t - 1, t - 2, \ldots \) are assigned to the variables \( v_1, v_2, \ldots \). Since we expect the importance of the previous information to decrease with time, we look just on a finite time window of the size \( W \), e.g. 2 entries in our example in Figure 5.3. The windowing method is generic and does not depend on the model we want to use for prediction.

For this thesis we implemented the double-window algorithm—an extension of the windowing method described above that uses two windows, for the past and for the future, and can handle two different datasets, e.g. if the input data in the window “past” were measured in one intersection, and the output data in the window “future” were measured in another.

The algorithm uses the concept of generators—the data-structure supported by Python, C#, Ruby, and a number of other programming languages [41]. A generator is a “lazy” iterator that allows one to iterate through a data-structure without the need to store the complete data-structure in memory. It can also generate the new elements “on fly”, which is very useful for processing large datasets.

Figure 5.3 shows the basic steps performed by the generator at each iterating step. At first, the new entry from the dataset is identified and added to the window of data we work with (Figure 5.3(a)). If the new entry has the same timestamp as the previous one, we perform the routine for handling duplicates: in simplest case just removing the current entry and jumping back to window identification (Figure 5.3(b)). If the new entry was measured more than 1 second after the previous one, we call the routine for handling the gaps again, in the simplest case, we just create a “virtual” measurement with the same data as in the last entry (Figure 5.3(c)). Finally, we output the values (Figure 5.3(d)). The output can be non-trivial. For example, we can aggregate the vehicle calls over a period of time to estimate traffic flow, or we can apply moving average for smoothing. The generators are implemented generic and can process arbitrary data using different hook methods for gaps and duplicates handling, as well as for output formatting.

The double window algorithm iterates over two (independent) generators so that the time between the latest output of one generator and the earliest output of another generator, the prediction horizon, is always fixed.

We use the windowing algorithm in two ways. On the one side, we iterate through the

\[ \text{Original Data} \]

\[ \begin{array}{|c|c|} \hline \text{time} & \text{value} \\ \hline 1 & 3 \\ 2 & 15 \\ 3 & 6 \\ 4 & 2 \\ 5 & 5 \\ 6 & 10 \\ \hline \end{array} \]

\[ \text{Transformed data} \]

\[ \begin{array}{|c|c|c|} \hline v_2 & v_1 & t \\ \hline 3 & 15 & 6 \\ 15 & 6 & 2 \\ 6 & 2 & 5 \\ 2 & 5 & 10 \\ \hline \end{array} \]

Figure 5.2.: Illustration of windowing method. For the time \( t \) the previous outputs \( t - 1, t - 2 \) are assigned to the variables \( v_1, v_2 \).
5.1. Pre-processing and feature extraction

(a) Step 1. Identify the window.

(b) Step 2. Remove duplicates.

(c) Step 2. Fill gaps.

(d) Step 3. Output the entries.

Figure 5.3.: Basic steps of the windowing algorithm.

original dataset and create a new dataset with aggregated features like the beginning and the end of a phase, its length, number, the number of vehicle calls, etc. On the other side, we iterate through this aggregated data and produce the final datasets that contains the features described in Table 5.1. For the prediction of the phase length, phase activity, and the next phase it is enough to use the same dataset for “past” and “future”. For the prediction of the traffic flow we need to use the data from the upstream intersection for “past” and the data from the downstream intersection as “future”.

5.1.2. Features

Extracting more condensed features from the raw data allowed us to reduce the size of the problems both in the number of variables and the size of the dataset. Therefore, the extraction of the useful features from the data was the first and one of the most challenging tasks in the project.

After referring the traffic engineering literature [10,42,43], analysis of the dataset, and brainstorming we assembled the list of features that may be extracted from the data. Table 5.1 summarizes the results of the feature engineering process. In the table we show how different types of the same features are used for different prediction tasks in fully-actuated (A) and coordinated (C) modes. We will reference this table when we will speak about individual problems.

Some of the features are polynomial values, for example, the number of the current phase in ring 1 can take values between 1 and 4, but it does not have an established metric, as the distance between 1 and 2 is not smaller than between 1 and 4. While C4.5 decision trees can handle this type of attributes directly (see below), for other machine learning methods we transformed these features into a set of binomial features. For example, the number of the current phase would become a vector of four binary values each responsible for the corresponding phase number.
Table 5.1.: Assignment of features to problems (A—actuated mode, C—coordinated mode, L—phase length prediction, Act—phase activity prediction, N—next phase prediction, F—traffic flow prediction).

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Values</th>
<th>L</th>
<th>Act</th>
<th>N</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>time of the day</td>
<td>N</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td></td>
</tr>
<tr>
<td>phase number in current ring</td>
<td>{1, 2, 3, 4}</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td></td>
</tr>
<tr>
<td>phase number in other ring</td>
<td>{5, 6, 7, 8}</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td></td>
</tr>
<tr>
<td>number of vehicle calls detected on the active phase in current cycle</td>
<td>$\mathbb{Z}_+$</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of vehicle calls detected on the active phase in previous $n$ cycles</td>
<td>$\mathbb{Z}_+^n$</td>
<td>AC</td>
<td>AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of vehicle calls detected on the active phase so far</td>
<td>N</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of vehicle calls detected on the active phase during the prediction horizon $\lambda$</td>
<td>$\mathbb{Z}_+^\lambda$</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>current state of vehicle calls on all 4 phases of the ring</td>
<td>$\mathbb{Z}_2^4$</td>
<td>AC</td>
<td>AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>current state of pedestrian calls on all 4 phases of the ring</td>
<td>$\mathbb{Z}_2^4$</td>
<td>AC</td>
<td>AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phase lengths in previous $n$ cycles</td>
<td>$\mathbb{N}_n$</td>
<td>AC</td>
<td>AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phase length so far</td>
<td>N</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>phase length of the concurrent phase so far</td>
<td>N</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cycle lengths in previous $n$ cycles</td>
<td>$\mathbb{N}_n$</td>
<td>AC</td>
<td>AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cycle length so far</td>
<td>N</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>barrier length</td>
<td>N</td>
<td>C</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>barrier length for the previous $n$ cycles</td>
<td>$\mathbb{N}_n$</td>
<td>C</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>length from the barrier so far</td>
<td>N</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


5.2. Motivation for the choice of machine learning methods

Machine learning techniques allow our system to face the challenges described in Section 1.3. The models incorporate only general nature a prediction problem, like feature set, but the instantiation of a concrete predictor can be simply trained and retrained to fit concrete traffic controllers (requirements to work in heterogeneous environment and adaptability).

As explained in Section 1.4, we do not attempt to conduct a comprehensive comparison of the state-of-the-art machine learning methods, we rather select a small number of methods that would satisfy our functional and nonfunctional requirements.

From the scientific point of view, we were interested in comparison of the methods from different families. Therefore, we choose k-nearest neighbors as a nonparametric method for its simplicity and extendability. We choose neural networks as a parametric method for their compactness, maintainability, and natural implementation of recursive neural networks architecture for the traffic flow prediction problem. We choose decision trees as a nonmetric method for its interpretability and the ability to learn deterministic rules and handle polynomial attributes, which makes it optimally suited for the next phase prediction problem.

5.3. k-nearest neighbors

The \( k \)-nearest neighbors (kNN) algorithm belongs to the nonparametric machine learning techniques for the estimation of probability density functions, classification, and regression. In contrast to the parametric techniques, kNN does not require any assumption about the form of the underlying density function. Despite the simplicity of the method, kNN shows good results for many real-world applications [44]. In fact, it can be shown that for \( k = 1 \) as the number of patterns goes to infinity the error rate of the nearest-neighbor classifier is at most twice the minimum achievable error rate of the optimal classifier, for which the true class distribution is known [45].

Let us assume that we have a dataset with \( N \) patterns consisting of \( N_i \) patterns of class \( C_i \) so that \( \sum_i N_i = N \). Consider a hypersphere with volume \( V \) that contains \( k \) elements from the training dataset centered at the point \( x \) for which we want to estimate the density \( p(x) \). We allow the radius of the hypersphere to grow until it contains precisely \( k \) data points. It is straightforward to show (see [46, p. 122]) that

\[
p(x) = \frac{k}{NV}.
\]  

(5.1)

Suppose that the hypersphere contains \( k_i \) patterns of class \( C_i \). Hence, using (5.1) we can estimate the conditional density of \( x \) as

\[
p(x|C_i) = \frac{k_i}{N_i V},
\]  

(5.2)

and the class priors are given by

\[
p(C_i) = \frac{N_i}{N}.
\]  

(5.3)

We can combine these equations using Bayes’ theorem to obtain the posterior probability of class membership

\[
p(C_i|x) = \frac{p(x|C_i)p(C_i)}{p(x)} = \frac{k_i}{k}.
\]  

(5.4)

The classification using (5.4) can be intuitively interpreted as follows: in order to classify the pattern \( x \) we choose the most frequent class from the \( k \) closest patterns in the training dataset.

\footnote{One can update the k-nearest neighbors predictor by removing old and adding new patterns to the training dataset.}
dataset (majority voting). Sometimes it may be useful to introduce weighting so that the closer patterns contribute more to the calculation of the class than the farther ones (in general case the contribution is uniform and equal to $1/k$).

The extension of the method for regression is straight forward: we calculate the (weighted) average of the target values of the neighboring patterns.

Finally, we would like to note that the estimation of the nearest neighbor strongly depends on the notion of distance. We use mixed Euclidean distance for the calculation: it behaves as the usual Euclidean distance for numerical variables, and for nominal variables it is equal to 1 if the values are different, and 0 if they are equal.

5.4. Prediction with feed-forward neural networks

Artificial neural networks is a parametric computational method used for function approximation. It may be missing the mathematical elegance, but it was theoretically proved that a three layer network can approximate an arbitrary function [47]. Unlike another methods for non-linear statistical data modelling, e.g. SVN, using neural networks often results in a compact and efficient model with same generalization properties [46, pp. 225f]. In contrast to SVN, the energy function minimized by learning of neural networks is not convex so that the restart of the training process is required to avoid local minimums. However, in many cases the longer training time is the reasonable price to pay for a model that can be efficiently evaluated. In this section we consider the most commonly used type of artificial neural networks called feed-forward networks (FFNN).

The fundamental principle of neural networks is to represent the target function as a non-linear combination of basis functions parametrized with regard to the dataset:

$$y(x, w) = f \left( \sum_{i=1}^{M} w_j \phi_j(x) \right),$$

(5.5)

where $w_j \in w$ are the weights of the basis functions $\phi_j(x)$ in the nonlinear activation function $f(\cdot)$.

The function $\phi_j(x)$ itself is also defined as a nonlinear function of a linear combination of inputs:

$$\phi_j(x) = z_j = h(a_j) = h \left( \sum_{i=1}^{D} w_{ji} x_i + w_{j0} \right),$$

(5.6)

with $a_j := \sum_{i=1}^{D} w_{ji} x_i + w_{j0}$

(5.7)

if the function depends directly on the inputs $x_i \in x$, or

$$\phi_j(x) = z_j = h(a_j) = h \left( \sum_{i=1}^{D} w_{ji} z_i + w_{j0} \right),$$

(5.8)

with $a_j := \sum_{i=1}^{D} w_{ji} z_i + w_{j0}$

(5.9)

if the function depends on the values of other activation functions $z_i \in z$. The parameter $w_{j0}$ is called bias, the values $a_j$ are activations. The transfer function $h(\cdot)$ is a logistic function, for example $\text{tanh}$. 

32
For a standard regression problem, as one can find in Chapter 6, the activation function is identity: $y_k = a_k$. For a binary classification problem that occurs in Chapter 7, the activation function is logistic sigmoid:

$$y_k = \sigma(a_k) = \frac{1}{1 + \exp(-a_k)}.$$  \hspace{1cm} (5.11)

For a FFNN with one hidden layer for a binary classification problem the overall network function can therefore be summarized as

$$y(x, w) = \sigma \left( \sum_{i=0}^{M} w_{ji} h \left( \sum_{i=0}^{D} w_{ji} x_i \right) \right).$$ \hspace{1cm} (5.12)

This network is traditionally visualized as an interconnection of nodes as depicted in Figure 5.4.

Figure 5.4.: Feed-forward neural network with one hidden layer that visualizes Equation (5.12).

The training of neural networks is performed by finding the weights vector $w$ that minimizes the sum-of-squares errors (for regression)

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} ||y(x_n, w) - t_n||^2 = \sum_{n=1}^{N} E_n(w).$$ \hspace{1cm} (5.13)

The optimization is performed using gradient descent—an iterative minimization algorithm with an update step

$$w^{\text{new}} = w^{\text{old}} - \eta \nabla E(w^{\text{old}}),$$ \hspace{1cm} (5.14)

where $\eta$ is the learning rate of the algorithm. For the estimation of the gradient $\nabla E(w)$ we use error backpropagation formula [46, Ch. 5]

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_i z_i, \hspace{1cm} (5.15)$$

$$\delta_j = \begin{cases} y_j - t_j, & \text{for output layer} \\ h'(a_j) \sum_k w_{kj} \delta_k, & \text{for hidden layers} \end{cases} \hspace{1cm} (5.16)$$

Algorithm 1 describes the training of FFNN using online backpropagation. Online algorithm means that updates occur after each pattern. Alternatively, one can use the batch method,

where several patterns are processed before each update. For every pattern—a combination of the input value vector \( x_n \) and the target value vector \( t_n \)—we evaluate the output of the network \( y_n \) using forward propagation. In the next step we recursively estimate \( \delta \)'s using equations (5.16). We start with the output layer and move towards the input layer. After the backpropagation is complete, we can estimate the derivatives of the energy function and update the weight parameters. The algorithm repeats these steps until convergence, e.g. until the error is smaller than a threshold \( \varepsilon \).

**Algorithm 1** Training Algorithm for FFNN using Online Backpropagation

```
repeat
    Choose training pattern \((x_n, t_n)\)
    Forward propagate \(x_n\) and find activations of hidden and output units using (5.12)
    Evaluate \( \delta_k = y_n - t_n \) for the output units
    Evaluate \( \delta \)'s for hidden units using backpropagation formula (5.16)
    Calculate the required derivatives \( \nabla E(w) \) using (5.15)
    Update weights using (5.14)
until \( E(w) > \varepsilon \)
```

Working with large neural networks, it is convenient to represent the layers not as a set of individual neurons but as a cluster. This effects the graphical representation of the networks as well as the mathematical formulation, since the connectivity weights between individual neurons are replaced by the weight matrices. The neural network in Figure 5.4 can be represented by three clusters \( x, y, \) and \( z \) of input, hidden, and output neurons respectively as shown in Figure 5.5.

![Cluster representation of an FFNN](image)

Figure 5.5.: Cluster representation of an FFNN. Individual neurons are grouped into clusters, matrices \( A \) and \( B \) contain the weights of individual interconnections.

The vector of activations \( z \) can be written in matrix-vector form using Equation (5.17):

\[
z = h(Ax).
\]

(5.17)

In the same way, we can use Equation (5.12) to estimate the output values:

\[
y = \sigma(Bz).
\]

(5.18)

We adopt the cluster representation and matrix-vector notation of neural network equations in the remain of the thesis.

### 5.4.1. Spatio-temporal unfolding with neural networks

We can look on the generating function of a time series as on a dynamical system with an observable output and an unobservable inner state. Figure 5.6 shows a dynamical system in a discrete time: at the moment \( t \) the system transfers to the new state \( s \) and produces the output \( y \) depending on the current input \( x \) and the previous inner state at time \( t - 1 \). This state transition is defined by the state equation

\[
s_t = \sigma(x_t, s_{t-1})
\]

(5.19)

and the produced output \( y_t \) is defined by the output equation

\[
y_t = \phi(s_t).
\]

(5.20)
5.4. Prediction with feed-forward neural networks

There is a number of methods to tackle the problem in this representation, including input-output Markov chains [48] or recurrent neural networks (RNN) [49]. In this section we focus on RNN as they represent the natural development of FFNN and offer the model for spatio-temporal unfolding used for traffic flow prediction in Chapter 9.

Unlike FFNN, where all connections between neurons are directed from lower to upper layer, RNN also allows backward connections. The relationship between $s_t$ and $s_{t-1}$ can be represented in form of time-delayed connection. Time delay means that the output of the source neuron is affecting the connected destination neuron only in the next time step.

Using matrix-vector form introduced in the previous section, we can write the state transition Equation (5.19) and the output Equation (5.20) as

$$s_t = \sigma(As_{t-1} + Bx_t), \text{ and}$$

$$y_t = Cs_t. \tag{5.21}$$

Figure 5.7 shows the corresponding neural network architecture. The weight matrix $A$ encodes the autonomous part of the dynamical system, while $B$ represents the external influences, and $C$ the transformation of the inner state to the output.

Time-delay recurrent neural networks we use were implemented using the technique called finite unfolding in time with shared weights matrices $A$, $B$, and $C$ [49]. Because the weights share the same space in memory, their values are the same at each time step of the unfolding.

In the example network shown in Figure 5.8 the unfolding is truncated after $t - 3$. For the discussion of the optimal windows size for the finite unfolding we refer to the paper by Zimmermann et al [49].

Previously, we assumed that the observable output of the dynamical system $y_t$ depends on the input $u_t$ in the same time interval. This can be easily generalized for an arbitrary lag $l$ between observables. The equations (5.21) and (5.22) will take then the form

$$s_{t-l} = \sigma(As_{t-l-1} + Bx_{t-l}), \tag{5.23}$$

$$y_t = Cs_{t-l}. \tag{5.24}$$

The finite unfolding of the corresponding RNN is depicted on Figure 5.9. Besides the lag $l$, we use the window size $k$. 

Figure 5.6.: Schematic representation of a dynamical system, where the output $y_t \in \mathbb{R}^n$ is determined by the input $x_t \in \mathbb{R}^k$ and the hidden state $s_t \in \mathbb{R}^d$. 

Figure 5.7.: Time-delay recurrent neural network for a dynamical system in Figure 5.6. The output of the source neuron of a time-delayed connection is affecting the destination neuron only in the next time step.

Figure 5.8.: Finite unfolding in time using shared weight matrices $A, B,$ and $C$. Each group $(x_i, z_i, y_i)$ corresponds to one step in time unfolding. Unfolding is truncated after $t - 3$. 
5.5. Decision trees

Decision trees (DT) create a sequence of rules that leads to a certain class, when applied to the attribute values of a pattern. Formally, classification and regression trees (CART) \([50]\) is a method of machine learning that solves classification problem by partitioning the parameter space into cuboids with the same target value. The edges of the cuboids coincide with the axes of the parameter space.

Figure 5.10 illustrates the principle of decision trees: a dataset is partitioned into decision areas by borders that coincide with the parameter axis and different classes are assigned to the different areas (Figure 5.10(a)). Figure 5.10(b) shows the corresponding decision tree. The inner nodes of the tree contain the comparison rules. The leafs contain the class assignments.

Figure 5.10.: Dataset partitioning and corresponding decision tree. The queries in the nodes of the decision tree partition the dataset into decision areas by borders that coincide with the parameter axis and different classes are assigned to the different areas.

From this scheme one can easily see the advantage of decision trees compared, for example,

to neural networks: the resulting models are easy to read and interpret. Besides, they are
good suited for prediction of the traffic controller algorithms, since, as described previously,
the logic of the control strategies is strictly deterministic and defined by the known finite set
of internal and external parameters.

Decision trees are a nonmetric technique, meaning that establishing the distance between at-
tribute values is not necessary. Therefore, this technique is commonly used to handle nominal
attributes that can take one of the several distinctive values in a set, e.g. phase number.

Algorithm 2 Decision tree learning [51, p. 537]

1: function \textsc{DecisionTreeLearning}(examples, attributes, default)
2: \hspace{1em} if \text{examples} is empty then return \text{default}
3: \hspace{1em} else if all \text{examples} have the same class then return the class
4: \hspace{1em} else if stopping criteria fulfilled then return MAJORITYVALUE(examples)
5: \hspace{1em} else
6: \hspace{2em} best ← \textsc{ChooseAttribute}(attributes, examples)
7: \hspace{2em} if best is a discrete finite set then
8: \hspace{3em} tree ← a new decision tree with root test best
9: \hspace{3em} for all \( v_i \) value of best do
10: \hspace{4em} \text{examples}_i ← \{elements of \text{examples} with best = \( v_i \)\}
11: \hspace{4em} \text{default} ← MAJORITYVOTE(examples)
12: \hspace{4em} subtree ← \textsc{DecisionTreeLearning}(\text{examples}_i, attributes, default)
13: \hspace{4em} add a branch to \text{tree} with label \( v_i \) and subtree \text{subtree}
14: \hspace{2em} end for
15: \hspace{1em} else if best is numerical then
16: \hspace{2em} \( v \) ← \textsc{ChooseThreshold}(best, examples)
17: \hspace{2em} tree ← a new decision tree with root test best and threshold \( v \)
18: \hspace{2em} \text{default} ← MAJORITYVALUE(examples)
19: \hspace{2em} \text{examples}_\leq ← \{elements of \text{examples} with best \leq v\}
20: \hspace{2em} \text{examples}_> ← \{elements of \text{examples} with best > v\}
21: \hspace{2em} add branches \textsc{DecisionTreeLearning} (\text{examples}_\leq, attributes, default) and \textsc{DecisionTreeLearning} (\text{examples}_>, attributes, default) to the tree
22: \hspace{1em} end if
23: \hspace{1em} end if
24: return \text{tree}
25: end function

In order to learn the model, we have to establish the structure of the tree, the attributes that
are used at every node, and the thresholds for the split. The general steps of building a decision
tree are show in Algorithm 2. The function \textsc{DecisionTreeLearning} receives a set of training
\text{examples}, a set of \text{attributes}, as well as the \text{default} value of the target class. The lines 2 to 4 cover
the cases when the current node is a leaf. If the learning algorithm needs to create a new deci-
sion node, it chooses the best suitable attribute according to the \textsc{ChooseAttribute} method,
splits the examples set by the values of the \text{best} attribute and recursively creates the branches
of the tree by calling \textsc{DecisionTreeLearning} using only subsets of examples \text{examples}_i,
where the attribute \text{best} has a particular value \( v_i \). Alternatively, the algorithm creates only two
subsets \text{examples}_\leq and \text{examples}_>, where the \text{best} attribute values are not greater or greater
than the threshold \( v_i \) if \text{best} is a numerical attribute (lines 15 to 22).

The functions \textsc{ChooseAttribute} and \textsc{ChooseThreshold} are essential for the quality of
the resulting decision tree. We want to find the combination of an attribute and a threshold for
splitting that minimizes the \textit{impurity} in the partitioned example subsets with respect to a cer-
tain \textit{information criterion}. The impurity describes the information complexity of the examples
in the node. We want the impurity \( i(N) \) of a node \( N \) to be 0, if all patterns that bear \( N \) are in the same class, and to be large if different classes are equally represented. Building a node in the decision tree we choose a query that reduces the impurity as much as possible. The impurity change is defined as

\[
\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R),
\]

(5.25)

where \( N_L \) and \( N_R \) denote the left and the right descendent nodes respectively, \( i(N_L) \) and \( i(N_R) \) are the corresponding impurities, and \( P_L \) is the fraction of examples at \( N \) that will transfer to \( N_L \).

The following information criteria are used in practice (\( P(C_j) \) denotes the fraction of patterns at \( N \) that belongs to class \( C_i \)) [52, Ch. 8]:

- **Entropy**

\[
i(N) = - \sum_j P(C_j) \log_2 P(C_j),
\]

(5.26)

- **Gini index**

\[
i(N) = \sum_{i \neq j} P(C_i) P(C_j) = \frac{1}{2} \left( 1 - \sum_j P^2(C_j) \right),
\]

(5.27)

- **Misclassification**

\[i(N) = 1 - \max_j P(C_j), \]

(5.28)

- **Gain ratio** with impurity change defined as

\[
\Delta i_B(s) = \frac{i(N) - \sum_{k=1}^B P_k \cdot i(N_k)}{- \sum_{k=1}^B P_k \log_2 P_k},
\]

(5.29)

where \( B \) is branching factor.

Stopping criteria is another important factor in building decision trees. There are some obvious criteria, for example, the minimal number of examples that can be splitted or the maximal size of a tree. An alternative stopping criterion is based on the statistical significance of the impurity reduction. Let us consider a split \( s \) that splits \( n \) examples of two classes: \( n_1 \) examples of the class \( C_1 \) and \( n_2 \) examples of the class \( C_2 \). Our null hypothesis states that the split \( s \) is random. If this would be the case, and it splits the data into \( Pn \) and \( (1 - P)n \) subsets, then we would expect to find \( Pn_1 \) examples of the class \( C_1 \) and \( Pn_2 \) examples of the class \( C_2 \) in the first (left) subset. No we apply chi-squared test [53] to calculate the distance between the split \( s \) and the corresponding weighted random split:

\[
\chi^2 = \frac{\sum_{i=1}^2 (n_{IL} - n_{IE})^2}{n_{IE}},
\]

(5.30)

with \( n_{IL} \) the number of examples in the class \( C_i \) sent to the left subtree after decision \( s \), and \( n_{IE} := Pn_i \) the expected number of examples in the subtree after a random split. The value of \( \chi^2 \) is very small if \( s \) behaves similar to the random split, and it is large if \( s \) significantly differs.

We can reject the null hypothesis with a certain confidence level if \( \chi^2 \) value is greater than the corresponding critical value (see examples in Table 5.2). If at a certain node the \( \chi^2 \) value of the “best” split does not exceed the threshold for the chosen confidence level, the splitting should be stopped.
Table 5.2.: The examples of confidence levels and corresponding $\chi^2$ critical values for two different degrees of freedom (df).

<table>
<thead>
<tr>
<th>df</th>
<th>Confidence interval</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.05</td>
</tr>
<tr>
<td>2</td>
<td>5.99</td>
</tr>
<tr>
<td>4</td>
<td>9.49</td>
</tr>
</tbody>
</table>

In order to improve the generalization properties of the tree, usually, a large tree is created and then reduced to a smaller one. This procedure is called pruning. The decision about merging of the subtrees induced by the node $N$ is made using the pruning criterion

$$C(N) = \sum_{\tau=1}^{|N|} i(N_{\tau}) + \lambda |N|,$$

(5.31)

where $N_{\tau}$ are the descendant leaf nodes, $|N|$ the number of leaf nodes and $\lambda$ the regularization parameter that determines the trade-off between the impurity of the node and the complexity of the model measured by $|N|$.

The decision tree learning algorithm used in this thesis, C4.5, extends the original CART in several different ways [54]. It is able to handle real-valued variables as CART and the polynomial variables ($B > 2$) using gain ratio impurity (5.29), and it supports pruning procedure.

5.6. Ensemble methods

We apply re-sampling methods that proved to work effectively with machine learning techniques, like neural networks and decision trees. Bagging (short for “bootstrap aggregation”) and boosting aim to increase the diversity in a set of model instances, called ensemble, by training (and retraining) them on different subsets of data. The output of an ensemble is then calculated as an average of the outputs of individual models (for regression) or as a majority vote (for classification). It can be shown that increasing diversity in an ensemble, while the average error of the individual models remains low, leads to the decreasing of the ensemble generalization error [55].

In the bagging method a set of models is trained on different datasets re-sampled from the original training data [56]. The fixed sample rate gives the probability for each pattern to be sampled, and every pattern can be sampled several times. Since the training on different datasets can be done simultaneously, we use bagging in combination with neural networks, which generally take longer time to train than decision trees. Besides increasing the diversity in an ensemble, bagging helps neural networks to avoid local minima, as networks are retrained several times.

Boosting produces a series of predictors in a number of stages [57–59]. The predictors from each following stage are trained on the bootstrapped re-sampling of the training set according to the performance of the predictors in the previous stage. Patterns that were incorrectly predicted in the previous stage gain more importance in the following. The empirical evaluations has shown the advantage of training C4.5 decision trees with boosting compared to the training of the decision trees alone [60]. We use the boosted decision trees for our phase activity classification problem in Chapter 7.
6. Phase Length Prediction

Each of the next four chapters puts forward a formal specification of the problem of interest, description of the prediction model based on paradigms from Chapter 5 and an evaluation of the approach on the test dataset described in Chapter 4.

The first problem is phase length prediction. We estimate the total time a phase will stay active based on the information available at the moment the traffic controller switched the phase. Formally, the function that generates the data $f : \mathcal{X} \times \mathcal{I} \to \mathbb{N}$ transforms the input from the Cartesian product of the space of the external $\mathcal{X}$ and internal $\mathcal{I}$ parameters in a natural number of the phase length duration.

Given the dataset $D = \{(x_i, y_i) | x_i \in \mathcal{X} \times \mathcal{I}, y_i \in \mathbb{N}, i \in \mathbb{N}\}$ of ordered inputs $x_i$ and outputs $y_i$ of a controller, we want to estimate the function $\tilde{f}_p$ that most correctly approximates the time function $f$. A general way to do this is by minimizing the distance between the true and the approximated outputs over the parameter space $P$:

$$\tilde{f}_p = \arg\min_{p \in P} ||f_p(x_i) - y_i||, \forall (x_i, y_i) \in D.$$ (6.1)

Depending on the machine learning method we choose, the parameter space and minimization methods can vary as described in Chapter 5.

6.1. Model

At the moment of prediction, our input space $\mathcal{X} \times \mathcal{I}$ needs to contain the information about the traffic movement during the phase. Unfortunately, we cannot have this information before the phase begins. Instead, we choose a set of features that we know and that, so our assumption, will correlate with the target or with the unknown state parameters.

We observed that the phase length is a discrete function with low variation. Figure 6.1 shows the durations of the phases 2 and 3 in consecutive cycles. Even so, these values are subject to variation, the lengths do not vary arbitrarily.

As shown in Section 2.3, the phase length depends on the immediate traffic situation on the active and conflicting lanes. However, as the generating function we observe has low variation, we can make an assumption that the previous lengths of the same phase serve as an important input for the approximator. For prediction of phase length in coordinated mode, where cycle length is fixed and the phases have to stop behind the barriers, the information about previous cycle lengths and barrier lengths can serve as a valuable input as well.

As summarized in Table 5.1, we decided to use 19 features for actuated mode

- time of the day,
- phase number in current ring,
- phase number in other ring,
- number of vehicle calls (VC) detected on the active phase in current cycle,

\[ \text{The phase activity is defined as green, yellow and all-red times of a phase. If we are interested in green time only, we can easily subtract yellow and all red times in the post-processing step, since they are always constant.} \]
Figure 6.1.: Length of the phases 2 and 3 at different times of the day. Although, the lengths of the phases vary in different cycles, this variation is comparably small.

- number of vehicle calls detected on the active phase in previous 5 cycles,
- phase lengths in previous 5 cycles,
- cycle lengths in previous 5 cycles.

For coordinated mode we used only one length of the previous cycle as the cycle length is fixed, and we add 6 features:

- barrier length so far,
- barrier lengths in previous 5 cycles.

The target variable is the “phase length” of the currently active phase.

### 6.2. Evaluation

We evaluate the machine learning algorithms on the data from the intersection Mowry Ave and Blacow Rd. The other intersections behave very similarly. The dataset for coordinated mode contains 11 368 entries and consists of 21 features and 1 target, while for actuated mode it contains 1 996 entries and consists of 19 features and 1 target. For actuated mode we considered the patterns with phase length greater than 100 seconds as outliers and filtered them out. Most of the cases with longer phases occurred deep in the night, so that we can justify leaving them out.

For the purpose of this evaluation, we compared kNN and two FFNN architectures—with one and two hidden layers. The estimation of the training parameters was carried out using 5-fold cross validation on 70% of the data used for training. Then the models were retrained on the complete training set using optimal parameters and the generalization error was evaluated using the remaining 30% of the dataset.

In order to eliminate the effect of neural networks falling into local minima, we performed bagging with 10 iterations and sample ratio of 0.9. The neural networks were trained using
Table 6.1.: Phase length prediction results. Best results are highlighted bold. “Configuration” field describes the optimal configuration of the models: k for kNN and the number of neurons in a hidden layer for neural networks. RMSE denotes root means squared error, ABS denotes average absolute error. FFNN with one hidden layer outperform all other methods.

<table>
<thead>
<tr>
<th>Mode</th>
<th>VC available</th>
<th>Method</th>
<th>Configuration</th>
<th>Generaliz error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RMSE [s]</td>
</tr>
<tr>
<td>Coord.</td>
<td></td>
<td>Baseline</td>
<td></td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>with VC</td>
<td>FFNN 1 hid. layer</td>
<td>hid. neur. = 4</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kNN</td>
<td>k=10</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>with VC</td>
<td>FFNN 1 hid. layer</td>
<td>hid. neur. = 6</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kNN</td>
<td>k=10</td>
<td>3.7</td>
</tr>
<tr>
<td></td>
<td>Actuated</td>
<td>FFNN 2 hid. layers</td>
<td>hid. neur. = 4</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>with VC</td>
<td>FFNN 2 hid. layers</td>
<td>hid. neur. = 4</td>
<td>10.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Baseline</td>
<td></td>
<td>17.5</td>
</tr>
<tr>
<td></td>
<td>without VC</td>
<td>FFNN 1 hid. layer</td>
<td>hid. neur. = 2</td>
<td>13.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kNN</td>
<td>k=30</td>
<td>14.0</td>
</tr>
<tr>
<td></td>
<td>with VC</td>
<td>FFNN 2 hid. layers</td>
<td>hid. neur. = 4</td>
<td>14.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kNN</td>
<td>k=10</td>
<td>13.2</td>
</tr>
</tbody>
</table>

learn rate of 0.1 and weight decay for 2000 cycles with inputs normalized on the interval between -1 and 1.

In Table 6.1 we present the results of retrained models with optimal parameters in terms of root mean square error (RMSE) and average absolute error (ABS). We compare the models trained in the same experimental setup with and without vehicle call information to investigate the impact of the traffic flow on prediction quality. For comparison we also present the baseline estimation, that is the error achieved using the average of the previous 5 phase lengths of the same phase as the prediction.

All models show better performance than the baseline. FFNN with one hidden layer indicate the best performance for both coordinated and actuated modes. FFNN with two hidden layers show moderate results, even though the model should be more powerful. This observation may be explained by the fact that they need more data or more time to train.

Table 6.1 also show the importance of traffic information. While without VC information the machine learning methods improved the baseline by less than 23%, with VC information this improvement was over 57% for coordinated mode and 38% for actuated mode. This results contradict the observations of Koukoumidis et al. that the traffic information does not have any considerable impact on the prediction quality.

In coordinated mode without traffic information, only FFNN with 1 hidden layer could show the accuracy of less than 5 seconds, although, the prediction accuracy of kNN is also acceptable. With vehicle calls, both FFNN and kNN showed sufficient accuracy for GLOSA and even for other applications with higher requirements.

For actuated mode, neither of the models could come close to the accuracy requirements of the applications. But as the baseline estimation indicates, the data itself was the subject to very

---

2The purpose of this evaluation is to show the general principle of importance of the features and to give an impression, in what order of magnitude the results can be expected, rather than what is the best possible result that can be achieved by performing exhaustive parameter optimization to fit a concrete dataset.
high variability even after filtering the long phases.

Using our feature extraction methods, we could build a nonlinear prediction model that shows satisfactory results using the historical data only. Adding the traffic information allows further improvement of the results. Unfortunately, the prediction with high accuracy is not possible in general, as even with vehicle call information, the prediction accuracy in actuated mode does not satisfy the requirements of the applications.
7. Phase Activity Prediction

In the previous chapter we considered methods for phase length prediction. However, for some applications, i.e. RLVA and RTCS, we are not interested in the length of the phase. In this chapter the problem is whether or not the phase will end within the next $\delta$ seconds. For example, $\delta = 1$ for RTCS and $\delta = 5$ for RLVA. We can formulate this problem as a binary classification: given the data, predict whether or not the currently active phase remains active in the following time interval.

The generating function $f: \mathcal{X} \times \mathcal{I} \rightarrow \{0, 1\}$ transforms the input from the Cartesian product of the space of the external $\mathcal{X}$ and internal $\mathcal{I}$ parameters to the binary values: 1 if the phase stays active and 0 otherwise.

Given the dataset $D = \{(x_i, y_i) | x_i \in \mathcal{X} \times \mathcal{I}, y_i \in \mathbb{N}, i \in \mathbb{N}\}$ of ordered inputs $x_i$ and outputs $y_i$ of the controller, we want to estimate the function $\tilde{f}_p$ that most correctly approximates the generating function $f$. The general way to do it is by solving the following maximization problem over the parameter space $\mathcal{P}$:

$$\tilde{f}_p = \arg \max_{p \in \mathcal{P}} \prod_{(x_i, y_i) \in D} f_p(x_i)^{y_i} \cdot (1 - f_p(x_i))^{(1 - y_i)}.$$  \hspace{1cm} (7.1)

We can easily generalize this problem by introducing the prediction horizon $\lambda$ and the prediction interval $\delta$ as illustrated in Figure 7.1. Given all information at the time point $t$, the goal is to predict if the phase stays active for more than $\delta/2$ seconds in the interval $[t + \lambda, t + \lambda + \delta]$.

The intuition is that $\delta$ corresponds to a “time resolution” capability of a predictor, while $\lambda$ corresponds to prediction horizon. We expect that, on the one side, the higher the horizon—the lower the accuracy. And, on the other side, the lower the “time resolution”, and so the larger the interval $\delta$—the higher the accuracy. Therefore, we can control the quality of the prediction by adjusting $\delta$ for the fixed value of $\lambda$ and vice versa. In retrospect on the application requirements in Section 1.2, $\delta$ corresponds to the maximal allowed prediction error called “accuracy” in Table 1.2.

Obviously, $\lambda = 0$ and $\delta = 1$ represent precisely the case described in the beginning of the section, but this formalization has much more power. For example, we can transform the predictions to estimation of the phase length with desirable accuracy by making the prediction for different values of $\lambda$ and then applying a threshold as shown in Figure 7.2.
7. Phase Activity Prediction

Figure 7.2: Phase length prediction using classification. The maximal prediction horizon $\lambda$ with probability of phase activity greater than the threshold defines the predicted phase length.

7.1. Model

All types of features we discussed in Section 5.1.2 are important for the prediction of the phase activity. As summarized in Table 5.1, we decided to use 31 features for fully-actuated mode:

- time of the day
- phase number in current ring,
- phase number in other ring,
- number of vehicle calls (VC) detected on the active phase in previous 5 cycles,
- number of vehicle calls detected on the active phase so far
- number of vehicle calls detected on the active phase during the prediction horizon $\lambda$,
- phase length so far,
- phase length of the concurrent phase so far,
- phase lengths in previous 5 cycles,
- current state of vehicle calls on all 4 phases of the ring,
- current state of pedestrian calls on all 4 phases of the ring,
- cycle length so far,
- cycle lengths of the previous 5 cycles.

For predictions in coordinated mode we used only the cycle length of the previous cycles instead of the previous 5 cycle and 6 additional features:

- barrier length so far,
- barrier lengths in previous 5 cycles,

The features “phase length so far”, “phase length of the concurrent phase so far”, “barrier length so far”, and “cycle length so far” can be additionally used for this kind of prediction since at the moment of the prediction we already have an additional information about the current phase. This was impossible in the prediction of the phase length in the previous chapter.
7.2. Evaluation

Discussing applications and requirements in Section 1.2, we discovered three clusters of applications that require different levels of magnitude for prediction horizon as well as significantly different requirements to accuracy (see Table 1.2). While RLVA, RTCS, and TSPT have the prediction horizon of less than 10 seconds and accuracy of less than 3 seconds, the applications GLOSA, RLDA, and IRLA have the horizon between 10 and 20 seconds and accuracy of 5 seconds. Finally, the application TSAN has the time horizon of 115 seconds and even lower accuracy.

In order to cover different cases of application requirements, but still hold our evaluation general enough to allow the usage of the methods for different applications, we conducted the experiments for two different scenarios: In the first scenario, we investigated the ability of the models to predict the data for a short time horizon ($\lambda \in \{1, 2, \ldots, 10\}$) and high time resolution ($\delta \in \{1, 2\}$). In the second scenario, we covered the higher prediction horizon ($\lambda \in \{10, 20, \ldots, 100\}$) with fixed time resolution $\delta = 5$.

The dataset with measurements in the coordinated mode contains 261,841 patterns with 32 features. For the actuated mode the dataset is smaller as the number of gaps that were filtered out is higher and the phases are generally longer. It contains 77,670 patterns with 31 features.

For this classification task, we use the FFNN and DT paradigms. Since we have a very large dataset, kNN method was intractable as it requires the storage of the complete training dataset in memory.

In a number of trials, we established the optimal training parameters. In the evaluation, we compared FFNN with one hidden layer of 30 neurons, trained for 2,000 epochs with $\eta = 0.05$ and 5 iterations of bagging for local minimums avoidance, and DT with information gain as impurity criterion, maximal depth 40, confidence level 0.25, and boosting with 5 iterations. For coordinated mode we used 40% of the data for training and 60% for the evaluating of the generalization error. For actuated mode we used 70% for training and 30% for generalization since we have much more data in coordinated mode than in actuated.

First, we discuss the results for the combinations of prediction horizon $\lambda$ and accuracy $\delta$ required by our applications. Table 7.1 presents the classification accuracy of feed-forward neural networks and decision trees for the applications RTCS, RLVA, RLDA, and GLOSA. For comparison we present a baseline always predicting the most frequent class in the training dataset independent from the input. The results show that the prediction accuracy of about or even over 98% is achievable. Both machine learning methods perform significantly better than the baseline, however, FFNN outperformed DT by several percent points in the three of four applications.

In Table 7.1, the performance in coordinated and actuated mode was comparable, which can be explained by availability of the vehicle call information. As shown in Figure 7.3 this information plays an important role for the quality of prediction. Figure 7.3 shows the accuracy of the FFNN models with and without VC information for fixed $\lambda = 10$ and different values of $\delta$ in coordinated and actuated mode. With VC information, the difference between the prediction accuracy in coordinated and actuated mode is marginal. Without VC the accuracy for coordinated mode drops by just 2.3%–3.3%, while in actuated mode the decrement is between 6.5% for $\delta = 1$ and 8.5% for $\delta = 2$ and $\delta = 3$. In the last case, the accuracy decreases to only 90% (though, it is still about 30% better than the baseline).

Figures 7.4 and 7.5 illustrate the change of the accuracy with increasing prediction horizon for neural networks and decision trees with fixed $\delta = 1$ and $\delta = 2$. On Figure 7.5 one can clearly see that accuracy decreases with the growing prediction horizon. The experiments with the larger values of $\delta$ show the higher accuracy, as especially good presented in Figure 7.4 for all modes and in Figure 7.5 for coordinated mode; though, these differences in accuracy are often just marginal (about 0.5% for coordinated mode and 2% for actuated mode). This
Table 7.1.: Phase activity prediction results for selected applications (C—coordinated, A—actuated). Best results are highlighted bold. FFNN show the best performance.

<table>
<thead>
<tr>
<th>Application</th>
<th>λ</th>
<th>δ</th>
<th>Mode</th>
<th>FFNN</th>
<th>DT</th>
<th>Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCS</td>
<td>1</td>
<td>1</td>
<td>C</td>
<td>97.55%</td>
<td>96.89%</td>
<td>95.80%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>98.36%</td>
<td>97.67%</td>
<td>96.61%</td>
</tr>
<tr>
<td>RLVA</td>
<td>5</td>
<td>2</td>
<td>C</td>
<td>98.27%</td>
<td>98.03%</td>
<td>79.29%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>97.91%</td>
<td>93.66%</td>
<td>83.31%</td>
</tr>
<tr>
<td>RLDA</td>
<td>10</td>
<td>5</td>
<td>C</td>
<td>98.00%</td>
<td>95.43%</td>
<td>61.13%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>97.93%</td>
<td>90.46%</td>
<td>68.15%</td>
</tr>
<tr>
<td>GLOSA</td>
<td>20</td>
<td>5</td>
<td>C</td>
<td>98.80%</td>
<td>97.12%</td>
<td>57.54%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>98.81%</td>
<td>97.04%</td>
<td>51.09%</td>
</tr>
</tbody>
</table>

Figure 7.3.: Comparison of the prediction accuracy for different δ for λ = 10 for controller in coordinated and actuated modes. With VC information, the difference between the prediction accuracy in coordinated and actuated mode is marginal. Without VC the accuracy for coordinated mode drops by just 2.3%–3.3%, while in actuated mode the decrement is between 6.5% for δ = 1 and 8.5% for δ = 2 and δ = 3.
can be explained by the fact that the results are already close to optimal. An additional effect of increased $\delta$ value is stable high accuracy in coordinated and actuated mode for neural networks. The accuracy of FFNN with $\delta = 1$ drops with $\lambda > 0$ and the gap between the accuracy of predictions in coordinated and actuated modes becomes larger, while for $\delta = 2$ there is no difference.

The output of the predictors also gives us the confidence of the prediction. Figure 7.6 illustrates one of the predictions for neural networks with $\delta = 1$ and prediction horizon of up to 10 seconds. The evaluation takes place at the 26th second of a phase with the total length of 31 seconds. One can see that after the seconds 6 and 7 the confidence of the predictor is close to the decision boundary 0.5. We can give this an interpretation that, for $\lambda \in \{6, 7\}$, the predictor is uncertain about the outcome.

![Graph showing prediction accuracy](image-url)

Figure 7.4.: Prediction accuracy using neural networks for $\delta = 1$ and $\delta = 2$. For $\delta = 2$ the accuracy difference between predictions in coordinated and actuated modes is non-existent, while for $\delta = 1$ this difference is larger.

Almost all ITS applications can profit from the phase activity prediction that allows to integrate the available information about the current phase as well as to control the horizon-accuracy trade. We have shown that the predictions for RTCS, RLVA, RLDA, and GLOSA have the accuracy of about 98% for both coordinated-actuated and fully-actuated modes. With future traffic flow information available, the prediction in fully-actuated mode is as accurate as in coordinated-actuated mode, in contrast to the results of the phase length prediction. Without traffic information, the accuracy drops to about 90%.
7. Phase Activity Prediction

Figure 7.5.: Prediction accuracy using decision trees for $\delta = 1$ and $\delta = 2$. The prediction accuracy decreases with growing prediction horizon.

Figure 7.6.: Phase activity prediction confidence. Phase 4: total phase length 31 seconds, evaluation at the second 26. Confidence values close to 0.5 indicate uncertainty of the prediction.
8. Next Phase Prediction

Another classification task is the prediction of the phase the traffic controller will choose next. But, unlike the prediction of phase activity in Chapter 7, here we have more than two classes to distinguish between. The next phase prediction problem is an instance of multi-label classification. It is not simple to formulate directly as a minimization problem. Instead, it is customary to transform a multi-label classification problem into a number of binary classification problems, which have to be solved simultaneously. For example, for prediction of the four phases in one ring, we would need to solve the following problem:

\[
\max_{p \in P} \prod_{(x_i, y_i) \in D} \prod_{k=1}^{4} f_{kp}(x_i) \delta_k(y_i) \cdot (1 - f_{kp}(x_i))^{1 - \delta_k(y_i)}
\] (8.1)

subject to

\[
\sum_{k=1}^{4} f_{kp}(x_i) = 1, \quad \forall i,
\] (8.2)

\[
f_{kp}(x_i) \in \{0, 1\}, \quad \forall k, i,
\] (8.3)

\[
\delta_k(y_i) = \begin{cases} 
1, & \text{if } y_i = k \\
0, & \text{otherwise}
\end{cases}
\] (8.4)

Several machine learning paradigms, e.g. decision trees, are able to handle polynomial classification directly without this transformation as we described in Section 5.5.

8.1. Model

We know very well what factors impact the choice of the next active phase in the ring. At the moment of phase change, the traffic controller chooses the next phase in the sequence with a registered call: vehicle call, pedestrian call, or recall. The sequence of the phases, nonetheless, can vary depending on the controller configuration for different times of the day. Hence, the application of machine learning methods for multi-label classification is justified and required.

The decision algorithm of the controller does not differ in actuated and coordinated modes, therefore, we choose the same set of features for both cases as shown in Table 5.1:

- time of the day,
- phase number in current ring,
- current state of vehicle calls on all 4 phases of the ring,
- current state of pedestrian calls on all 4 phases of the ring.

We decided to use decision trees because, in our opinion, they are most suitable for the task. Unlike the neural network learning and classification paradigm that is more suitable for approximation of processes that occurs in nature and are characterized by smoothness, C4.5 decision trees can learn strict rules for multi-label classification.
8. Next Phase Prediction

8.2. Evaluation

As it can be seen from the theory, the controller chooses the next phase according to the internal rules parametrized by the current state. Therefore, we decided to use decision trees as the prediction model. Depending on parametrization, DT can produce very different quality of the results. As shown in this section, using parameter search we could achieve the accuracy of more then 99% on the test dataset.

We selected the phase changes for one intersection on workdays and performed cross-validation on 70% of the data. The remaining 30% were used for the estimation of the generalization error. In the cross-validation DT showed the best performance if “gain ratio” was used as impurity criterion, the time resolution (time interval) of 60 seconds, maximal tree depth 40, and the confidence level 0.05. The low time resolution helps to improve generalization properties of the tree, the depth 40 stops trees from getting unnecessary complex (Occam’s razor), and the small confidence level 0.05 makes our model to fit better the training data as we know that we do not have too much noise in the measurements.

We compare four different configurations of the experiment: In the most constrained settings we predict the phase change in coordinated mode and actuated mode on workdays only. Additionally, we investigate the performance of DT on a combined dataset with operation in coordinated and actuated modes on workdays only as well as on workdays and weekends.

Table 8.1 summarizes the results of the experiments. Besides the accuracy of the classification of the generalization dataset we give the baseline obtained using a simple rule: predict the phase that followed the current phase in the previous cycle. Depending on the configuration, the decision trees showed the accuracy between 95% and 100%.

DT showed the best accuracy for the next phase in coordinated mode on workdays. This is also the case we are most interested in. For the reasons discussed below, the accuracy is not 100%, and the accuracy for prediction in actuated mode is only 95.3%. The last two cases show the accuracy between these boundaries. The results show that all-day prediction on workdays is possible with about 98% accuracy, and the all-week prediction is possible with accuracy of about 96%.

Table 8.1.: Next phase prediction accuracy of the decision trees compared to the baseline. Best results are highlighted bold. Prediction for coordinated mode in workdays is possible with almost 100% accuracy.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Prediction accuracy</th>
<th>Days</th>
<th>DT</th>
<th>Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>coordinated</td>
<td>workdays</td>
<td>99.37%</td>
<td>93.41%</td>
<td></td>
</tr>
<tr>
<td>actuated</td>
<td>workdays</td>
<td>95.33%</td>
<td>73.95%</td>
<td></td>
</tr>
<tr>
<td>coordinated+actuated</td>
<td>workdays</td>
<td>97.89%</td>
<td>80.41%</td>
<td></td>
</tr>
<tr>
<td>coordinated+actuated</td>
<td>workdays+weekends</td>
<td>95.77%</td>
<td>78.80%</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8.1 shows the number of misclassifications depending on time for the total observation period. One block corresponds to one hour of observations between 12:00 a.m. (first block) and 11:59 p.m. (last block). The misclassifications have two main reasons. The two largest pikes on the histogram correspond to the times around 540 minutes, or 9:00 a.m., and 1100 minutes, or 6:30 a.m. These are the times when the controller switches from the am peak dial to the normal dial and from the pm peak dial to the normal dial. The transition between dials does not behave by the same rules as the normal process—in this period the controller changes from one sequence to another, but not always at the same time. Also, because these periods occur just several times a day, we do not have many patterns that support them so that DT can not be trained properly.
The intervals between 0 minutes, or 12:00 a.m., and 420 minutes, or 7:00 a.m., as well as between 1230 minutes, or 8:30 p.m., and 0 minutes represent the operation in actuated mode. The traffic patterns are very unstable, and the measurements contain larger gaps. It can happen that if the gaps between the entries are several seconds long, the vehicle can pass a detector in the time between the last measurement and the phase change. In this case the controller takes the call into account, but we do not have any log data with this call. The inspections of the individual misclassification cases supports this hypothesis.

![Histogram of misclassifications](image)

Figure 8.1.: Histogram of misclassifications. Most misclassifications occur during the dial transition times or at night.

Nevertheless, decision tree prediction results are already good enough to be used, for example, in traffic simulations or other applications. In order to improve the quality of the predictors, we would need to change the method of information gathering, but that could be troublesome as it requires the change in the firmware for all traffic controllers.
8. Next Phase Prediction
9. Traffic Flow Prediction

Traffic flow information is important for accurate phase length and phase activity predictions as shown in Chapters 6 and 7. Moreover, it is essential for prediction of future phase changes. Therefore, in this chapter we discuss the problem of predicting vehicle calls from the traffic detectors.

Research in the area of short term traffic flow prediction concentrates on the prediction of changes in speed and traffic density for the next five to twenty minutes. Unfortunately, this resolution is too coarse for us. In order to produce accurate traffic signal prediction, we need to be able to predict the traffic flow on the second by second basis.

What information can be used as inputs for this prediction problem? Traffic controllers use the induction loop detectors to identify the demand on phases. This detectors are able to detect vehicles passing the intersection. Newer detectors can measure more information, e.g. vehicle speed. They can also differentiate between cars, e.g. using bluetooth. However, we can not rely on the availability of such information. Therefore, in this chapter we develop a model for traffic prediction that needs only the information about the vehicles passing the intersection. The models we develop can be extended with additional features, like speed or car identity, should these become available.

Our goal is to estimate the number of vehicles $X(t)$ passing the intersection in one direction during the time interval $[t, t + T]$. The number of vehicles is measured by the number of vehicle calls registered from the vehicle detectors. Formally, this is a regression problem and is formulated similar to that in Chapter 6. Therefore, we discuss only the issues specific to the traffic flow prediction.

Independently of the prediction models used, we can say that the number of vehicles consists of the “predictable” and “unpredictable” parts [31]:

$$X(t) = X^p(t) + X^u(t),$$  \hspace{1cm} (9.1)

where $X^p(t)$, the predictable part, represents the vehicles detected on the upstream intersection, while $X^u(t)$, the unpredictable part, represents the vehicles that arrive from the other sources like parking lots, garages, shops, etc. We have no means to estimate the unpredictable part and so we have to treat it as noise. The remainder of this chapter focuses on the prediction of $X^p(t)$.

9.1. Models

We discuss two models for traffic flow prediction. One is grounded in probability theory, the other is an adaptation of RNN for spatio-temporal unfolding. Although these models use different techniques, the features we will employ are the same and given by the number of detected vehicles in the upstream intersection at different moments of time in the past.

Both models can integrate new types of detectors: probabilistic model using DKF, and RNN as an additional input to the input cluster. Hence, these models allow the system to satisfy the requirements to extendability and the use of minimum amount of information described in Section 1.3.
9.1.1. Probabilistic traffic model

Let us consider a sequence of adjacent intersections as shown in Figure 9.1. Two intersections $S$ (source) and $D$ (destination) are connected along the main road from west to east. Formally, the activity of the phase $i$ in the intersection $D$—the function $f_{iD}(t; X_{iD}(t), I_{iD}(t))$—is a function of time $t$, the state of external variables $X_{iD}(t)$ and the state of the internal variables $I_{iD}(t)$. Both variables depend on time themselves. An example for external variables is vehicle calls, while for internal variables it is cycle length. Our task is to find and estimation for the external state $X_{iD}(t)$, which, in our case, is reduced to the vehicle calls only.

![Figure 9.1: Two intersection S (source) and D (destination) each have multiple phases. Our goal is to predict all vehicle calls for intersection D based on data from adjacent intersections (e.g. S). The phases of interest are in direction from S to D.](image)

Vehicle calls turn on every time the car passes the vehicle detector. Vehicles moving from intersection $S$ to intersection $D$ cause first the detectors on $S$ and then the detectors on $D$ to fire calls. We can formulate this causality in terms of probabilities:

$$p(\text{vehicleCall}i_D(t_S + T) | \text{vehicleCall}i_S(t_S)) = f_{SD}(T) \cdot p_{iSD}, \quad (9.2)$$

where $f_{SD}(T)$ is the probability that a vehicle drives from $S$ to $D$ in time $T$, and $p_{iSD}$ is the probability that a vehicle is passing the detector $i_S$ and then the detector $j_D$.

If we can estimate (9.2), then we can calculate the number of vehicle calls as

$$X_{iD}(t) = \sum_{T = t - \Delta t}^{t} \sum_{i_S} p(\text{vehicleCall}i_D(t) | \text{vehicleCall}i_S(T)) \cdot X_{iS}(T), \quad (9.3)$$

with $X_{iS}(T)$—the number of detected vehicles on phase $i_S$ at time $T$— being either an estimation or a true measurement from the detector at time $T$. Equation (9.3) means that the total number of vehicles detected on lane $i_D$ at time $t$ is equal to the expected number of vehicles coming from every lane $i_S$ in the upstream intersection that would take time between $\Delta t$ seconds and 0 to cross the link between $S$ and $D$.

Let us now consider the ways to estimate $f_{SD}(T)$ and $p_{iSD}$ from data.
9.1. Models

Estimation of $f_{SD}(t)$

We assume that the vehicles are traveling with constant random velocity drawn from normal distribution $V \sim N(\mu, \sigma)$. The work by Bennett supports this assumption \[61\]. We can estimate the cumulative distribution function for time as

$$F_T(t) = P(T \leq t) = P\left(\frac{|SD|}{V} \leq t\right) = P\left(V \geq \frac{|SD|}{t}\right) = 1 - P\left(V \leq \frac{|SD|}{t}\right) = F_V\left(\frac{|SD|}{t}\right),$$

and then the probability density function as its derivative:

$$f_{SD}(t) = \frac{\partial}{\partial t} F(t) = \frac{\partial}{\partial t} \left(1 - F_V\left(\frac{|SD|}{t}\right)\right) = f_V\left(\frac{|SD|}{t}\right) \cdot \frac{|SD|}{t^2} \cdot \exp\left(-\frac{\left(\frac{|SD|}{t} - \mu\right)^2}{2\sigma^2}\right).$$

(9.5)

The parameters $\mu$ and $\sigma$ can be estimated using maximum likelihood method with initial average speed equal to the speed limit on the link and standard deviation equal to 10 mph.

Estimation of $p_{isjd}$

Let us consider the lanes in one direction of the arterial road as shown in Figure 9.1. Depending from which lane in intersection $S$ it is coming, a vehicle has different choices to move. These choices are represented by the Markov model shown in Figure 9.2. The states of the Markov model represent the lanes, the transitions represent the movement from one lane in source intersection to another in destination intersection.

![Markov model](image)

Figure 9.2.: Markov model describing all possible transitions of vehicles coming from $S$ and going through $D$.

Let us consider the transition probability matrix for this model. The matrix has entries $p_{ij}$ for a probability that a vehicle will pass the intersection $S$ on lane $i$ and the intersection $D$ on lane $j$. For the expected numbers of vehicles passing the intersections the following equation should hold:
9. Traffic Flow Prediction

\[
\begin{bmatrix}
    p_{2S2D} & p_{7S2D} & p_{8S2D} \\
    p_{2S5D} & p_{7S5D} & p_{8S5D}
\end{bmatrix}
\begin{bmatrix}
    n_{2S} \\
    n_{7S} \\
    n_{8S}
\end{bmatrix}
= \begin{bmatrix}
    n_{2D} \\
    n_{5D}
\end{bmatrix},
\]  
(9.6)

with \( n_{iX} \)—the number of vehicles detected on the lane \( i_X \).

Let us rewrite Equation (9.6) as

\[
\begin{bmatrix}
    p_{2P}^T \\
    p_{5D}^T
\end{bmatrix}
\begin{bmatrix}
    x
\end{bmatrix}
= y.
\]  
(9.7)

Now it can be transformed to a linear equation in \( p \)

\[
\begin{bmatrix}
    x^T & 0 \\
    0 & x^T
\end{bmatrix}
\begin{bmatrix}
    p_{2D} \\
    p_{5D}
\end{bmatrix}
= y.
\]  
(9.8)

Aggregating measurements \( x \) and \( y \) for different time slots into matrices \( X \) and \( Y \), we can compile an overdetermined linear system that can be solved by minimizing

\[
\min_{p \in [0,1]^6} \| Xp - Y \| \quad (9.9)
\]

s.t. \( p_{7S2D} + p_{7S5D} = 1 \)  
(9.10)

\( p_{8S2D} + p_{8S5D} \leq 1 \)  
(9.11)

\( p_{2S2D} + p_{2S5D} \leq 1 \).  
(9.12)

The constraint (9.10) means that a vehicle coming from \( 7S \) has to appear either on \( 2D \) or on \( 5D \), while constraints (9.11) and (9.12) says that the vehicles from \( 2S \) and \( 8S \) may pass intersection \( D \), or they also may disappear from our detectors if they drive in northbound or southbound directions.

One can further improve the estimation of the transition probability \( p \) from Equation (9.8) using new observations of \( n_i \). We suggest that this can approached using the discrete Kalman Filter (DKF) paradigm. Appendix \( A \) suggests the adjustment of the DKF equations for estimation of \( p \). A full treatment of this approach is a direction of further work.

9.1.2. Spatio-temporal unfolding with recurrent neural networks

Let us again consider the two intersections in Figure 9.1. Some vehicles detected in the intersection \( S \) get to the link between \( S \) and \( D \), and after a certain time they are detected by the detectors in the intersection \( D \). The speed they move through the intersection depends on the driver’s abilities and the driving speed, but also on the link capacity and the load level.

Now consider the link between \( S \) and \( D \) as a dynamical system. The detectors on \( S \) produce the input to the system every second. This input corresponds to the output of the system after a certain time lag—the vehicles detected in the intersection \( D \) and coming from direction \( S \).

We can now adapt RNN for predicting the vehicle calls as shown in Figure 9.3. The inputs \( x \) of the RNN in a certain cluster are the numbers of vehicles detected by the vehicle detectors near stop lines on the three input directions. In our example, they contain the information from the vehicle detectors on lanes \( 2S,7S, \) and \( 8S \). Similarly, the outputs \( y \) of the networks are the vehicles detected on the two destination directions of the downstream intersection. In our example, they contain the information from the vehicle detectors on lanes \( 2D \) and \( 5D \). We can think of the system as being unfolded in space and in time. Indeed, inputs and outputs are physically separated, and we use finite unfolding in time method for RNN. Using delayed connections we can reproduce the dynamics from the real world, since knowing the length of the link and its speed limit, we can estimate the size of the temporal unfolding. This temporal unfolding window should be sufficient to capture the same vehicles on the input and output even if we have no identification for individual vehicles.
9.2. Evaluation

We can not present the empirical evaluation of the traffic flow prediction methods described. As it turned out, the dataset has only the summarized vehicle call information that contains the data from one or several vehicle detectors (e.g. advancing and stop-line detectors for two lanes in the same direction) together with recalls, but we could not obtain any additional information that would help us to separate the signals of individual detectors from each other. Without the ability to clean the signal, the experiments we conducted did not show any conclusive results we could interpret as a success or failure.

The approach to obtain the data from the traffic simulation programs turned out to be unfeasible in the scope of this thesis. The modern traffic simulation programs, like TRANSIMS \[14\],...
or SUMO [62], require the reproduction of a complete city with real population statistics and activity information to obtain plausible information about vehicle movements. This simulations and application of the discussed traffic prediction methods for very short term prediction could make a scope for a master thesis on their own.

Based on our work for this thesis, we can say that solutions for very short term traffic flow predictions are important for traffic control applications and the prediction of traffic signal in actuated mode. There are also promising methods to solve the problem, which, altogether, make it an interesting topic for further research.
10. Components Integration and Interaction

So far, we described the individual modules for the solution of the prediction problems. This modular structure allowed us to handle the problem complexity as well as to face the challenge of minimal human intervention. Different prediction modules cover all aspects of traffic signal prediction and can be trained from the single dataset.

In this chapter, we show how these modules can be combined to a system used by ITS applications. We illustrate the integration ideas using two applications: GLOSA and RTCS. Finally, we show how, using a sequence of predictors, one can extend the prediction horizon of the system.

We conducted our evaluation using RapidMiner v. 5.1 [63]. It has great capabilities for experiments, including data transformation and logging, it is also able to export the trained models as PMML files. PMML, or Predictive Model Markup Language, is an XML-based markup language for vendor independent serialization of machine learning models [64]. It was developed by Data Mining Group and is supported by the machine learning and data mining products of the major vendors, including Google, IBM, SPSS, Oracle, Microsoft, and SAS. Therefore, the integration of the models into a product should not present any significant technical challenges. In following we concentrate on the conceptual description of the integration.

10.1. Integration in Green Light Optimized Speed Advisory application

The flowchart in Figure 10.1 illustrates the routine that integrates individual prediction components for the traffic signal prediction used in GLOSA. The traffic controller sends the new information about its internal state and the measurements from the detectors to the prediction system.

At first, the new data are pre-processed and combined with the past measurements using windowing algorithm. If the phase was just activated, the system estimates the length of the new phase. If the phase is active for some time already and the data about the current phase are available, the system performs the long-term activity estimation to refine the length prediction. If the expected phase length is less than 10 seconds, the system uses the short term phase activity prediction to obtain an especially accurate estimation. The estimation of the current phase length gives the information about the expected green time to the routine that formats the output for GLOSA.

Concurrently to the estimation for the active phase, the system also calculates the probability for each other phase in the ring to be activated and the expected length of the phases. The sum of these expected phase lengths gives the estimation of the red time for the current movement direction in the output routine.

Figure 10.2 shows how different estimations influence the duration of the red and green periods of the phase activity function. The values of this function are then encoded and transmitted to the GLOSA receivers of the vehicles traveling in the corresponding direction.

If the traffic flow prediction module will be available in the future, the system can easily integrate it to improve the quality of the traffic signal prediction.
Figure 10.1.: Integration of individual prediction modules into a traffic signal prediction system for GLOSA application.
10.2. Integration in Realistic Traffic Control in Simulations application

The flowchart in Figure 10.3 illustrates the routine that integrates individual traffic signal and phase prediction components into a traffic controller system used by a traffic simulator. The open-source traffic simulation programs like SUMO or TRANSIMS allow developers to integrate their own traffic controller algorithms as libraries that implement predefined interfaces.

The traffic simulation program calls the algorithm for the short term activity prediction. Given the information from the simulation program about the time of the day, state of the vehicle detectors, current phase length, etc., as well as the information from the past retrieved from the database, the algorithm decides if the currently active phase has to be interrupted or not. If the phase change is needed, the corresponding prediction module estimates the new active phase and sends the updated information back to the simulation program.

Using this method one can train arbitrary many predictors with the real log data from the emulated traffic controllers so that the integrated controller algorithms will show the behavior of the real controllers in the simulated area.

10.3. Extending prediction horizon

In Section 10.1 we already showed how to use the estimation for several phases in one ring to extend the prediction to the whole cycle. Using multistage prediction we can extend the prediction horizon even further. The principle of multistage prediction is illustrated in Figure 10.4.

Let us first recall the traffic flow predictors described in Chapter 9. In order to predict the number of vehicle calls in the future in intersection $D$, we used the number of vehicles calls measured in the adjacent intersection $S$ in the past. Suppose the intersection $D$ is connected to the intersection $E$ farther along the arterial road. We can use the same schema and predict the number of vehicle calls in the intersection $E$ even further in the future, if we use the predicted values from the intersection $D$. Hence, we can extend the prediction horizon by using the predictions in the near future as an input for the predictions in the further future.

Let us now consider a more comprehensive example. Suppose, we have two predictors $f$ and $g$. The predictor $f$ estimates the length of the phases in the next cycle depending on the
Figure 10.3.: Integration of individual prediction modules into a traffic signal prediction system for RTCS application.
past information about traffic flow and the lengths of phases in the past cycles. The predictor $g$ estimates the traffic flow using the data from adjacent intersections described previously. By alternating the predictions of $f$ and $g$ and using the predictions for the time intervals $t, t + 1, \ldots, t + N - 1$ as an input we can calculate the prediction for the time interval $t + N$. However, one should be cautious about extension of the prediction horizon for a long period of time as the error of the predictors will accumulate with every step.

Figure 10.4.: Multistage prediction process. The estimations of two alternately applied predictors $g$ and $f$ (bold squares) serve as inputs for next iterations. The method allows to increase the prediction horizon, although, it is prone to error accumulation.

Alternatively, one could use overshooting \cite{49} to train RNN to extend the prediction horizon. The scheme of an RNN with overshooting is shown in Figure 10.5: the first 4 time steps of the dynamical system are trained with the corresponding input, but the last 3 do not receive any input and are used to predict future outputs. RNN with overshooting are not as prone to error accumulation as multistage models, but they may be harder to train.

Figure 10.5.: RNN with overshooting. The states $s_{t+1}, s_{t+2}, s_{t+3}$ do not receive any input but make predictions. The method allows to increase the prediction horizon, although, such RNN are harder to train.
11. Conclusions and Future Work

Although the absolutely precise traffic signal prediction for actuated traffic control is impossible, we could show that by decomposing the problem into a number of sub-problems, we are able to obtain the predictors with accuracy that satisfies the requirements of a number of concrete ITS applications.

Even without traffic information available, we could build a predictor with RMSE of less than 5 seconds for coordinated-actuated operation. Such predictor satisfies the requirements of GLOSA, TSAN, RLDA, and IRLA that require longer prediction horizon and lower accuracy. The predictors for the traffic signal in fully-actuated mode have shown RMSE of 11 seconds and more so that these predictors cannot be used. However, the integration of the ITS applications in the time between 7:00 a.m. and 20:30 p.m. can already bring significant improvements.

Almost all ITS applications can profit from the phase activity prediction that allows to integrate the available information about the current phase as well as to control the horizon-accuracy trade. We have shown that the predictions for RTCS, RLVA, RLDA, and GLOSA have the accuracy of about 98% for both coordinated-actuated and fully-actuated modes. With future traffic flow information available, the prediction in fully-actuated mode is as accurate as in coordinated-actuated mode, in contrast to the results of the phase length prediction. Without traffic information, the accuracy drops to about 90%.

The prediction of the phase that will be activated next heavily depends on the availability of the future traffic information. While the “naive” prediction with fixed phase sequence assumption would lead to accuracy of only between 74% and 93%, using decision trees, we could achieve the accuracy of between 95% and 100%. These results can be farther improved with the development of the methods for information gathering from the traffic controllers. The module for next phase prediction is important for GLOSA, TSAN, RLDA, and RTCS.

The traffic flow prediction, so it was not the primary objective of this work, is essential for robust and correct traffic signal and phase prediction. Moreover, TSPT can use these predictions directly for safe phase switching. We presented two methods for very short term traffic flow prediction using probabilistic model and recurrent neural networks, although, we could not perform the evaluation of the methods.

Besides functional, the nonfunctional requirements are important for successful integration if ITS applications. We focused on developing prediction models that would incorporate only general nature a prediction problem, like feature set. The instantiation of a concrete predictor from these models can be simply trained and retrained to fit concrete traffic controllers (requirements to work in heterogeneous environment and adaptability). The modular structure of individual predictors allowed us to handle the problem complexity as well as to face the challenge of minimal human intervention, since prediction modules cover all aspects of traffic signal prediction and can be trained from the single dataset. Given enough computational capacity, the optimal training parameters for every intersection can be automatically obtained using cross-validation method. This would guarantee the optimal prediction quality.

The gathering of the traffic data and evaluation of the traffic prediction methods is a promising direction for future work, along with the extension of the probabilistic prediction method with discrete Kalman filter. Another important step for future work is practical testing of the prediction methods and ITS applications in the real-world applications as well as quantification of the advantages for fuel efficiency and gas emission reduction.

Another question that raises in the context of the work is the controller-vehicle influence loop. Currently, traffic controllers react on the signals from vehicle detectors. The integration
of the applications, like GLOSA, would make the drivers to change their speed according to the future traffic controller behavior. Theoretically, this could lead to the instabilities and traffic congestions, though, we do not consider this as probable. Using traffic simulations, one could investigate the dynamic of the controller-vehicle loop in different scenarios. The traffic signal prediction methods developed in the scope of this thesis can contribute to these simulations.

As already described in Section 1.4, we ignored the cases where the normal traffic signal schedule is distorted due to preemptive calls from a fire truck or an emergency vehicle. The cases of preemptive calls occur seldom enough to be treated as outliers at first, e.g. in the available dataset, we did not have any single case of a preemptive call. In future work, one could study the algorithms for preemptive calls handling as a special case of the traffic signal prediction problem.

In this thesis we deliberately avoid the discussion of car-infrastructure communication technologies and protocols, since we focused on the prediction problems and not on the technical realization. Nevertheless, there is a number of issues that should be investigated before the ITS applications, like GLOSA, can find their way into the vehicles, e.g. definition of the communication protocols, communication technology, or the trend between centralized and decentralized computation and communication.
Appendix
A. Dynamic Filtering for Estimation of Transition Probabilities

Discrete Kalman filter (DKF) is a statistical method for estimation of model parameters in a noisy environment [65]. DKF can allow us to combine the information from different detectors and to automatically adjust the transition probabilities to changes in traffic patterns over time.

Originally, the method aims to estimate the state $x_k$ of a discrete-time controlled process, where the state is governed by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \quad (A.1)$$

and the observable measurements $z_k$ by equation

$$z_k = Hx_k + v_k. \quad (A.2)$$

The random variables $w_{k-1}$ and $v_k$ represent the process and measurement noises respectively. The method assumes, that the noise variables are normal distributed with zero mean and covariance matrices $Q$ and $R$. The matrices $A$, $B$, and $H$ represent linear transformation in the equations, the variable $u_{k-1}$ is an optional input.

DKF algorithm consists of two steps: predict and update, that are alternately repeated in an iterative process. In the step “predict”, the new a priori state estimate $\hat{x}_{k-1}$ is obtained from the a posteriori state estimate in the previous step $\hat{x}_{k-1}$:

$$\hat{x}_k = A\hat{x}_{k-1} + Bu_{k-1}. \quad (A.3)$$

Additionally, we estimate the new a priori error covariance matrix

$$P_k = AP_{k-1}A^T + Q. \quad (A.4)$$

In the step “update”, we estimate the a posteriori values of state $\hat{x}_k$ and error covariance matrix $P_k$ using new observation $z_k$ and Kalman gain matrix $K_k$:

$$K_k = P_k H^T (HP_k H^T + R)^{-1}, \quad (A.5)$$

$$\hat{x}_k = \hat{x}_{k-1} + K_k(z_k - H\hat{x}_{k-1}), \quad (A.6)$$

$$P_k = (I - K_k H)P_{k-1}. \quad (A.7)$$

For detailed explanation of the algorithm and for mathematical background of the equations, we refer to the literature (e.g. [65]).

Similar to DKF equations, we can use the new observations of $n_{ij}$ and Equation (9.8) to improve the estimation of $p$ (state of the system) in an iterative process. In order to adapt DKF Equations (A.1) and (A.2), we define the variables:

$$x_k := p$$

$$A := I$$

$$B := 0$$

$$z_k := \begin{pmatrix} n_{2D} \\ n_{5D} \end{pmatrix}$$

$$H_k := \begin{pmatrix} n_{2S} & n_{7S} & n_{8S} & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{2S} & n_{7S} & n_{8S} \end{pmatrix}$$
A. Dynamic Filtering for Estimation of Transition Probabilities

The difference to the canonical formulation is in the definition of $H$, which is constant in DKF and variable in our form. However, we do not see any problems with our formulation apart from the increased computational costs.

Moreover, the noise covariance matrices $Q$ and $R$ can be estimated automatically from data [66], so that the algorithm can process without human intervention.
Bibliography


