Travel Time Prediction Using Ensemble Learning Algorithms

by Xia Li

Thesis submitted to The University of Nottingham
for the degree of Doctor of Philosophy, July 2018


The University of
Nottingham

Contents

List of Figures

List of Tables

Notations

Abbrevations

ATIS	Advanced Traveller Information System
GPS	Global Positioning System
VDS	Vehicle Detection System
APC	automatic passenger counter
AVI	automatic vehicle identification
GIS	geographic information system
BPR	US Bureau of Public Roads
CNZOS	the Port of Ningbo-Zhoushan, China
SD	standard deviation
CV	coefficient of variation
IDR	interdecile range
IQR	interquartile range
BI	buffer index
MI	misery index
CA	Celluar Automaton
TVC	time-varying coefficient
LSE	least squares estimation
OLS	ordinal least squares estimation
GLS	generalised least squares estimation
CCF	cross-correlation function

CG	conjugate gradient
LM-BR	Bayesian regularised Levenberg-Marquardt algorithm
L-BFGS	the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm
GA	genetic algorithm
GML	Gaussian maximum likelihood
ES	exponential smoothing
LEM	local estimation method
SE	squared exponential kernel
RQ	rational quadratic kernel
RQ_{ARD}	rational quadratic kernel with automatic relevance determination
PER	periodic kernel
LIN	linear kernel
BO	Bayesian optimisation
BBO-LP	a batch Bayesian optimisation via local penalisation
TSA	time series analysis
AR	autoregression
MA	moving average
ARMA	autoregressive moving average
ARIMA	autoregressive integrated moving average
ARCH	autoregressive conditional heteroskedasticity
GARCH	generalised autoregressive conditional heteroskedasticity
SV	stochastic volatility
KF	Kalman filter
EKF	extended Kalman filter
MRF	Markov random field
ANN	artificial neural network

MLPs	multilayer perceptrons
SHL-MLPs	single-hidden-layer multilayer perceptrons
RNN	Recurrent artificial neural network
Elman NN	Elman neural network
SSNN	state-space recurrent neural network
TDNN	time-delay neural network
NARX	nonlinear autoregressive with exogenous inputs model
TDSSNN	time-delayed state-space neural network
STDNN	space-time delay neural network
LSTMNN	long short-term memory neural network
DNN	deep Neural network
SAE	stacked autoencoder
CNN	convolutional neural network
eRCNN	error-feedback recurrent CNN
RBM	restricted Boltzmann machine
PCA	principal components analysis
RR	ridge regression
SVR	support vector regression
GPR	Gaussian process regression
SGPR	sparse Gaussian process regression
CART	the classification and regression trees algorithm
RF	random forests
ABRT	adaptive boosting regression trees
GBRT	gradient boosting regression trees
DTC	the deterministic training conditional for sparse Gaussian process regression
VDTC	the variational deterministic training conditional for sparse Gaussian process regression

FITC	the fully independent training conditional for sparse Gaussian process regression
CV	cross validation
CVGS	cross validated gird search
BDP	prediction before departure
OGP	prediction for ongoing trips
PTTS	the pseudo travel time sampling algorithm

Symbols

TT	travel time
t, t^*	departure time
t'	arrival time
r	route name/index
$TT_i(t)$	individual travel time of vehicle i with departure time t
$TT(p)$	aggregated travel time of vehicle i with departure time interval p
$TT_i^d(t)$	individual departure travel time of vehicle i with departure time t
$TT_i^a(t)$	individual arrival travel time of vehicle i with departure time t
$TT^d(p)$	aggregated departure travel time with departure time interval p
$TT^a(p)$	aggregated arrival travel time with departure time interval p

TT_{free}	expected (or mean) departure travel time under free-flow condition with departure time t
v_{free}	expected (or mean) speed under free-flow condition
v	general expression of speed
u	mean speed
L	road (or path) length
D	general expression of delay (or average delay per vehicle)
$D(t)$	expected (or mean) delay with departure time t
ΔT	sampling rate of trajectory data
s	standard deviation
$\widehat{c_v}$	coefficient of variance
r_{IDR}	interdecile range
r_{IQR}	interquartile range
r_{BI}	buffer index
q_n	the nth percentile travel time
ε	positional error of a trajectory (GPS) report
ε_u	computational error for mean speed calculation
χ	flow-capacity ratio
q	traffic flow rate, or the vehicle arrival rate (vehicles/hour)
c	capacity of intersection approach (vehicles/hour)
g	effective green light interval (seconds)
C	traffic signal cycle length (seconds)
N_{queue}	number of vehicles in queue
$\mathcal{D}$	dataset
N	the number of instances (or examples) in $\mathcal{D}$
d	the dimensionality of the input features in $\mathcal{D}$
$\mathbf{X}$	input vectors or input matrix

$\mathbf{X}^T$	the transpose of $\mathbf{X}$
$\mathbf{y}$	output vector
$\mathbf{x}_i$	the ith input vector
$\mathbf{x}_*$	new test input vector
y_i	the corresponding output for $\mathbf{x}_i$
e	error
ϵ	noise
σ_ϵ^2	variance of noise
$f(\mathbf{x})$ or $\mathbf{f}$	Gaussian process (or vector of) latent function values, $\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_n))^T$
$\mathbf{f}_*$	Gaussian process (posterior) prediction (random variable)
$\bar{\mathbf{f}}_*$	Gaussian process posterior mean
$\mathbb{V}$	variance
$\boldsymbol{\theta}$	vector of hyperparameters (the parameters of covariance functions)
$m(\mathbf{x})$	the mean function of a Gaussian process
$\mathbf{m}$	the vector of the mean functions
k or $k(\mathbf{x}, \mathbf{x}')$	covariance (or kernel) function evaluated at $\mathbf{x}$ and $\mathbf{x}'$
K	$n \times n$ covariance (or Gram) matrix
K_*	$n \times n_*$ matrix $K(\mathbf{X}, \mathbf{X}_*)$, the covariance between training and test cases
K_{**}	$n_* \times n_*$ matrix $K(\mathbf{X}_*, \mathbf{X}_*)$, the covariance of test cases
$\|K\|$	determinant of K matrix
I or I_n	the identity matrix (of size n)
$y\|x$	conditional random variable y given x
$p(y\|x)$	probability (density) of conditional random variable y given x
$p(x, y)$	joint probability density of random variables x and y

$\mathcal{L}(\cdot)$	loss function in general
$\sim$	distributed according to; example: $x \sim \mathcal{N}(\mu, \sigma^2)$
$\mathcal{N}(\mu, \sigma^2)$	a Gaussian (Normal) distribution with mean μ and variance σ^2
$diag(w)$	(matrix argument) a vector containing the diagonal elements of matrix W
$Tr(\boldsymbol{A})$	trace (or square) of matrix $\boldsymbol{A}$
$KL(p\|\|q)$	the Kullback-Leibler divergence between probability distribution p and q
$\mathbb{E}[x]$	the expected (or mean) value of a random variable x
$\Re$	the real numbers
$\log(z)$	natural logarithm (base e)
$\mathcal{O}(\cdot)$	big Oh; for functions f and g on N, we write $f(n) = \mathcal{O}(g(n))$ if the ratio $f(n)/g(n)$ remains bounded as $n \to \infty$

Abstract

In the research area of travel time prediction, the existing studies mainly focus on aggregated travel time prediction (without distinguishing vehicle types) or travel time prediction for passenger vehicles. The travel time prediction for freight transportation has not received enough attention from researchers. Only a few relevant studies can be found in the literature, and the proposed methods are usually very simple and lack comparisons with more advanced methods. Although many believed that a better prediction model can be developed using more advanced techniques such as artificial neural networks or autoregressive conditional heteroscedastic models, it is usually difficult and costly to train these models and the model interpretability is poor. There is a demand for 'off-the-shelf' methods with good performance, ease of implementation and good model interpretability. Thus, the aims of this thesis are: (1) developing some 'off-the-shelf' data-driven methods to predict travel time for freight transportation; (2) creating a comprehensive understanding of how the developed methods can be more effectively applied for general travel time prediction problems. Its two main contributions are: (1) it develops data-driven travel time prediction methods for freight transportation by utilising freight vehicles' trajectory data; (2) it investigates the relation between features and performance and discovers the combinatorial effects of features under the effects of different noise processes and different model fitting strategies. The experimental results show that useful features can be mined from the trajectory data to enhance the travel time prediction for freight transportation. The developed methods outperform some of the state-of-the-art data-driven methods.

Acknowledgements

First, sincere gratitude to my supervisor, Professor Ruibin Bai, for his trust, patience, advice, and encouragement. He is always supportive and helped me 'hack my way in the jungle'. His challenging of my half-baked ideas during my PhD study helped make me become a better critical thinker and independent researcher.

Sincere appreciation to my UK supervisors, Dr Peer-Olaf Seibers and Dr Christian Wagner, for their advice, feedback, support, and encouragement.

Dedicated to my wife Jessica, for the unwavering support and encouragement, and for her love.

Dedicated to my parents, my parents in law, and all my friends, for their support and encouragement.

Thanks to my friend Horia Alexandru Maior, for the encouragement during the final push.

My graduate study was supported by the International Doctoral Innovation Centre of The University of Nottingham Ningbo China, The University of Nottingham, Ningbo Science and Technology Bureau, China's MoST and a grant from National Natural Science Foundation of China.

Declarartion

I hereby declare that this thesis is my own work, contains no plagiarism, and has not been submitted, either in the same or different form, to this or any other university for a degree.

Signature:

Chapter 1

Introduction

1.1 Background

Travel time is essential to transportation management. It is important for road network performance evaluation. For road network authorities, accurate travel time information can help evaluate the network performance, thus better traffic management strategies or tools can be designed and implemented. For individual travellers, accurate travel time information can help make better travel plans (e.g., transportation mode, departure time and route selection). It can also help public transportation (e.g., bus, tram) and freight transportation service providers make better service planning and scheduling. A large number of travel time prediction models have been incorporated as part of the Advanced Traveller Information System worldwide.

Performing travel time prediction is a challenging task. There are so many latent influencing factors (e.g., driver's driving behaviours, surrounding traffic condition, weather, incidents, road works) so that it is extremely difficult to jointly and explicitly model the relation between travel time and those factors. Even if one can explicitly define a model that works well in one case, it may dramatically fail in many other cases because the effect of these factors on travel time can be different from case to case. This motivated researchers to develop methods that are not only

able to capture the complex relation between travel time and influencing factors, but also have good generality. Thus, many data-driven methods have been developed in the recent decade. These data-driven methods are mainly based on statistical learning techniques. With these methods, prediction models can be built only using data. Thus, the quality of a prediction model very much relies on the quality of data.

When there is no significant difference between historical and future traffic, travel time prediction can fully rely on historical travel time. However, if road capacity drops due to an incident or temporary road work, or the quality of historical data is poor, the prediction that only relies on historical data may yield a poor result. Thus, real-time data are usually incorporated into the model. Typically, traffic flow data that is collected using roadside sensors such as loop detectors or automatic vehicle detectors are used for the prediction. These data are usually owned by traffic management authorities which may not be accessible to researchers and private transportation service providers. As a result, many researchers turn to seek an alternative for traffic flow data. A popular approach is using trajectory data of probe vehicles (e.g. taxi) to learn the traffic of upstream roads. However, this approach may not be feasible if only few probe vehicles are available on the target roads. Unfortunately, data is not the only issue for applying data-driven methods. There are many other concerns about the methods including the ease of implementation, the computational cost, the degree of automation, the scalability and the model interpretability.

Freight transportation has been rapidly developed due to the rise of consumption-driven lifestyle and the acceleration of economic globalisation in recent years. For freight transportation service providers, travel time prediction not only helps make better transportation service tracking but also helps make better task planning, thus increasing the competitiveness. A large number of data-driven methods have been developed for short-term aggregated travel time prediction (without distinguishing vehicle type) or travel time prediction for passenger vehicles. Only a few have been

developed for freight transportation. There lacks an in-depth understanding of how data-driven methods can be effectively applied for freight transportation. Besides, many freight transportation service providers own a huge amount of trajectory data of their fleet, but these data are far away from being effectively utilised [94]. Thus, there exists a mismatch between the demand for travel time analysis and the utilisation of trajectory data for freight transportation. That motivates us to conduct travel time prediction study for freight transportation by utilising their trajectory data.

Next, the objectives and scopes of this PhD research are defined, followed by the main contributions and the relevance of this research. The outlines of this thesis are introduced in the last section of this chapter.

1.2 Research Objectives and Scope

1.2.1 Research Objectives

The main objectives of this PhD research are to: (1) developing some data-driven methods for travel time prediction for freight transportation by utilising freight vehicles' trajectory data; (2) and creating a comprehensive understanding of how the methods can be effectively applied for general travel time prediction problems.

Several indicators are selected to evaluate the performance of the methods to be developed:

- *Accuracy* is the most critical indicator for the performance evaluation. It measures the prediction error, i.e., the difference between the predicted result and the actual result. The smaller the error, the higher the accuracy. Both absolute and relative error metrics will be used in our study. Comparative study will be conducted to see if the methods yield higher accuracy than some state-of-the-art data-driven methods.

- *Robustness* is another important indicator for the performance evaluation. The robustness will be evaluated in two criteria: (1) the capacity of handling the variation of traffic condition; (2) the capacity of handling the variation of data quality [120]. The first criterion can be assessed by comparing the prediction error over different traffic conditions or different road segments. The second criterion can be assessed by observing the performance stability on the change of data quality.

- *Practicability* is important for real-world applications. The practicability will be evaluated in three criteria: (1) the cost of implementation; (2) the cost of training; (3) and the cost of prediction. The first criterion can be assessed by examining the difficulty of implementation. The second criterion can be assessed by examining the difficulty and the computational cost of the model fitting process. The third criterion can be assessed by observing the computational cost of the prediction process.

- *Model interpretability* has receives more attention in recent years. It helps researchers to understand the prediction process better, particularly when things go wrong. The model interpretability can be assessed by examining the difficulty of feature importance analysis with the resulted model.

Features are essential to the success of the data-driven learning approach. As Flach described in his book, '*a model is only as good as its features*' [47]. Thus, the construction and selection of features become crucial. Features may interact in various ways which can result in positive or negative combinatorial effects on prediction performance. Moreover, it is hard to determine if such interaction should be exploited or ignored. To utilise the developed methods more effectively for general travel time prediction problems, various feature combinations will be composed and investigated. The investigation aims to: (1) examine the prediction performance using different features; (2) compare the prediction performance among different

data-driven methods over different features; (3) and discover any combinatorial effects of the features. The above four performance indicators will be used to evaluate the prediction performance in the investigation study.

1.2.2 Research Scope

Typically, freight vehicles can be classified into three types by weight: light, medium, and heavy. Light and medium freight vehicles are mainly used for urban freight transportation. Heavy freight vehicles are mostly used for interurban transportation and intermodal transportation. This research focuses on heavy freight vehicles. We expect the findings of this research to benefit the logistics industry.

Travel time prediction for long-distance transportation is different from that for short-distance transportation. There is potentially more uncertainties for long-distance prediction than short-distance prediction which affects the way we design the methods. Most heavy freight vehicles use arterial roads or highways for interurban and intermodal freight transportations. In most cases, the routes are fixed. Thus, this research does not take into account route selection.

Travel time prediction can be categorised into long-term and short-term prediction based on the length of prediction horizon. Long-term prediction relies on the statistical analysis of historical data and the assumption of future traffic conditions. Inaccurate assumption may result in a poor prediction result. Short-term prediction is able to incorporate real-time data so that the generalisation of future traffic conditions does not rely on the assumption of future traffic. Thus, short-term prediction models are usually more robust to the variation of traffic conditions. This research focuses on short-term travel time prediction with the maximum prediction horizon of 30 minutes.

1.3 Contributions and Relevance

1.3.1 Contributions

The two main contributions of this PhD research are summarised as follows:

- It develops some 'off-the-shelf' data-driven travel time prediction methods for freight transportation by applying ensemble learning algorithms. All the features are extracted by mining the freight vehicles' trajectory data. The developed methods are able to handle temporally sparse and noisy data. They are easy to implement, fast to train and predict, and the resulted models are relatively easy to interpret.

- It creates a comprehensive understanding of how two popular decision tree ensemble methods, random forests and gradient boosting regression trees, can be used more effectively for general travel time prediction problems. It investigates the relation between features and performance, and discovers the combinatorial effects of the features under the effects of different noise processes and different model fitting strategies.

1.3.2 Scientific Relevance

Although decision tree ensemble algorithms such as random forests and gradient boosting regression trees are rising to stardom in the data mining community, they have not received enough attention for travel time prediction. This thesis not only shows the feasibility of travel time prediction for freight transportation using decision tree ensemble algorithms but also investigates their performances using feature combinations, under the effect of different noise processes and model fitting strategies. Real-time speed data are mined from vehicles' trajectory data and incorporated into the model, which is a promising attempt to extract some useful real-time data from trajectory data to enhance the prediction. This thesis also raises the concern

about the combinatorial effects of features when applying decision tree ensemble algorithms. That is, although decision tree ensemble algorithms have built-in feature importance analysis that can select 'best' features for the prediction, the effects can have a negative impact on the prediction performance. Hence, careful feature analysis is still needed before formally building the prediction model.

1.3.3 Practical Relevance

Incorporating real-time traffic flow data can be helpful to predict future traffic conditions. Unfortunately, these real-time data may not be accessible to researchers and private freight transportation service providers. Many freight transportation service operators own a huge amount of trajectory data of their fleets, but these data are far away from being effectively utilised. This thesis provides an economical and effective solution that when real-time traffic flow data is not available, we can still extract some useful real-time information by mining vehicles' trajectory data. A Bayesian optimisation approach is applied to replace traditional time-consuming cross validated grid search for model fitting. This makes the developed methods fast to train. Since the developed methods use ensemble learning algorithms to build prediction model, they are easy to scale to large data.

1.4 Thesis Outline

The rest of this thesis is organised as follows:

Chapter 2 introduces some definitions of travel times. Those definitions are used throughout the entire thesis. Understanding travel time is mainly about understanding the travel time variability. Various travel time variability measures are reviewed in this chapter.

Chapter 3 provides a comprehensive review of state-of-the-art travel time prediction methods. Naïve methods are the simplest type of methods because they

have no additional assumptions other than the data used and use very simple mathematical formulas. Except for naïve methods, most of the existing methods can be categorised into two types: model-based methods and data-driven methods. Essential works of the two types of methods are reviewed in detail. The advantages and disadvantages of these methods are discussed. Based on the review, one Bayesian ensemble algorithm and two decision tree ensemble algorithms are selected for the further studies.

Chapter 4 presents a travel time prediction study for freight vehicles using sparse Gaussian process regression (SGPR). Three road segments are selected from the intermodal drayage road network of the Port of Ningbo-Zhoushan (CNZOS). Two types of prediction tasks, prediction before departure (BDP) and prediction for ongoing trips (OGP), are performed with some real-time speed data mined from the vehicles' trajectory data. Different inducing set initialisation methods and inducing set sizes are examined to find out the best practice for SGPR in our case. The prediction performance of the SGPR models are compared with the support vector regression (SVR) models and the multilayer perceptrons (MLPs) for all cases.

Chapter 5 presents a travel time prediction study for freight vehicles using three decision tree ensemble algorithms: random forests (RF), adaptive boosting regression trees (ABRT), and gradient boosting regression trees (GBRT). The same datasets and prediction tasks are used as for Chapter 4. This chapter also addresses the drawback of the real-time features used in Chapter 4. The computational costs of these decision tree ensemble algorithms are measured and compared. The performance of the decision tree ensemble models are compared with the SVR models and the MLPs. In addition, a batch Bayesian optimisation approach is introduced to replace the traditional cross validated grid search approach for the model fitting.

Chapter 6 investigates the travel time prediction performance by the two popular

decision tree ensemble algorithms, RF and GBRT, using different feature combinations. The investigation is first conducted with pseudo travel time data generated using a pseudo travel time sampling algorithm (PTTS). PTTS allows generating travel time data using different noise processes so that we can study the prediction performance under the effect of different noises. Then, a case study is performed to verify the findings of the investigation using the artificially generated data. Besides, modular based model fitting strategies are examined in both cases to see if they are beneficial to the prediction.

Chapter 7 summarises the main conclusions and discusses the directions of the future research.

1.5 Published and Submitted Works

The published and submitted works are listed below:

Published:

X. Li, R. Bai, P. O. Siebers and C. Wagner, "Modeling urban road risky driving behaviors in China with multi-agent microscopic traffic simulation", *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Qingdao, 2014, pp. 1740-1745. DOI: 10.1109/ITSC.2014.6957944.

X. Li and R. Bai, "Freight Vehicle Travel Time Prediction Using Sparse Gaussian Processes Regression with Trajectory Data", *17th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, Yangzhou, 2016, pp. 143-153. DOI: 10.1007/978-3-319-46257-8_16.

X. Li and R. Bai, "Freight Vehicle Travel Time Prediction Using Gradient Boosting Regression Tree", *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Anaheim, CA, 2016, pp. 1010-1015. DOI: 10.1109/ICMLA.2016.0182

Submitted For Peer-review:

X. Li, R. Bai, P. O. Siebers and C. Wagner, "An investigation of decision tree ensemble methods for travel time prediction using various feature combinations", submitted for *Data Mining and Knowledge Discovery*.

Chapter 2

Travel Time Fundamentals

2.1 Introduction

Before getting into the problem of travel time prediction, it is important to lay out some basic definitions and elements involved in travel time prediction. Travel time prediction is mainly about understanding and modelling travel time variability. Various travel time variability measures are introduced in this chapter.

2.2 Definitions of Travel Time

This section lays out some definitions of travel time. Their definitions are given based on the definitions used in [120, 200].

Definition 1 ***Travel time*** *TT of a trip can be defined as the length of time that a vehicle takes from the start point of the trip to the end point of the trip. It can be expressed as:*

$$TT = t' - t$$

where t and t' denote the departure and arrival time instance of this trip respectively.

Definition 2 ***Individual travel time*** *$TT_i(t)$ is the time that an individual vehicle i traverses a road of interest at departure time t.*

Definition 3 ***Aggregated travel time*** *$TT(p)$ is the mean travel time time of all the vehicles that traverse a road of interest within the departure time interval $p = [t, t + \rho]$. The aggregated travel time can be expressed as:*

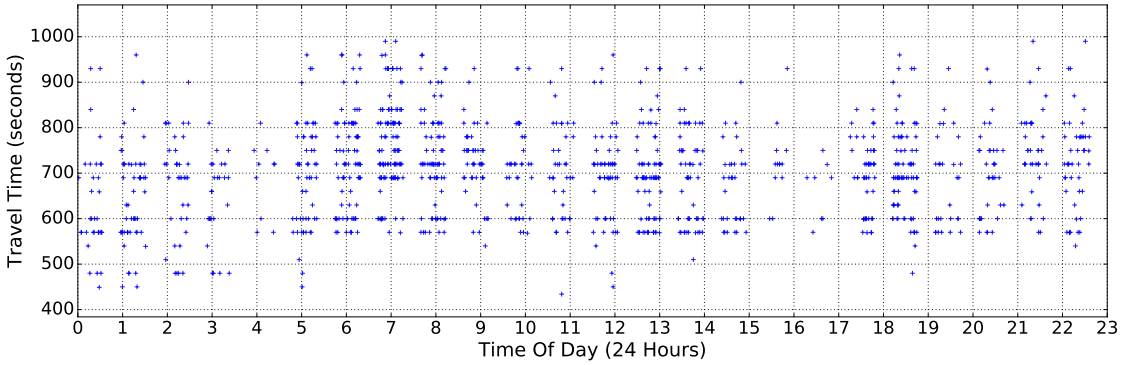
$$TT(p) = \frac{1}{N} \sum_{i=1}^{N} TT_i(t_i), \quad where\ t_i \in p$$

where N denotes the number of vehicles observed, t_i denotes the departure time of the ith vehicle.

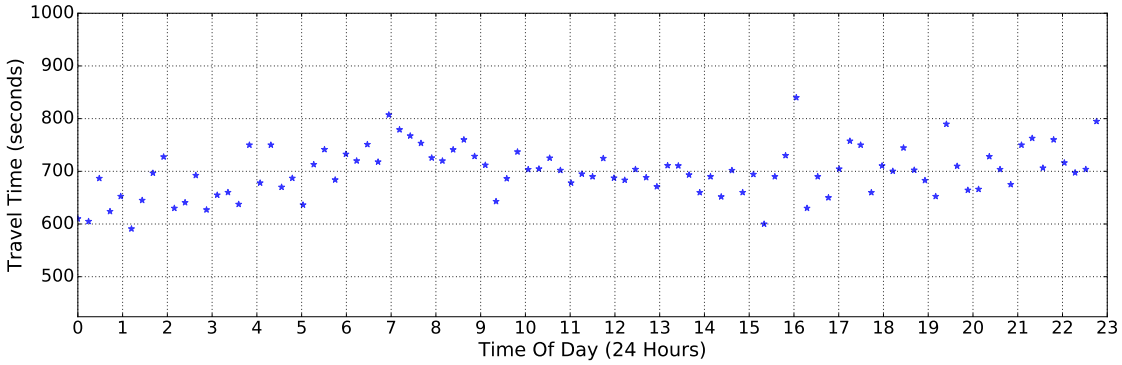
Definition 4 ***Departure travel time*** *is the travel time of a trip given the departure time. Individual departure travel time $TT_i^d(t)$ is the individual travel time with departure time instance t. Aggregated departure travel time $TT^d(p)$ is the aggregated travel time with departure time within interval p.*

Definition 5 ***Arrival travel time*** *is the travel time of a trip given the arrival time. Individual departure travel time $TT_i^a(t)$ is the individual travel time with arrival time instance t. Aggregated arrival travel time $TT^a(p)$ is the aggregated travel time with arrival time within interval p.*

Each vehicle that traverses on a road of interest with finite length has both departure and arrival travel time. For each vehicle i, $TT_i^d(t) = TT_i^a(t) = t' - t$. However, travel time usually differs from vehicle to vehicle. For the same vehicle, travel time also can be different from day to day even with a same departure or arrival time. Aggregated travel time mainly depends on the data. It is sensitive to outliers. Sometimes, median travel time is used to obtain aggregated travel time instead of mean as median is more robust to outliers. Figure 2.1 shows the example of individual and aggregated travel time for freight vehicles. Large variations can be observed in the plots of individual travel times, but such information is not provided when looking at aggregated travel time.



(a) Individual Departure Travel Time



(b) Aggregated Departure Travel Time

Figure 2.1: The example of individual and aggregated departure travel times of freight vehicles. For aggregated departure travel time, the travel times are aggregated every 15 minutes.

2.3 Travel Time Observation, Estimation and Prediction

Travel time information can be obtained via direct observation, estimation or prediction. This section gives the introduction of these approaches.

2.3.1 Travel Time Observation

Direct observation of travel time requires both departure time and arrival time. Arrival time can be obtained using site based automatic vehicle detectors. In most cases, travellers are interested in travel time with departure time rather than arrival

time. In this case, departure time should be estimated. Given vehicle i, we can use the arrival time t' and the observed arrival travel time $TT_i(t')$ to estimate the departure time:

$$t = t' - TT_i(t')$$

The estimated departure time should have the same format as the arrival time. The latest camera systems are able to detect both the entry and exit time of a road of interest. In this case, departure time, arrival time and travel time can be directly observed. The above approach may not be valid in the case of aggregated travel time. This is simple because vehicles that arrive within the same arrival time interval may not depart within the same departure time interval. In this case, the estimation of aggregated departure time is more complicated.

2.3.2 Travel Time Estimation

Travel time estimation is the calculation of travel time for the occurred (i.e., up to the present time instance or time interval) traffic conditions when direct observation of travel time is not available. The calculation is based on historical traffic or travel time data other than departure and arrival time instances. Simple estimation approaches are usually based on the assumption of vehicle length and the traffic flow theory. More complicated estimation approaches can be based on statistical learning techniques such as maximum likelihood estimation and statistical sampling. Travel time estimation reveals the past traffic conditions. Thus, it is mainly used to evaluate the performance of a road network.

2.3.3 Travel Time Prediction

Travel time prediction is the calculation of travel time for future traffic conditions. The calculation is usually based on both historical traffic or travel time data and the assumption of future traffic conditions. A common assumption for travel time prediction is that the future traffic conditions are more likely to be similar to the

recent past traffic conditions given the same departure time and day-of-week. This assumption is not robust to the variation of road capacity caused by non-recurring events such as incidents or road works. Hence, real-time traffic flow data is usually incorporated to enhance the prediction. Travel time prediction reveals the future traffic conditions. Thus, it is more useful for travellers to make better travel plans. It is also more useful for public and private transportation service providers to make better task planning. Note that travel time prediction is usually referred to as departure travel time prediction unless explicitly stated.

Instantaneous travel time is a naïve method for travel time prediction. The definition is given as follows [200]:

Definition 6 ***Instantaneous travel time*** *is the calculation of travel time that a vehicle would traverse a road of interest with departure time instance t within departure time interval p if the traffic conditions for any intervals $p^* \geq p$ remain stationary.*

Note that there is no overlap between p and p^* in this case. Instantaneous travel time is widely used in practice because it is very easy to implement. The main problem of instantaneous travel time is that the assumption of stationary future traffic conditions is too simple that travel times can be underestimated as congestion sets in and overestimated as congestion dissolves [200].

2.4 Decomposition of Travel Time

In general, the travel time $TT(t)$ can be decomposed as:

$$TT(t) = TT_{free} + D(t)$$

where TT_{free} is the expected (or mean) travel time under free-flow condition; $D(t)$ is the estimation of the total delay with departure time t. $TT_{free} = L/v_{free}$ where

L is the length of the road of interest; and v_{free} is the '*free*' (desired) speed which is commonly treated as a constant value. Note that there is always an uncertainty of the delay estimation. If such uncertainty is modelled as a random variable, $TT(t)$ then becomes a random variable.

Delay estimation or prediction can be hard for urban or arterial roads as there are more types of delay involved such as the delay of passing through signalised intersections, the delay caused by signal control strategies, the delay caused by offsets between adjacent signals, and the delay caused by overflow queue. Many early studies [214, 4, 135, 165] tend to explicitly model the relation between delay and some influencing factors such as flow rate, road capacity, traffic light cycle, effective green light cycle and queuing behaviours caused by congestion. These model-based approaches require a lot of assumptions which can be inaccurate. Hellinga et al.[76] proposed a link based approach to address the difficulty of travel time estimation. In their study, the target road is first divided into a set of consecutive links by traffic intersections; then the delay at link level can be estimated. Because we are interested in modelling the relation between latent influencing factors and corresponding travel time endogenously, explicitly modelling of link level delay is out of the scope of this thesis. For detailed discussion of link level delay estimation, please refer to [76]. Note that although this link based approach makes the travel time estimation of the road of interest relatively easier, it may yield larger estimation error due to the large variation of road travel time at link level [28].

2.5 Travel Time Variability

Travel time can be decomposed into free-flow travel time and delay. Since free-flow travel time is a constant value, the variation of travel time thus is mainly due to the different experiences of delay.

Definition 7 ***Travel time variability*** *is the degree of variation of travel time mainly due to the different experiences of delay.*

Delay can be categorised into recurrent delay and non-recurrent delay. Recurrent delay arises from fluctuations in traffic demand, the manner in which a road network is operated, as well as the physical layout of the road network. Thus, it is systematic [49]. Time-of-day and day-to-day travel time variability are the results of recurrent delay. They can be learned using historical data. Non-recurrent delay is caused by infrequent events (e.g., incidents, road works, public events) which is hard to foresee. In this case, the travel time variability is hard to learn.

2.5.1 Travel Time Variability Measures

There are various ways to measure travel time variability. Standard deviation (SD) is one of the most widely used measures in the early travel time variability study [107, 149, 26]. It is a popular statistical measure of the variation, given a data sample. It can be expressed as:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

where N is the sample size; y_i is the ith travel time observation; $\bar{y}$ is the sample mean. $N-1$ corresponds to the number of degrees of freedom in the data. The convention "$\frac{1}{N-1}$" provides an unbiased estimate of the true population variance. Coefficient of variation (CV) is another popular measure which can be expressed as [124]:

$$\widehat{c_v} = \frac{s}{\bar{y}}$$

CV is a ratio value thus it is dimensionless. Both SD and CV are not robust to outliers. They can not reveal the asymmetry of a travel time distribution. Thus they can not be used if the given travel time distribution is significantly statistically skewed.

Interdecile range (IDR) and interquartile range (IQR) are the two popular range

measures [106, 203]. They can be expressed as:

$$r_{IDR} = p_{90} - p_{10}$$

$$r_{IQR} = p_{75} - p_{25}$$

where p_{90}, p_{75}, p_{25} and p_{10} denote the 90th, 75th, 25th and 10th percentile travel times, respectively. Buffer index (BI) is another type of range measure which indicates the extra *buffer time* needed for travellers to be on time 95% of the time [124]. It can be expressed as:

$$r_{BI} = \frac{p_{95} - \bar{y}}{\bar{y}}$$

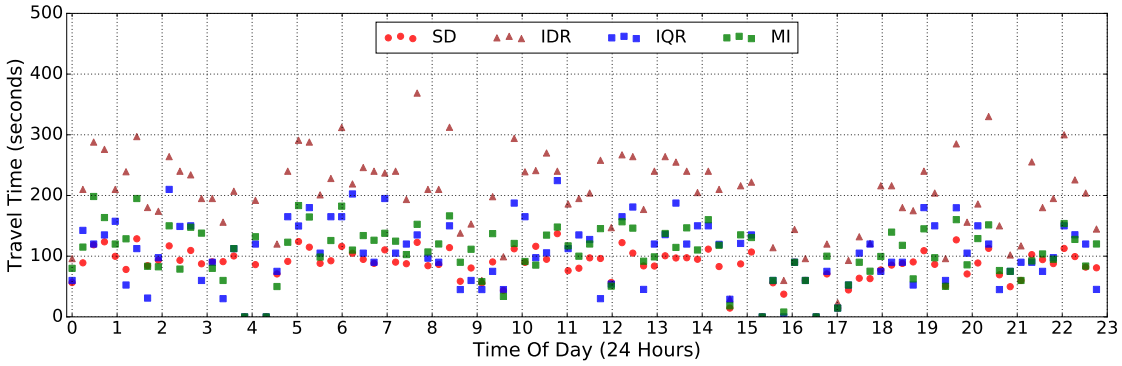
where p_{95} denotes the 95th percentile travel time. Other percentiles (e.g., 85th, 90th, 99th) also can be used for BI, depends on the level of desired reliability for the measure. BI relies on mean, thus it is also not robust to outliers. Misery index (MI) tends to capture the difference between the mean and the mean of 20% worst trips. It is a useful measure if the travel times of the 20% worst trips are much larger than the mean.

van Lint et al. [203] proposed the width index (WI) to measure travel time variability. WI can be expressed as:

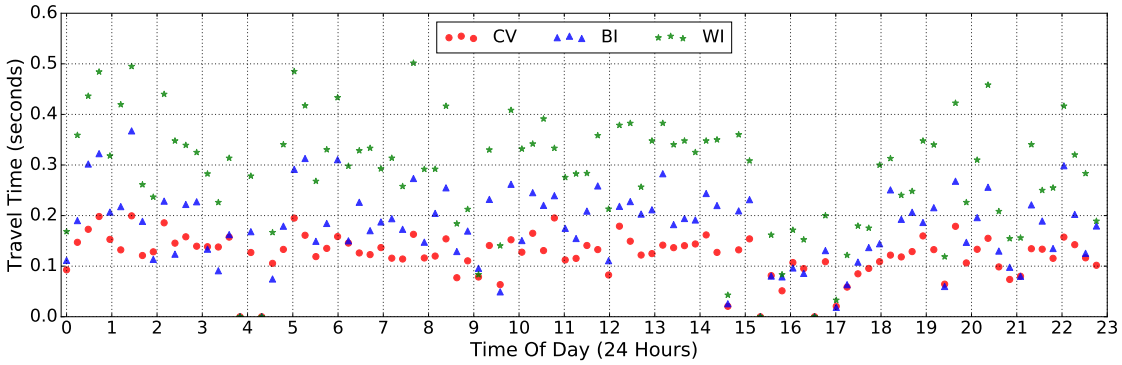
$$r_{WI} = \frac{p_{90} - p_{10}}{p_{50}}$$

where p_{50} is the 50th travel time percentile which is also the sample median. WI is considered more robust to outliers as it uses the median instead of the mean.

The above measures can be grouped into absolute measures (SD, IDR, IQR, MI) and ratio measures (CV, BI, WI). Figure 2.2 shows the example of travel time variability measures using these measures where Figure 2.2a shows the absolute measures and Figure 2.2b shows the ratio measures. It can be seen that travel time variability can be interpreted very differently using different measures. Although these measures are able to reveal the variation of travel time to some extent, it is



(a) Absolute measures (SD, IDR, IQR, MI)



(b) Ratio measures (CV, BI, WI)

Figure 2.2: The example of travel time variability measures for freight transportation. The target road is selected from the intermodal drayage road network of the Port of Ningbo-Zhoushan, China (CNZOS).

not good enough to generalise traffic conditions. For example, very small travel time variability can be found both under free-flow conditions and heavily congested conditions; very large travel time variability can be found both when congestion sets in and dissolves. Also, small size sample can bias the interpretation of travel time variability.

2.5.2 The General Categorisation of Travel Time Variability

2.6 Trajectory Data and Speed

Travel time variability can be generally categorised into two types: vehicle-to-vehicle variability and same-condition variability [120]:

- *Vehicle-to-vehicle* travel time variability refers to the variability of individual travel times within the same departure time interval. It can be influenced by both individual factors (e.g. driver's driving courtesy, vehicle type, load, emergency) and situational factors (e.g., time-of-day, day-of-week, weather, incident). Collecting data for individual factors can be difficult which makes vehicle-to-vehicle variability analysis very challenging. In real-world cases, travel time can be different for the same driver under the same situation (e.g., same departure time interval, same day-of-week, same weather). Figure 2.3 shows the example of individual travel times of four container trucks within the same departure time interval (15 minutes per interval). It can be seen that travel time is not only different by vehicle, but also different for the same vehicle.

- *Same-condition* travel time variability refers to the variability of individual

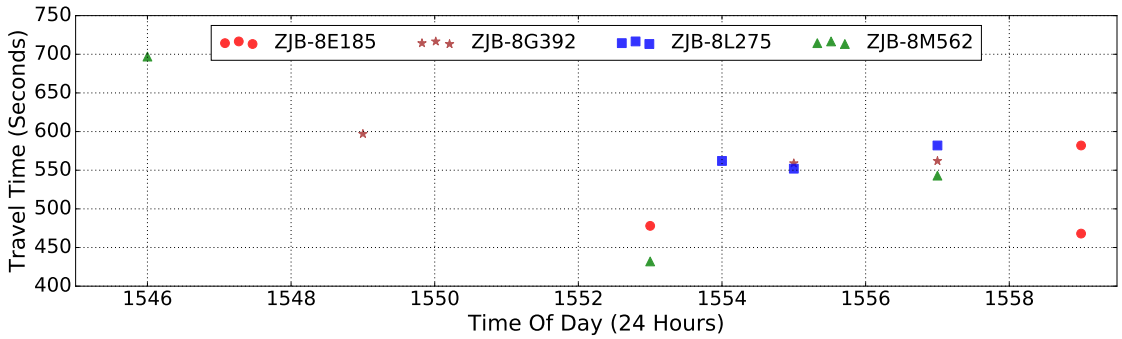


Figure 2.3: The example of individual travel times of four container trucks. Departure time interval size: 15 minutes.

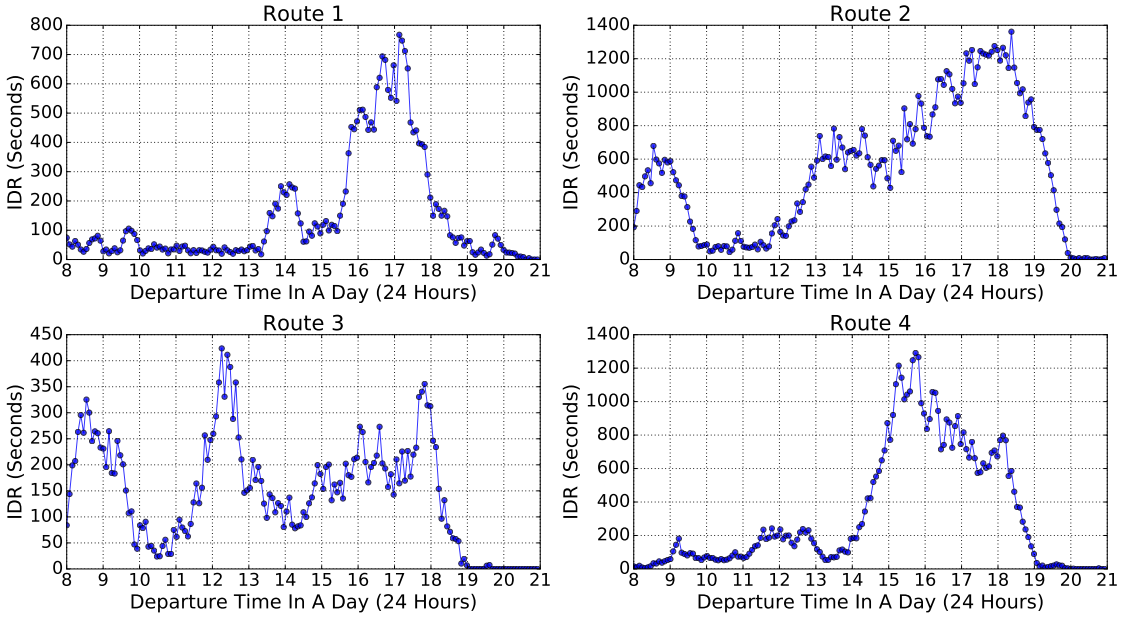


Figure 2.4: The example of variability measure of aggregated travel times on four different highway road segments under the same weather conditions. Travel times are aggregated every 5 minutes. Travel time variability measure: IDR.

travel times under a same condition (e.g. time-of-day, day-of-week, traffic conditions, weather). Same-condition variability can be examined for both individual and aggregated travel time data. Figure 2.4 shows the example of variability measure of aggregated travel times on four different highway road segments under the same weather conditions. It can be seen that the variability measure is very different from road to road even under the same time-of-day, day-of-week and weather conditions.

There are various ways to collect data for travel time prediction. One way is to use stationary sensors such as loop detectors, transponders, cameras to collect traffic flow data (e.g., vehicle counts, occupancies). Since direct observation of travel time is not available (except for dual loop detectors), travel times should be estimated first using the data. The quality of travel time estimation can be significantly influenced by the deployment density and the locations of the sensors. Another approach is using site based automatic vehicle detectors. In this case, direct observation of

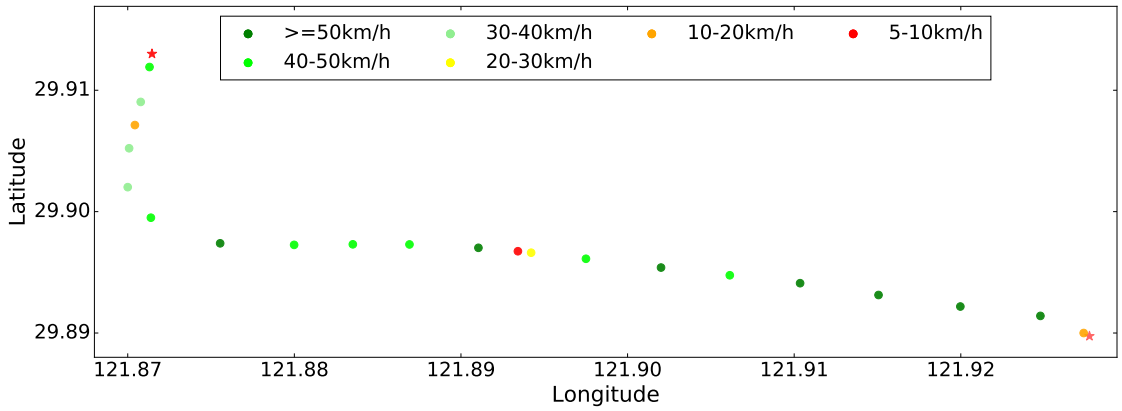


Figure 2.5: The trajectory data of a container truck in the intermodal drayage road network of CNZOS.

individual travel times becomes available. In the recent decade, trajectory data of probe (or floating) vehicles becomes a popular choice mainly because of two reasons: (1) road side sensor data is usually owned by traffic management authorities which may not be accessible to researchers and private transportation service providers; (2) GPS devices have been widely equipped on vehicles which makes real-time observation of travel time and speed available. Trajectory data sampling rate (i.e., the time interval between two consecutive trajectory data reports) can affect the quality of travel time estimation and prediction. Many studies [159, 76, 138] have shown that higher sampling rate (i.e., $\leq$ 10 seconds) is beneficial to travel time estimation and prediction. Figure 2.5 shows the trajectory data of a container truck in the intermodal drayage road network of CNZOS. The sampling rate is 30 seconds. The colour of a trajectory data point indicates the level of the corresponding instantaneous speed. That is, the redder the colour is, the faster the speed is.

2.6.1 Instantaneous Speed

Instantaneous speed is usually included in a trajectory report. The conglomeration of low instantaneous speeds from multiple trips may indicate that traffic flow is interrupted at the conglomeration area. Figure 2.6 shows the trajectory data of

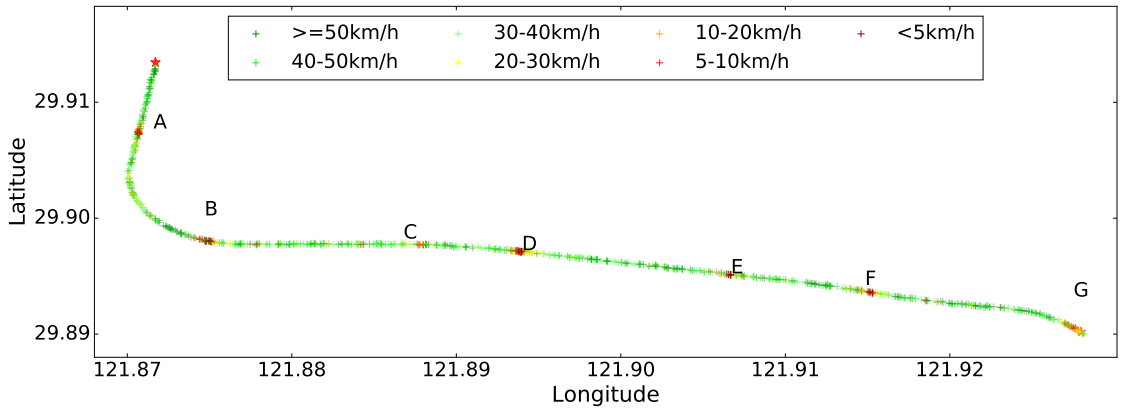
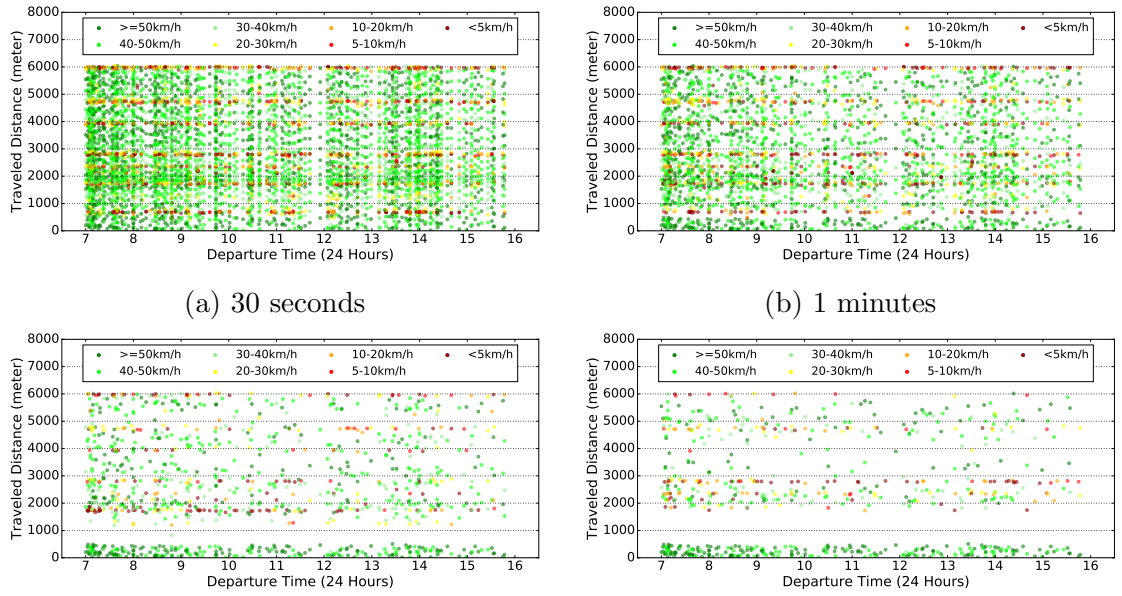


Figure 2.6: The trajectory data of 50 randomly selected trips of a road segment selected from the intermodal drayage road network of CNZOS. The trajectory data sampling rate is 30 seconds.



(a) 30 seconds

(b) 1 minutes

(c) 3 minutes

(d) 5 minutes

Figure 2.7: The time-space scattered plot of the instantaneous speeds of 500 randomly selected trips of a road segment selected from the intermodal drayage road network of CNZOS.

50 randomly selected trips of a road segment selected from the intermodal drayage road network of CNZOS. The red stars indicate the start and end points of the road segment. There are six conglomeration areas in the figure, labelled from A to G. The field study shows that these six conglomeration areas are signalised and

non-signalised intersections. That indicates that the delay is mainly the delay at the intersection. In this case, travel time variability can be studied by exploring these conglomeration areas given different departure time intervals. Note that instantaneous speeds may not be able to 'tell a good story' if the sampling rate is too low. As is shown in Figure 2.7a, the delay at some intersections become unobservable when the sampling rate becomes 3 minutes or higher.

2.6.2 Trajectory Error and Mean Speed Estimation

The mean speed can be estimated as:

$$u \approx \frac{L}{t_P - t_0} = \frac{L}{TT_L}$$

where L is the length of a road of interest; t_0 and t_P are the interpolated times associated with the entrance and exit of the road perspectively; TT_L is the travel time.

Because each trajectory data has a positional error ε associated with it, which yields error when interpolating time. This error then can translate into the computational error when estimating u. Quiroga et al. used error propagation theory [56] to the computational error ε_u [159]:

$$\varepsilon_u \approx \frac{\sqrt{2}\varepsilon}{t_P - t_0} = \frac{\sqrt{2}\varepsilon}{TT_L}$$

Note that the above estimation is based on the assumption that there is a linear interpolation scheme for computing t_0 and t_P. Since instantaneous speeds are usually provided in trajectory data, Quiroga et al. suggested to use them to estimate the mean speed:

$$u \approx \frac{1}{t_P - t_0}\left\{v_0\left(\frac{t_1 - t_0}{2}\right) + \left[\sum_{k=1}^{P-1} v_k\left(\frac{t_{k+1} - t_{k-1}}{2}\right)\right] + v_P\left(\frac{t_P - t_{P-1}}{2}\right)\right\}$$

Based on Equation (2.6.2), ε_u can be estimated as [159]:

$$\varepsilon_u \approx \frac{\sqrt{P-0.5}}{P}\varepsilon_v \approx \frac{\sqrt{\Delta T(TT_L - 0.5\Delta T)}}{TT_L}\varepsilon_v$$

where ε_v is the error in speed associated with individual trajectory points and P is the trajectory points count with the road of interest (in addition to P_0). Note that wether to use Equation (2.6.2) or Equation (2.6.2) for mean speed calculation depends on the GPS device positional error ε and the error associated with individual instantaneous speeds ε_v. To apply Equation (2.6.2), we need to determine which trajectory points should be associated with the target road. That can be done using e.g., geographic information system (GIS) neighbourhood functions. In this thesis, we are not able to get information about ε and ε_v, thus Equation (2.6.2) will be used for mean speed calculation if needed. Note that it is always recommended to take the trajectory error (i.e. positional error and instantaneous speed error) into account for mean speed estimation if they are available. For the more detailed discussion of travel time analysis under the effect of trajectory error, please refer to [159].

2.7 Summary

This chapter starts with the introduction of some fundamental elements for travel time prediction including the definitions of travel times, travel time observation, travel time estimation, and travel time prediction.

Travel time can be decomposed into free-flow travel time and delay. Free-flow travel time is often treated as a constant value. Delay can be categorised into: recurrent delay and non-recurrent delay. Recurrent delay arises from fluctuations in traffic demand, the manner in which a road network is operated, as well as the physical layout of the road network. Thus, it is systematic. Recurrent delay makes time-of-day and day-to-day travel time variability, which can be learned using his-

torical data. Non-recurrent delay is caused by infrequent events such as incidents, road works, public events. It is hard to foresee thus the resulted variability is hard to learn using historical data. Estimating delay involves the estimation of traffic conditions. Various types of travel time variability measures are introduced, but none of these commonly used measures can directly generalise traffic conditions accurately. It is always necessary to incorporate other types of data analysis techniques to gain a bigger and better picture of traffic conditions.

Trajectory data is considered as an alternative data source for travel time prediction when traffic flow data is not available. With trajectory data, direct real-time observation of speeds and travel times become available. That can be very helpful when we want to incorporate real-time data into the model. Instantaneous speeds are usually included in trajectory data. They can not only be used to identify delay phenomenon but also be used to estimate mean speed. Note that there are errors (i.e., GPS positional error and instantaneous speed estimation error) associated with individual trajectory data points. They can negatively affect the quality of mean speed estimation and travel time estimation. It is recommended to take them into account for mean speed or travel time estimation if they can be obtained (from GPS device manufacturer).

In the next chapter, we provide a comprehensive review of state-of-the-art travel time prediction methods.

Chapter 3

Literature Review

3.1 Introduction

Naïve methods such as instantaneous travel time, historical mean/median are the simplest travel time prediction methods. They have no additional assumptions other than the data used and simple formulas (e.g., travel time = distance / speed) [202]. Naïve methods are still widely used nowadays because they are easy to implement and have very low computational cost. However, they are too simple to model the complex travel time variability in real-world cases, and the performances are usually poor. Nowadays, naïve methods are mainly used as baseline methods or integrated into more advanced methods.

Except for naïve methods, most of the methods in the literature can be categorised into: model based methods and data-driven methods. Model based methods require pre-defined models that address the physical traffic processes. These models can be analytical models (e.g., delay models, queuing models) or traffic simulation models (e.g., macroscopic models, mesoscopic models, microscopic models). Data-driven methods do not require pre-defined models. The models can be learned using data. Data-driven methods are easy to implement, and the performances are usually good in practice.

3.2 Model Based Methods

There are mainly two model based approaches: analytical models and traffic simulation models. The rest of the section gives a brief introduction of these two approaches. Their advantages and disadvantages are also discussed.

3.2.1 Analytical Models

There are two popular types of analytical models: delay models and sandglass queuing models. Many later-day analytical models are derived from these two type of models.

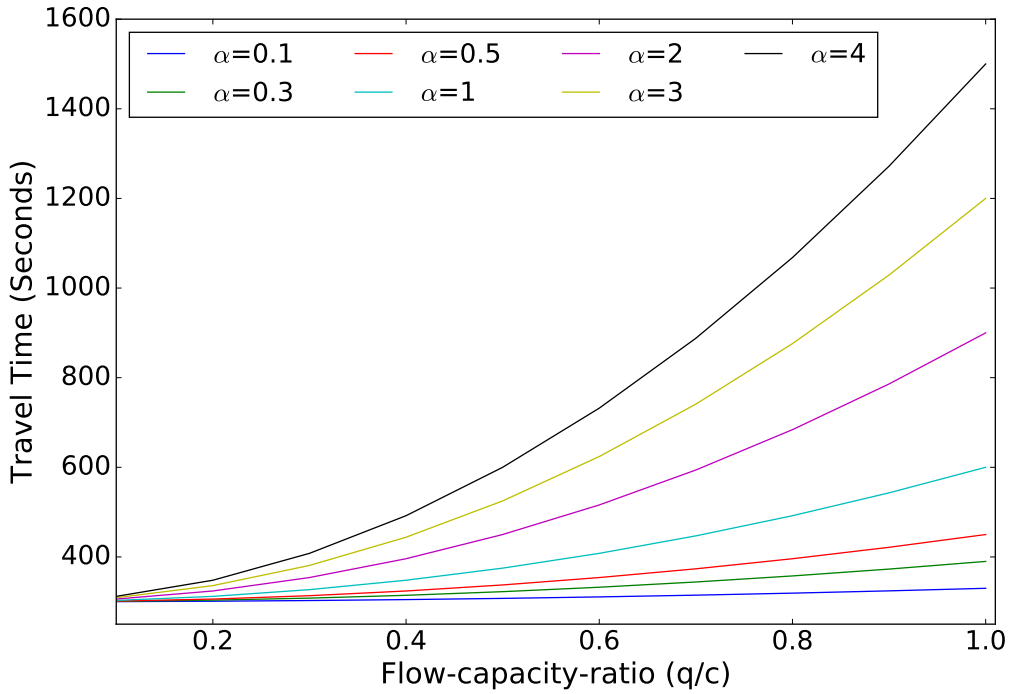


Figure 3.1: The BPR functions with different values of α. $TT_{free} = 300$, $\beta = 2$.

Delay Models

The US Bureau of Public Roads (BPR) functions is one type of widely used delay models. It can be generally expressed as:

$$TT = TT_{free}[1 + \alpha \cdot \chi^{\beta}] \tag{3.1}$$

where $\chi = v/c$ is the flow-capacity-ratio; q is the traffic flow rate, or the vehicle arrival rate (vehicles/hour); c is the capacity of intersection approach (vehicles/hour). When $\chi = 1$, the traffic condition is considered saturated. α and β are the model parameters that can be learned offline using historical data. As is shown in Figure 3.1, travel time is the function of flow-capacity-ratio. Since TT_{free} is usually treated as a constant value, BPR functions thus focus on the delay function $\alpha(\chi)^{\beta}$.

Webster [214] proposed a *steady-state stochastic model* to estimate delay at signalised intersections. The model can be expressed as [214]:

$$D = \frac{C(1-\lambda)^2}{2(1-\frac{1}{\lambda}\chi)} + \frac{\chi^2}{2q(1-\chi)} - 0.65(\frac{c}{q^2})^{\frac{1}{3}}\chi^{2+5\lambda} \tag{3.2}$$

where D is the average delay per vehicle (seconds/vehicle); $\lambda = g/C$ is the green light cycle ratio; g is the effective green light interval (seconds); C is the traffic signal cycle length (seconds). The first term is the average delay assuming uniform arrival rate. The second term is the additional delay attributed to the randomness of vehicle arrivals. The third term is the correction term for specific field conditions [36]. Many other stochastic models have since been developed by modifying Webster's model. The main problem of these steady-state stochastic models is that the estimated delay becomes infinity as χ approaches 1. Thus, these models can only be used when traffic condition is under-saturated (i.e., $\chi < 1$).

To address the limitation of steady-state stochastic delay models, *time-dependent stochastic models* were developed [4, 135, 44]. The general concept of time-dependent delay models is originally devised by Robertson [165] and improved by Kimber and

Hollis [101] using coordinate transformation technique [36]. These time-dependent stochastic delay models have been incorporated into numerous of capacity guides of many countries including the United States [1], Canada [187], and Australia [5]. These models are also known as *the capacity guide delay models*, which can be expressed as follows [36]:

$$D = d_1 \cdot f_{PF} + d_2 + d_3 \cdot f_r \tag{3.3}$$

with

$$d_1 = 0.5C\frac{(1-\lambda)^2}{1-\frac{1}{\lambda}\cdot\min(\chi,1)} \tag{3.4}$$

$$d_2 = 900\chi^n T\left[(\chi-1)+\sqrt{(\chi-1)^2+\frac{mkI}{cT}\cdot(\chi-\chi_0)}\right] \tag{3.5}$$

$$f_{PF} = \frac{(1-\zeta)f_p}{1-\lambda} \tag{3.6}$$

where d_1 is the uniform delay (seconds/vehicle); d_2 is the additional delay accounting for randomness of vehicle arrivals and over-saturation delay (seconds/vehicle); d_3 is the residual delay for over-saturation queues that may have occurred before the evaluation period (seconds/vehicle); f_{PF} is the scale factor accounting for the quality of progression in coordinated systems; f_r is the scale factor for d_3; f_p is the scale factor for situations in which the platoon arrives during the green light interval (0.9 - 1.2); T is the evaluation period (hours); n, m are the capacity guide model parameters; k denotes the incremental delay factor accounting for pre-timed or actuated signal controller settings; I denotes the adjustment factor for upstream filtering/metering; ζ denotes the proportion of vehicles arriving during effective green light interval g; and χ_0 denote the flow rate below which the overflow delay is negligible. The discussion of the configuration of these parameters can be found in [36].

There are some limitations with the above stochastic delay models: (1) they assume steady-state traffic conditions; (2) they assume the flow rate follows a Poisson

distribution that remains constant over time; (3) they assume the headways between departures have a known distribution with a constant mean value; (4) and they do not include all of the delay caused by arriving or discharging vehicles during traffic control periods [36]. Besides, these models are sensitive to the configuration of model parameters. That is, a small change of some parameters can have a significant impact on the model performance. Overall, these stochastic delay models are not good enough to capture the complexity of queueing behaviours at signalised intersections.

Sandglass Queuing Model Approach

Sandglass queuing models is another type of widely used analytical model. It is called *sandglass queuing model* because the way it models queue discharging is very similar to the process that sand is flowing through an hourglass. The general form of the sandglass queuing model which can be expressed as follows [194]:

$$TT = \frac{L - L_{queue}}{v_{free}} + \frac{N_{queue}}{q} \tag{3.7}$$

where L denotes the length of the target link; L_{queue} is the estimated queue length; and v_{free} is the free-flow speed. The first term is the time spent on non-congested part of the link. The second term is the time spent for queuing. Takaba et al. [185] further enhanced the model by introducing the jam density k_{jam}. Thus the model becomes:

$$TT = \frac{L - L_{queue}}{v_{free}} + \left[k_{jam} \cdot \frac{L_{queue}}{q} - L_{queue}(\frac{k_{jam}}{s_a} - \frac{1}{v_{queue}}) \right] \tag{3.8}$$

where k_{jam} denotes the jam density; v_{queue} is the running speed in queue.

There are also some limitations with the sandglass queuing models: (1) they are not applicable for heavy congested arterial roads; (2) they require data collection for queue length estimation which is often hard in real-world cases [232]. Similar to stochastic delay models, these queuing models are also sensitive to the configuration

of model parameters. Inappropriate parameter configuration can result in a poor model performance [8, 202].

3.2.2 Traffic Simulation Models

Traffic simulation models are widely used to help develop and verify traffic and travel time prediction models. It is mainly because running a simulation experiment is generally much cheaper than setting up a test bed in the real world. Traffic simulation models can be classified into three types in terms of *the level of details* with which they represent the traffic systems [85]: macroscopic models, mesoscopic models, microscopic models (including submicroscopic models).

Macroscopic Traffic Simulation Models

Macroscopic traffic simulation models have low-level details of the traffic systems they represent. They simulate traffic at a high-level of aggregation as a flow without distinguishing its constituent parts. The traffic stream is usually represented as flow-rate, density and/or speed [85]. For example, given a simulation time instance t, a user can only obtain the flow rate, density and/or mean speed of the traffic stream at t and there are no details of vehicles and their behaviours such as lane changing, over-taking, queuing, turning etc. Early macroscopic traffic simulation models are mainly based on the wave theory [117]. Second-order traffic models were later proposed and extended [153, 152, 34]. Delay models and queuing models can be integrated into microscopic traffic simulation models to increase the fidelity of the models. Macroscopic traffic simulation models are easy to implement and the computational costs are usually low due to the low-level complexity. Note that macroscopic traffic simulation models are often used to estimate traffic states together with some filtering techniques such as Kalman filtering so that real-world traffic or travel time observations can be incorporated. However, since they do not distinguish individual vehicles, they can only be used for aggregated travel time

prediction.

Mesoscopic Traffic Simulation Models

Mesoscopic traffic simulation models have medium-level details of the traffic systems they represent. Traffic is represented by (small) groups of entities where the details of their activities and interactions are described at low-level. For example, lane-changing might be modelled as an instantaneous event based on, e.g. relative lane densities and speed differentials [85]. Mesoscopic traffic simulation models are more flexible on capturing traffic dynamics and modelling important traffic behaviours such as queuing, merging and weaving when compared with macroscopic traffic simulation models. They are also more flexible on handling route selection [105]. Hoogendoorn et al. summarised that there are three well known types of mesoscopic traffic simulation models: headway distribution models [20, 83], cluster models and gas-kinetic continuum models [156, 157, 84]. Mesoscopic traffic simulation models are a bit more complicated than macroscopic traffic simulation models. They require more governing parameters thus have higher computational costs.

Microscopic Traffic Simulation Models

Microscopic traffic simulation models have the highest details of the traffic systems among the three types of traffic simulation models. They describe both time-space behaviours of entities (i.e. vehicles and drivers) and present the details of their interactions at high-level. For example, lane-changing of a vehicle is modelled as a series of actions made by the vehicle including evaluating the feasibility of lane-changing, performing lane-changing and reducing speed to keep safe distance gap if there is a leading vehicle. The responses of the surrounding vehicles also can be modelled. Submicroscopic traffic simulation models is a special type of microscopic traffic simulation model. It describes the details of time-space behaviours of entities at even higher level. By far there is no clear boundary between general microscopic and submicroscopic traffic simulation models. In microscopic traffic simulation mod-

els, vehicles can be distinguished by vehicle type (e.g., car, bus, truck) and driver's characteristics (e.g. age, driving age, traffic criminal records). Thus, vehicle's behaviours are not only influenced by situational factors such as surrounding traffic, weather but also influenced by its vehicle type and driver's characteristics [116].

Early microscopic traffic simulation models are mainly based on car-following models in which the main concept is avoiding collisions between vehicles. Different types of car-following models have been proposed since 1950s including safe-distance models [155, 48, 92], stimulus-response models [24, 58, 147, 134] and psycho-spacing models [111, 82]. A review of car-following models can be found in [86]. Celluar Automaton (CA) models (or particle hopping models) are also widely used to implement microscopic traffic simulation models. CA models describe the traffic system as a lattice of cells of equal size (typical cell length is 7.5 meters) [144]. They are discrete simulation models where a vehicle *hops* from one cell to the next (downstream) cell during one simulation time step. Because they are easy to implement and fast to run. A lot of CA based microscopic traffic models have been developed [164, 43, 207, 69, 15] for real-world applications. Microscopic traffic simulation models use more assumptions simply because not all the model parameters can be set based on real-world data. Thus, they are more difficult to calibrate than macroscopic and mesoscopic traffic simulation models. Because they are much more complicated, the computational costs are usually much larger than that for the other two types.

Summary of Traffic Simulation Models

Traffic simulation models usually require origin-destination matrices or route selection models. The accuracy of origin-destination matrices and route selection models have a significant impact on the model accuracy. Traffic simulation models are able to explicitly interpret the physical traffic process and examine the effects of influencing factors of travel time in isolation. However, they require high degree of expertise for design, development and maintenance. They are more complicated than analytical models described earlier and relatively more difficult to implement for real-time

applications [120]. For microscopic traffic simulation models, the computational costs are usually very large.

3.3 Data Driven Methods

A large number of data-driven methods have been developed for travel time prediction in the recent decade mainly due to their superior performance. They do not need pre-defined models and models can be learned only using data. In data-driven methods, solving travel time prediction problem is equivalent to solving the following regression problem:

$$y = \mathbf{f}(\mathbf{x}, \mathcal{D}, \boldsymbol{\theta}) + \epsilon \tag{3.9}$$

where $\mathcal{D}$ denotes the data that is comprised of N input vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ of dimension d ($d \geq 1$) and a corresponding output vector $\boldsymbol{y} = \{y_i\}_{i=1}^{N}$; $\mathbf{X} \in \Re$; $\mathbf{y} \in \Re$; $\mathbf{f}$ is the approximation of travel time function; $\boldsymbol{\theta}$ is the model hyperparameters that can be fitted using $\mathcal{D}$; ϵ is a noise process that represents factors that are unobservable. If $\mathbf{x}$ is an input vector in $\mathcal{D}$, the above form becomes in-sample prediction; if $\mathbf{x}$ is not in $\mathcal{D}$, it becomes out-of-sample prediction.

3.3.1 Categorisation

Some researchers classified data-driven methods into *parametric* and *non-parametric* methods based on whether the model is parametric or non-parametric. By definitions, a parametric model has a fixed number of parameters, and the model structure is specified a prior. A non-parametric model grows the number of parameters with the amount of training data, and the model structure is not specified a prior but determined using data [141, p. 16]. That binary categorisation of data-driven methods is clearly incorrect. For example, as is shown in Table 3.1, recognising time series models as parametric methods without explicitly listing the algorithms and

techniques is inaccurate because there are also non-parametric time series analysis techniques including non-parametric spectral analysis, non-parametric smoothing, non-parametric kernel estimation, nonlinear additive autoregression models etc. Also, whether a Kalman filtering based method is parametric or non-parametric depends on the linearity of the state (or process) and the observation equations used in the method. If the state and the observation equations are nonlinear, one can use non-parametric extended Kalman filter to approximate the true belief by a Gaussian distribution. To avoid making any conceptual ambiguity and vagueness, we summarised data-driven methods into nine categories, based on the core algorithms or techniques they used: linear regression, time series analysis based approaches, Kalman filter based approaches, artificial neural networks, kernel methods, decision tree ensembles, distance based methods and hybrid approaches.

3.3.2 Linear Regression

One simple data-driven approach is using linear regression. In this approach, prediction function is assumed to be a linear combination of its covariates. Recall that $\mathcal{D}$ denotes the data that consists of N input vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ of dimension d ($d \geq 1$) and corresponding output vector $\mathbf{y} = \{y_i\}_{i=1}^N$, $\mathbf{X} \in \Re^d$, $\mathbf{y} \in \Re$. A general form of linear regression model can be expressed as:

$$y_i = \beta_0 1 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + ... + \beta_d x_{i,d} + \epsilon_i = \mathbf{x}_i^T \boldsymbol{\beta} + \epsilon_i \tag{3.10}$$

where $\boldsymbol{\beta} = \{\beta_i\}_{i=1}^d$ are the model parameters; ϵ_i is the noise. In regression analysis, y_i is known as the response variable (or dependent variable), $x_{i,1}, x_{i,2}, ..., x_{i,d}$ are known as the explanatory variables (or independent variables). The most common approach for solving linear regression problem is least squares estimation (LSE). The easiest LSE is ordinal least squares (OLS) estimation. OLS assumes ϵ_i is uncorrelated with $\mathbf{x}_i$, and it solves $\boldsymbol{\beta}$ by minimising the sum of squared residuals. This can be done by taking the partial derivate of the sum of squared residuals w.r.t $\boldsymbol{\beta}$. Equating the

Table 3.1: The existing categorisation approaches (parametric versus non-parametric) of data-driven methods for travel time prediction.

Researcher(s) (Year)	Parametric	Non-parametric
Yu et al. (2008) [230]	time series models Kalman filter	artificial neural networks nearest neighbours models traffic simulation models
Fei et al. (2011) [46]	linear regression time series models dynamic traffic assignment models Kalman filter	artificial neural networks traffic simulation models Bayesian models support vector regression
van Lint et al. (2012) [202]	analytical models traffic simulation models	linear time series models nonlinear time series models linear regression nonlinear regression artificial neural networks
Yildirimoglu et al. (2013) [226]	linear regression time series models KF	artificial neural networks support vector regression traffic simulation models
Oh et al. (2015) [150]	linear regression time series models (including Kalman filter)	support vector regression nearest neighbours models

derivative to zero, we get $\boldsymbol{\beta}$ [167]:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3.11}$$

Note that OLS assumes the noises have finite variance and are homoscedastic. This may lead to poor prediction performance in real-world cases as the noises can hardly be homoscedastic. Generalised least squares (GLS) can be used to estimate $\boldsymbol{\beta}$ if the noises are heteroscedastic but still independent from $\mathbf{X}$. GLS solves $\boldsymbol{\beta}$ by minimising the weighted sum of squared residuals, where the weights are inversely proportional to the variance of noises. For the more detailed discussion of linear

regression analysis, please refer to [39].

Kwon et al. [106] used a linear regression model to perform travel time prediction for highway roads. In their study, the explanatory variables are departure time, day-of-week, and real-time traffic flow data collected within the past ΔT time (ΔT is equivalent to the prediction horizon used in the study). They selected 8 to 10 most important explanatory variables out of 22 variables. The results show that historical data is more helpful for long-term prediction and real-time traffic flow data is more helpful for short-term (up to 20mins) prediction. They also concluded that their model is more suitable for short stretches of road (up to 15km). Zhang et al. [234] proposed a time-varying coefficient (TVC) linear regression model to predict travel time for highways. The model aims to model the relation between estimated travel time (based on current flow speed) and future travel time. The TVC model is estimated using weighted least squares (a special case of GLS) with historical travel time data. The performance of the model was evaluated using the same dataset as was used in [106]. The results show that the TVC model outperforms the model proposed in [106]. The model was also compared with a principal components analysis (PCA) based model and a nearest neighbours (NN) based model [163]. The results show that the TVC approach outperforms the PCA and NN approaches using the prediction horizon of 60 minutes. Sun et al. [182] used a locally weighted linear regression model to predict traffic flow. The results show that their weighted linear regression approach outperforms two kernel regression approaches and a k-NN approach. This mainly because real-time data is incorporated into the model while the other approaches only use historical data for the prediction. Hinsbergen et al. [196] combines a linear regression model and a locally weighted linear regression model using Bayesian framework for travel time prediction. Two combination strategies were proposed in the study: winner-take-it-all (WTIA) and weighted linear combination (WLC). With WTIA, the model with the highest evidence is selected as the predictor. With WLC, the weight on each model is assigned using the normalised evidences. The results show that the hybrid approach outperforms single model

approaches. Fei et al. [46] developed two state-space linear models using Bayesian dynamic linear model to perform short-term travel time prediction for highways. In their study, both historical and real-time data are used. The prior distribution of travel times is first specified. Then, the prior distribution is recursively updated given newly observed data. The concept of recursively updating the distribution of travel times using observable data is very similar to Kalman filter. Unlike those linear regression models reviewed earlier, the model parameters are solved using Bayesian analysis but not the traditional least squares estimations.

Linear regression models have some significant advantages: (1) it is easy to implement; (2) it is easy to incorporate with other types of models; (3) and it is fast to run. However, they can be sensitive to outliers, and careful monotonic transformation of inputs are usually needed [234]. Although linear regression models can work with multi-dimensional input data, in reality, the complex relationship between inputs and corresponding outputs can hardly be modelled in a linear approach. One treatment is projecting such multi-dimensional input data into a feature space so that linear regression analysis can be performed on this feature space. Such projection can be made efficiently using *kernel trick*, and the regression methods that utilise the kernel tick are known as the *kernel methods*. These methods are reviewed in Section 3.3.6. Another issue of linear regression models is that most of the existing linear regression models do not take feature correlation into account particularly for multivariate regression models. To address the feature correlation, full feature covariance matrix can be used. However, this can result in very high computational cost due to the inversion of the matrix if the training data size is large.

3.3.3 Time Series Analysis Approaches

A *time series* is defined as a sequence of observations taken sequentially in time [17]. In a time series, adjacent observations are correlated. Travel time data is not a typical time series because not all travel time observations are correlated, e.g.

in free-flow conditions. However, since travel time is highly correlated with traffic flow which is often considered a time series, many studies utilise time series analysis (TSA) techniques for travel time prediction. The rest of this section introduces the most widely used TSA techniques for travel time prediction.

Linear Time Series Analysis Approaches

Time series prediction involves modelling the relation between current observation y_t and past observations y_{t-i}, and the possibility of relation between y_t and the errors from the past. The first relation can be modelled using autoregression (AR) model which can be expressed as:

$$y_t = \gamma_0 + \sum_{i=1}^{p} \gamma_i y_{t-i} + e_t \tag{3.12}$$

where γ_0 is a constant; γ_p is the coefficient for the lagged variable in time $t-p$ and p is the lagging parameter. This model is sometimes abbreviated as AR(p). The second relation for the errors can be modelled using moving average (MA) model which can be expressed as:

$$y_t = \rho_0 + e_t + \sum_{i=1}^{q} \rho_i e_{t-i} \tag{3.13}$$

where ρ_0 is a constant; ρ_q is the coefficient for the lagged error in time $t-q$. This model is abbreviated as MA(q). Combining the AR(p) and MA(q) models, we get the ARMA(p,q) model:

$$y_t = [\gamma_0 + \sum_{i=1}^{p} \gamma_i y_{t-i}] + [\rho_0 + e_t + \sum_{i=1}^{q} \rho_i e_{t-i}] \tag{3.14}$$

The ARMA(p,q) model assumes the time series is generated from a stationary process in which the mean and variance do not change over time. That means such process does not have any structural trend. Because most of the real-world stochastic

processes are not stationary including traffic processes, differencing technique can be used to remove latent structural trends and make the process stationary: instead of using y_t, we use the difference in the order of d denoted as $\nabla^d y_t$, e.g., $\nabla^1 y_t = y_t - y_{t-1}$, $\nabla^2 y_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$ ... etc. The resulted model is known as the ARIMA(p, d, q) model. The selection of the right d depends on if the transformed time series (after differencing) is stationary. This can be verified using stationary test such as Dickey-Fuller test [35]. For the more detailed discussion about ARIMA models, please refer to [17].

A standard ARIMA model is essentially a regression model. It is easy to understand and implement. It is also computational tractable and requires very little storage. Early implementations of ARIMA models for traffic flow prediction can be found in [3, 148, 112] where the models are mainly the order-1 ARIMA models, abbreviated as ARIMA(p, 1, q). Williams et al. [217] argued that the first difference of a traffic flow series will not yield stationarity if a strongly cyclical nature of the traffic flow series exists. Thus they suggested using seasonal ARIMA (SARIMA) model. Many studies implemented SARIMA model or its variations for traffic flow prediction [178, 60, 22, 175]. Lee et al. [109] implemented a subset ARIMA model for traffic flow prediction in which a traffic flow series is divided into a number of subsets where each has its own coefficients and lagging parameters. The results show that their subset ARIMA model outperforms a standard ARIMA model (without pre-partition).

Williams [216] implemented a multivariate ARIMA (ARIMAX) model for traffic flow prediction. The ARIMAX model is able to take upstream traffic data as an input and then transfers it into an ARIMA model. Cools et al. [33] investigated daily traffic variability using ARIMAX, SARIMA and SARIMAX models. The results show that two ARIMAX models are better than a SARIMA model on identifying the influencing factors. Although ARIMAX models are able to handle multivariate traffic series and perform better than univariate ARIMA models, there are some concerns when implementing an ARIMAX model: (1) they may not be consistent

from time to time; (2) they are more complicated than univariate ARIMA models; (3) they are not robust to missing data; (4) and cross correlations will be changing rapidly during flow regime transitions [216]. Kamarianakis et al. [96] applied vector ARMA (VARMA) and space-time ARIMA (STARIMA) to predict traffic flow. One unique advantage of STARIMA models is that they are able to model the traffic conditions of entire road network which is difficult to do with other types of ARIMA models [97]. Tsirigotis et al. [193] compared the performance between VARMA model and ARIMAX models. Their results show that VARMA model with exogenous variables (VARMAX) performed significantly better than ARIMAX models.

Following the idea for traffic flow prediction, many studies applied ARIMA model and its variants for travel time prediction. Billings et al. [16] implemented an ARIMA model to perform one-step ahead travel time prediction for urban roads. The results show that the predicted travel time series has much smaller fluctuations than the real observations and large prediction errors can be seen in some cases especially with shorter link distance and intersections. This result may reveal that direct implementation of standard ARIMA model may not yield good prediction performance. Guin [66] implemented a SARIMA model to perform travel time prediction for highways. Suwardo et al. [183] implemented an ARIMA model for bus travel time prediction. Ruimin et al. [168] implemented an ARIMA(p,1,q) model to predict travel time for urban roads. The study results show that their ARIMA model performs better than an ANN, and the performance is better in path-based prediction than link-based prediction. Reza et al. [162] incorporated travel time series from multiple road segments into an ARIMA model to predict travel time under incident. In their ARIMA model, a cross correlation function (CCF) is used to identify the relations of travel times from different road segments. A pre-whitening technique is used to reduce the complexity of the AR process due to the application of the CCF.

Although ARIMAX models can handle multivariate traffic or travel time series,

they are more complicated and difficult to fit. Techniques such as pre-whitening are usually needed to reduce the model complexity caused by multivariate time series transformation. Although some ARIMA models are able to handle exogenous traffic factors, they are generally not robust to outliers and not able to handle mixed-type multi-dimensional inputs. They are also not good at dealing with high volatilities (i.e., increasing variance) in the data, nonlinear TSA techniques such as generalised autoregressive conditional heteroskedasticity are usually applied to address the high volatilities. More importantly, it is very difficult to perform departure travel time prediction due to the uncertainty of the availability of observations from the past. Thus, ARIMA models are considered less applicable for online predictions and predictions for long distance route [86].

Nonlinear Time Series Analysis Approaches

Traffic conditions are observed to be more volatile (i.e., increasing variance), especially when traffic is congested [96]. Such high volatility cannot be captured by ARIMA models as they assume constant variance of the errors. One solution is using autoregressive conditional heteroskedasticity (ARCH) model which allows the conditional error variance in ARMA process depending on the past squared innovations, rather than being a constant as in ARMA models with independent errors [17]. Recall the AR(p) model in Equation (3.12), the variance of e_t in ARCH models can be expressed as follows:

$$h_t \equiv Var[e_t|e_{t-1}, e_{t-2}, ...] = E[e_t^2|e_{t-1}^2, e_{t-2}^2, ...] = \omega_0 + \sum_{i=1}^{s} \omega_i e_{t-i}^2 \tag{3.15}$$

where ω_0 is a constant that $\omega_0 > 0$; ω_s is the coefficient for variance variable in time $t - s$ where s is the lagging parameter. To further generalise the ARCH model, we assume [17]:

$$e_t = \sqrt{h_t}\epsilon_t \tag{3.16}$$

where ϵ_t is randomly independent and identically distributed with zero mean and variance of 1. The resulted ARCH model is known as the generalised ARCH (GARCH) model. Since ϵ_t is independent of the past, with the GARCH(s, r) model, we can model the variance of e_t as follows [17]:

$$h_t = Var[e_t|e_{t-1}, e_{t-2}, ...] = \omega_0 + \sum_{i=1}^{s} \omega_i e_{t-i}^2 + \sum_{i=1}^{r} \zeta_i h_{t-i}^2 \tag{3.17}$$

where ζ_r is the coefficient for variance variable in time $t - r$ where r is the lagging parameter.

Driven by the successful applications of GARCH models in econometric field, some researchers used GARCH models to model volatilities in traffic and travel time data. Kamarianakis et al. [95] implemented a GARCH model to capture traffic flow volatility for urban roads. In this study, an ARIMA process is used to model the conditional mean and a GARCH process is used to model the conditional variance. The resulted model is known as the ARIMA-GARCH model. Tsekeris et al. [191] further improved the ARIMA-GARCH model in [95] by incorporating fractionally integrated components in both the conditional mean and the conditional variance equations. The resulted model is named as the ARFIMA-FIAPARCH model. Yang et al. [225] used a GARCH model to study travel time volatility for urban roads with aggregated travel time series. In their study, an ANN is first used to obtain the variance of noise, then a GARCH model is constructed based on the examination of the effect of the ARCH process for the variance of the noise. In this case, the mean equation is not restricted to ARMA models. Xia et al. [221] combined a vector AR (VAR) model and a multivariate GARCH (MGARCH) model for traffic flow prediction. The resulted VAR-MGARCH model is able to handle multivariate traffic flow series. The study results show that the VAR-MGARCH model outperforms a VAR model and an ARIMA model. Guo et al. [67] proposed a method which combines a SARIMA and a GARCH model (SARIMA-GARCH) for online traffic flow prediction. To make the SARIMA-GARCH model more suitable for online

prediction, the model is converted into a state space model which is estimated using the adaptive Kalman filter. A recent study by Zhang et al. [237] implemented a component GARCH (C-GARCH) model and a multiplicative component GARCH (MC-GARCH) model for travel time prediction for highways. The C-GARCH model decomposes a travel time series into three components: long-term, short-term and cyclical components. The MC-GARCH model decomposes the volatility into several different components. Their study results show that both the C-GARCH and MC-GARCH models outperform a standard implementation of GARCH model, for both non-peak and peak predictions.

An alternative to the GARCH model is using stochastic volatility (SV) model. With SV models, the conditional variance of traffic flow level can be modelled as a latent stochastic (low-order Markov) process. SV models are not as popular as GARCH models. That is mainly because of the difficulty in likelihood function evaluation in SV models. Tsekeris et al. [192] implemented an SV model to predict traffic volatilities. The study results show that their SV model outperforms the GARCH models in both 3 minutes ahead and 15 minutes ahead predictions. The results also show that the SV model is sensitive to the data transformation. Zhang et al. [236] combined an ARIMA model and an SV model (ARIMA-SV) to perform travel time prediction for highways. In their study, the travel times is presented as a distribution with mean and prediction interval. The study results show that the ARIMA-SV model outperforms an ARIMA-GARCH model. Zhang et al. [238] proposed a hybrid approach for traffic flow prediction for highways. In their study, the traffic flow data is decomposed into three parts: the periodic part, the deterministic part, and the volatility part. The periodic part is modelled using spectral analysis, the deterministic part modelled using an ARIMA model and the volatility part is modelled using a GJR-GARCH model [1] The results show that the hybrid approach performed better than the ARIMA-GARCH approach.

[1]GJR-GARCH model was first proposed by Lawrance R. Glosten, Ravi Jagannathan and David E. Runkle [62].

Nonlinear TSA approaches are more powerful on modelling the traffic and travel time volatilities when compared with linear TSA approaches. Thus, their performances are usually better. However, they are more complicated, the models are more challenging to fit, and it is not easy to handle exogenous input variables.

3.3.4 Kalman Filter Based Approaches

Kalman filter (KF) is over 50 years old. It is still one of the most important and most commonly used data fusion algorithms today. It is an algorithm that allows exact inference in a linear dynamic system [45]. It is a popular choice of the optimal estimator for state-space models due to its recursive properties and computational efficiency. It can be applied to both stationary and non-stationary environments. The KF assumes that the state of a system at a time t can be derived from the previous state at time $t-1$ according to [45]:

$$\mathbf{x}_t = \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \epsilon_t \tag{3.18}$$

where $\mathbf{x}_t$ is the vector of the system states at time t; $\mathbf{u}_t$ is the vector of control inputs; $\mathbf{A}_t$ is the state transition matrix which describes how the state evolves from $t-1$ to t without controls or noises; $\mathbf{B}_t$ is the control input matrix which describes how $\mathbf{u}_t$ changes from $t-1$ to t; ϵ_t is the vector of the noises for each parameter in $\mathbf{x}_t$. ϵ_t is assumed to be a multivariate Gaussian noise that $\epsilon_t \sim N(0, \mathbf{Q}_t)$. The above equation is also known as *the state model*. Note that $\mathbf{B}_t\mathbf{u}_t$ is not requisite for the state model if the state evolves to itself. The observations of the above linear system can be expressed accordingly [45]:

$$\mathbf{z}_t = \mathbf{C}_t\mathbf{x}_t + \delta_t \tag{3.19}$$

where $\mathbf{z}_t$ is the vector of the observations of the system at time t; $\mathbf{C}_t$ is the matrix that maps $\mathbf{x}_t$ to $\mathbf{z}_t$; δ_t is the vector of the observation noises. δ_t is assumed a

multivariate Gaussian noise that $\delta_t \sim N(0, \mathbf{R}_t)$. The above equation is also known as *the observation model*. $\mathbf{Q}_t$ and $\mathbf{R}_t$ are also referred to as the uncertainty involved in the state and the observation, respectively. Both the state and the observation models are linear models.

In many cases, the true state $\mathbf{x}_t$ can not be directly observed, but it can be estimated by KF. There are two steps involved for $\mathbf{x}_t$ estimation: prediction and observation update. The standard KF equations for the prediction step can be expressed as follows [45]:

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{A}_t \hat{\mathbf{x}}_{t-1} + \mathbf{B}_t \mathbf{u}_t \tag{3.20}$$

$$\mathbf{P}_{t|t-1} = \mathbf{A}_t \mathbf{P}_{t-1} \mathbf{A}_t^T + \mathbf{Q}_t \tag{3.21}$$

where $\hat{\mathbf{x}}_{t|t-1}$ is the predicted system state estimation at t; $\mathbf{P}_t$ is the predicted error covariance matrix (or the predicted uncertainty) that can be also expressed as [45]:

$$\mathbf{P}_{t|t-1} = E[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})^T] \tag{3.22}$$

The equations for the observation update step can be expressed as follows [45]:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}_t \hat{\mathbf{x}}_{t|t-1}) \tag{3.23}$$

$$\mathbf{P}_t = \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{C}_t \mathbf{P}_{t|t-1} = (\mathbf{I} - \mathbf{K}_t \mathbf{C}_t)\mathbf{P}_{t|t-1} \tag{3.24}$$

where:

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{C}_t^T (\mathbf{C}_t \mathbf{P}_{t|t-1} \mathbf{C}_t^T + \mathbf{R}_t)^{-1} \tag{3.25}$$

Here $\mathbf{K}_t$ is called the Kalman gain. It is a weighting factor that tells how much the prediction of the system can be trusted. $\mathbf{K}_t$ depends on both the predicted error covariance matrix $\mathbf{P}_{t|t-1}$ and the observation noise variance $\mathbf{R}_t$. As is shown in Equation (3.23), the system state estimation at time t is updated by adding the

predicted estimation and the correction based on the observations. The error covariance matrix at time t is updated based on the Kalman gain and the predicted error covariance matrix. Thus, the system state and the uncertainty can be recursively updated. For the more detailed information about KF, please refer to [113].

KF has low computational cost as it does not store values from the past but only '*the current belief*'. It is also easy to implement and it is an optimal estimator for linear models and Gaussian distributions. Many studies used KF for traffic and travel time predictions. Early studies that apply KF for traffic density estimation and traffic flow prediction can be found in [59, 25, 151]. Chen et al. [28] applied KF to predict travel times for highways with simulated synthetic vehicle data as the real-time observations. Chien et al. [31] re-evaluated the model developed in [28] with historical and real-time automatic vehicle identification(AVI) data. They reported that the model performance can be affected by the probe vehicle market penetration rate and network congestion level. A similar study also can be found in [32]. Nanthawichit et al. [145] incorporated KF with a macroscopic traffic simulation model to predict travel times for highways. Chen et al. [29] proposed a hybrid approach to predict bus arrival time using the data from Automatic Passenger Counter (APC) system. In their approach, an ANN is first constructed to predict travel times. Then the predicted travel times can be corrected recursively using KF by feeding the real-time bus arrival information. Their study results show that the proposed approach outperforms an ANN. Shalaby et al. [173] also applied KF to predict stop-to-stop travel time and dwell time for bus by utilising automatic vehicle location data and APC data. Yang et al. [224] applied KF to perform online travel time prediction. The results show that the performance is better with small prediction horizons, and the choices of $\mathbf{R}_t$ and $\mathbf{Q}_t$ can have a significant impact on the performance.

Note that the above studies use discrete KF. Discrete KF assumes traffic flow or travel time data is the result of a stochastic discrete-time process which can be governed using linear stochastic difference equations. However, if the relation be-

tween the process and the observation is nonlinear, nonlinear KFs such as extended KF (EKF) are preferred. In EKF, the stochastic process and the observation can be modelled using the following nonlinear functions [75]:

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{3.26}$$

$$\mathbf{z}_t = h(\mathbf{x}_t) \tag{3.27}$$

Let $\tilde{\mathbf{x}}_t$ and $\tilde{\mathbf{z}}_t$ be the approximated state and observation vectors using the functions g and f without taking noises into account. The new state and observation can be linearised using the first-order Taylor expansion [75]:

$$\mathbf{x}_t = \tilde{\mathbf{x}}_t + \mathbf{A}(\mathbf{x}_{t-1} - \hat{\mathbf{x}}_{t-1}) + \mathbf{W}\epsilon_{t-1} \tag{3.28}$$

$$\mathbf{z}_t = \tilde{\mathbf{z}}_t + \mathbf{H}(\mathbf{x}_t - \tilde{\mathbf{x}}_t) + \mathbf{V}\delta_t \tag{3.29}$$

$$\tag{3.30}$$

where $\hat{\mathbf{x}}_t$ is an posterior estimate of the state at time t; ϵ_t and δ_t are the state and observation noise processes that are assumed multivariate Gaussian with covariance matrix $\mathbf{O}_t$ and $\mathbf{R}_t$, respectively; $\mathbf{A}$ is the Jacobian matrix of partial derivatives of g w.r.t. $\mathbf{x}$; $\mathbf{H}$ is the Jacobian matrix of the partial derivatives of h w.r.t. $\mathbf{x}$; $\mathbf{W}$ is the Jacobian matrix of the partial derivatives of g w.r.t. ϵ and $\mathbf{V}$ is the Jacobian matrix of the partial derivatives of h w.r.t. δ. Similar to KF, EKF also has two steps: prediction and observation update. The prediction equations can be expressed as follows [75]:

$$\hat{\mathbf{x}}_{t|t-1} = g(\hat{\mathbf{x}}_{t-1}, \mathbf{u}_t, 0) \tag{3.31}$$

$$\mathbf{P}_{t|t-1} = \mathbf{A}_t \mathbf{P}_{t-1} \mathbf{A}_t^T + \mathbf{W}_t \mathbf{Q}_{t-1} \mathbf{W}_t^T \tag{3.32}$$

and the observation update equations are expressed as follows [75]:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{z}_t - h(\hat{\mathbf{x}}_{t|t-1}, 0)) \tag{3.33}$$

$$\mathbf{P}_t = \mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{H}_t\mathbf{P}_{t|t-1} = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_{t|t-1} \tag{3.34}$$

where [45]:

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T(\mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T + \mathbf{V}\mathbf{R}_t\mathbf{V}^T)^{-1} \tag{3.35}$$

One important feature of EKF is that $\mathbf{H}_t$ in the Kalman gain helps to propagate only the relevant components in the observation. However, if there is no mapping between the state and the observation, EKF will diverge quickly. Also, because the stochastic process here is nonlinear, there is no guarantee that the estimator is optimal. For the more detailed information about EKF, please refer to [75, 113].

Many studies can be found using EKF for traffic prediction. Gazis et al. [57] applied EKF to estimate vehicle counts for two road links where the state is the vehicle counts on individual links, and the observation is the estimated traffic speeds. Wang et al. [212] applied EKF to estimate traffic state estimation. The developed EKF based model is evaluated using a macroscopic traffic simulation model. The results show that the model performance is appealing and the model is insensitive to the initial settings of the model parameters and the related error deviations. Hinsbergen et al. [195] proposed the localised EKF algorithm (L-EKF) for traffic state estimation for large road networks. The results show that the L-EKF algorithm is much faster than the standard EKF.

Several studies can be found incorporating KF with other types of learning techniques. Shekhar et al. [175] incorporated KF with ARIMA. Lippi et al. [119] incorporated KF with SARIMA. Both studies use KF to automatically estimate and update the model parameters for the ARIMA models. Lippi et al. reported that their KF-SARIMA approach outperforms support vector regression (SVR) and

ANNs. Liu et al. [121] incorporated EKF into a state space recurrent ANN to predict travel time for urban arterial roads. In their SSNNEKF approach, the state is the weight parameters of the state space recurrent ANN, which is specified as a stationary process, and the observation is the training errors. They compared the SSNNEKF model with the KF based model developed in [32] and a state space recurrent ANN trained using Levenberg-Marquardt (SSNNLM). The results show that both the SSNNEKF model and the SSNNLM model outperform the KF based model. Although the SSNNEKF model yields slightly larger error than the SSNNLM model, the SSNNEKF model is much faster to compute and easier to implement, which makes the SSNNEKF approach more suitable for real-time applications. Guo et al. [67] applied a so-called the adaptive KF algorithm upon the state space representation of a SARIMA-GARCH model to perform short-term traffic flow prediction. Their results show that the performance of the adaptive KF algorithm can be stabilised by increasing the computation memory size. There are other types of nonlinear KFs applied for traffic and travel time prediction, including the iterated EKF [10], the unscented KF [10], and the ensemble KF [218].

The advantages of KF based approaches are summarised as follows: (1) it is easy to incorporate both historical and real-time data into the model; (2) both departure and arrival travel time prediction can be performed; (3) it is easy to implement and fast to run; (4) and the prediction uncertainty can be estimated. To address nonlinearity and volatility in the data, nonlinear KFs can be applied. However, they are often more complicated and computationally more expensive.

3.3.5 Artificial Neural Networks

Artificial neural networks (ANNs), commonly referred to as 'neural networks', was inspired by analogy with the (human) brain which is capable of organising its structural constituents, known as neurons, which perform computations tremendously faster than the fastest digital computer today. Aleksander et al. gave the definition

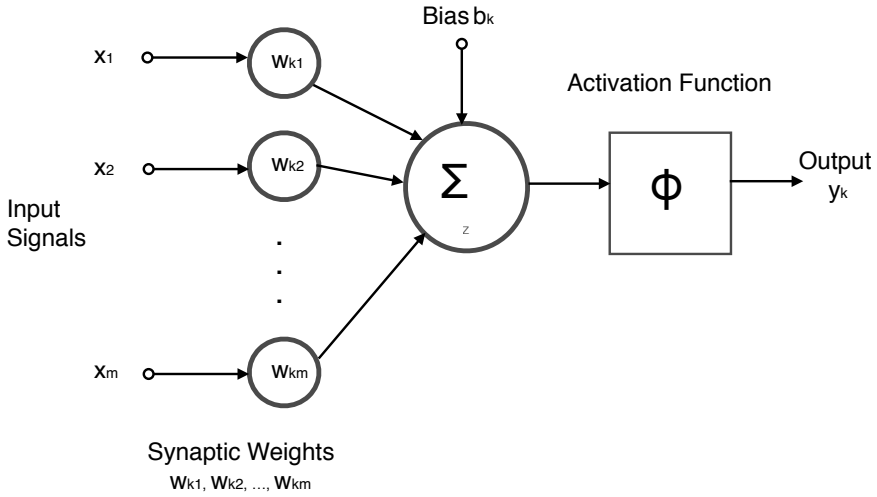


Figure 3.2: Modelling of neuron (recreated using Figure 1.5 in [184]).

of a neural network as follows [7]:

Definition 8 ***A neural network*** *is a massively parallel distributed processor made up of simple processing units*[2]*, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:*

- *Knowledge is acquired from its environment through a learning process.*
- *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

The fundamental element (or unit) of an ANN is neuron. Figure 3.2 shows how a neuron takes input signals, sums them with synaptic weights, and produces output signals using an activation function. The bias b_k may increase or decrease the net input for the activation function, depending on whether it is positive or negative.

[2]They are referred to as neurons.

The neuron k in the figure can be mathematically expressed as follows [184]:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{3.36}$$

and

$$\begin{aligned} y_k &= \phi(v_k) \\ v_k &= u_k + b_k \end{aligned} \tag{3.37}$$

where $x_1, x_2, ..., x_m$ are the input signals; $w_{k1}, w_{k2}, ..., w_{km}$ are the synaptic weights; u_k is the linear combiner output based on the input signals; b_k is the bias; v_k is the output with the effect of b_k; $\phi(\cdot)$ is the activation function; y_k is the output signal of the neuron k.

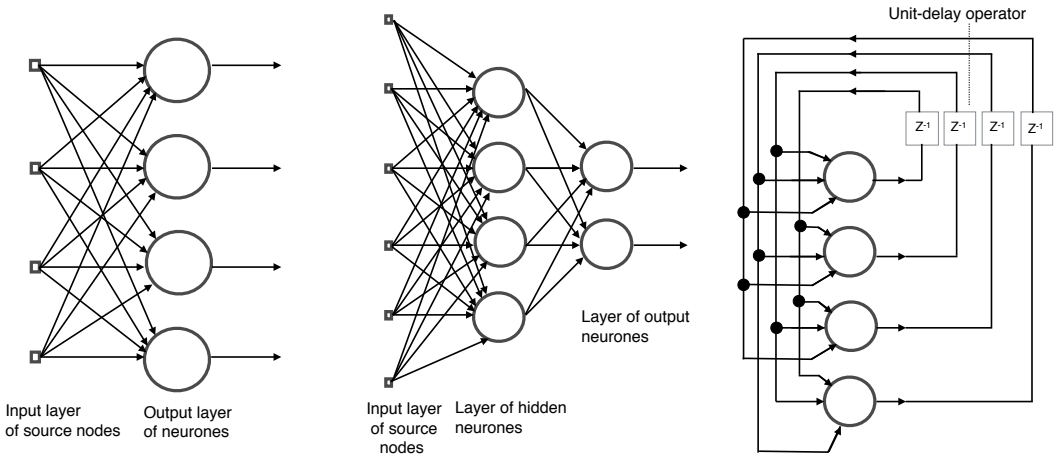
Various types of activation functions have been developed for ANN including threshold functions, piecewise-linear functions, sigmoid functions, softmax functions, and rectified functions. The commonly used hyperbolic tangent function and logistic function are sigmoid functions.

ANNs can be classified into single-layer feedforward neural networks[3], multilayer feedforward neural networks, and recurrent neural networks based on their network architectures [184]:

- **Single-layer feedforward neural networks** only have one input layer of source nodes which directly projects onto an output layer of neurons. They are 'single-layer' because the input layer does not count for ANNs. The concept of single-layer feedforward neural networks is shown in Figure 3.3a.

- **Multilayer feedforward neural networks** have one or more *hidden layers* whose computation nodes are called hidden neurons or hidden units. Adding hidden layers makes the networks capable of extracting higher-order statistics

[3]The graphic representation of an ANN is a directed acyclic graph, thus it is called feedforward neural network.



(a) Single layer feedforward neural network

(b) Multilayer feedforward neural network

(c) Recurrent neural network)

Figure 3.3: Three types of ANN: single-layer feedforward neural network, multilayer feedforward neural network and RNN (recreated based on Figure 1.15, Figure 1.16 and Figure 1.17 in [184]).

particularly when the size of input neurons is large. A multilayer feedforward neural network does not have to be *fully connected* (i.e., each nodes in one layer is connected with each nodes in the next layer). The exception is called *partial connected* network. A multilayer feedforward neural network is also known as the multilayer perceptrons (MLPs). The concept of fully connected MLPs is shown in Fig. 3.3b.

- **Recurrent neural networks (RNNs)** are able to carry at least one *feedback loop* so that the neuron in a feedback loop can feed its output signal back to the inputs of all the other neurons. RNNs accept self-feedback loop in which a neuron feeds back itself. Hidden neurons are not compulsory for RNNs. Figure 3.3c shows the concept of RNNs with no hidden neurons and no self-feedback loops.

Multilayer Perceptrons

Figure 3.3b shows the concept of single hidden layer MLPs. MLPs have one or more hidden layers of hidden neurons, which are able to extract higher-order statistics of input signals. MLPs also have a high degree of connectivity which makes MLPs more complicated when dealing with feedbacks and synaptic weights updates, especially when multiple hidden layers are involved. There are two types of signals in MLPs: function signals and error signals (see Fig. 3.4) [184]:

- **Function signals** (solid green lines in Figure 3.4). A function signal comes from the input source nodes, *propagates forward* from layer to layer, emerges at the output layer, and becomes the output signal at the end. The functional signal is also referred to as the input signal because it is the input of its connected neuron(s) at the next layer.
- **Error signals** (dashed red lines in Figure 3.4). An error signal comes from the output layer, *propagates backforward* from layer to layer. It is called the

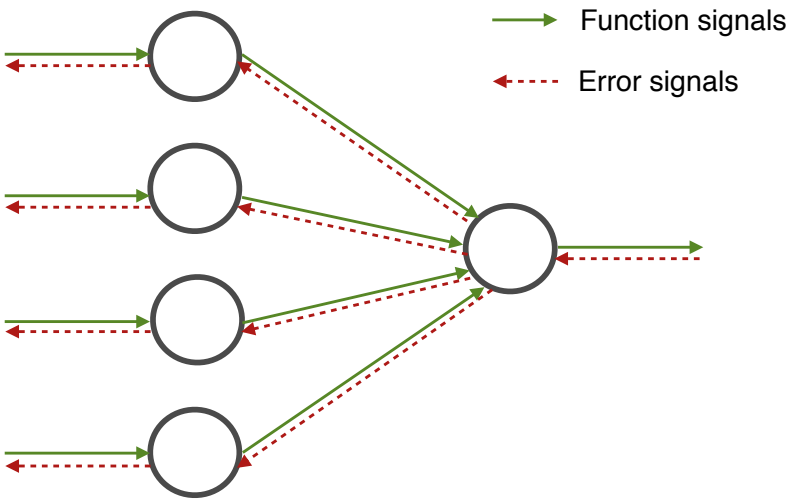


Figure 3.4: Two types of signal flows in MLPs: the forward propagation of function signals (solid green lines) and the back-propagation of error signals (dashed red lines) (recreated using Figure 4.2 in [184]).

error signal because it involves an error-dependent function in one neuron from another.

There are two computations involved in each hidden neurons: (1) the computation of the output for the next layer with a nonlinear function based on the input signals and the synaptic weights associated with the neuron; (2) the estimation of the gradients of the error surface w.r.t. the synaptic weights connected to the input of the neuron, which is required for the back-propagation throughout the network [184]. The back-propagation of the gradients of the errors aims to update the synaptic weights to improve the performance of the network. The update can be performed in many iterations. Let m denote the number of inputs for the neuron j, at the nth iteration, the update of the synaptic weight w_{ji} can be expressed as [184]:

$$\Delta w_{ji} = \eta \delta_j(n) y_i(n) \tag{3.38}$$

where η is the learning rate; $\delta_j(n)$ is the local gradient which is defined as [184]:

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial v_j(n)} = e_j(n)\phi'_j(v_j(n)) \tag{3.39}$$

where $v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n)$ is the induced local field produced at the input of the activation function; $e_j(n)$ is the prediction error at the nth iteration; $\phi'_j(v_j(n)$ is the derivative of the activation function. To to compute the derivatives for activation functions used in ANN, activation functions are required to be continuously differentiable. The most commonly used activation function is sigmoidal nonlinearity function (e.g., logistic function, hyperbolic tangent function). The nonlinearity of activation functions is important for MLPs because otherwise the input-output relation can be reduced to that for single-layer perceptrons. Also, the nonlinearity is smoother than the linearity [184].

Smaller η yields smaller changes per iteration thus make the entire updating

trajectory in the weight space more smoother. However, it also makes the learning slow. Increasing η can increase the learning speed, but it can result in a less stable model. Rumelhart et al. [169] introduced a momentum term to the updating process of synaptic weight. In this case, we can increase the learning speed without increasing the risk of model instability. Hence, Equation (3.38) becomes [184]:

$$\Delta w_{ji} = \alpha \Delta w_{ij}(n-1) + \eta \delta_j(n) y_i(n) \tag{3.40}$$

where α is the momentum constant, usually a positive value; $\Delta w_{ij}(n-1)$ is the synaptic weight from the last iteration. MLPs can be trained in both sequential mode and batch mode. The key difference between these two modes is how the training example is presented for updating the weights in each iteration. For sequential mode, the operation is performed after the presentation of an individual training example; for batch mode, the operation is performed after the presentation of all the training examples. The optimal number of hidden neurons, learning rate and momentum constant can be determined using cross validation. The procedure is usually very time-consuming. For the more comprehensive explanation of MLPs, please refer to [184, Chapter 4].

The applications of MLPs in the area of transportation research can be traced back to early 1990s. Dougherty [37] provided a comprehensive review of the early studies including those for traffic predictions. Taylor et al. [186] implemented a single-hidden-layer MLPs (SHL-MLPs) (with 28 hidden neurons) for traffic flow prediction for highways. In their study, flow volumes and occupancies data in the past are used as the inputs for the network. A customised scale method was developed to deal with the missing values in the training data. The results show that the performance is appealing for 5-mins ahead predictions. Dougherty et al. [38] implemented an SHL-MLPs (with 10 hidden neurons) to perform multi-step ahead traffic flow predictions for inter-urban motorways. Instead of using fixed number of inputs, they selected different inputs for different step-ahead predictions. The

results show that the speed prediction performance with their MLPs is not very appealing. Ledoux [108] implemented an SHL-MLPs (with 4 hidden neurons) to perform one-step ahead queue length prediction. Huisken et al. [88] implemented an SHL-MLPs (with 5 hidden neurons) to perform short-term travel time prediction for highways. Twelve inputs are used in their study including speed information at bottlenecks and in between congested areas. The results show that their model outperforms two travel time estimation methods. Mark et al. [133] developed an SHL-MLPs (with 8 hidden neurons) to predict travel times using time-lagged aggregated traffic flow and travel time data generated from the Parallel Microscopic simulation system (PARAMICS). The results show that: (1) the model performance is not affected by varying the time lag (up to 20 minutes); (2) speed data is a better input source than flow data; (3) and including more incident data yields a better performance. A more recent study [68] was found implemented an SHL-MLPs to predict stop-to-stop bus travel times.

It can be seen that the number of hidden neurons for MLPs varies from study to study. In general, the choice is between 4 to 10 neurons. For large-scale road networks, the hidden neurons can reach 50 or more, depending on the number of inputs used. One can jointly tune the number of hidden neurons with other model hyperparameters. However, such joint tuning can yield much longer model fitting time. Sigmoid function is the most popular choice for activation functions, particularly if the back-propagation is used. However, there lacks in-depth discussion on the selection of other important parameters such as momentum factor, learning rate etc.

Multilayer Perceptrons with Better Converging Training Method

The back-propagation commonly searches along the steepest decent directions. Although such search strategy has low computational cost, it has two main problems: (1) it only guarantees to converge to a local optimal; (2) it may converge slowly. Many ANN experts do not worry about the local minima problem too much due

to the following reasons: (1) some local minima are caused by identical models; (2) slightly less optimal local minima does not yield a significant difference in model performance; (3) it is too costly to find global optimal, which may result in an over-fitted model. Hence, many researchers focus on seeking a better converging training approach for MLPs. Innamaa [90] used Fletcher-Reeves update method to train a MLPs for travel time prediction. The method is a conjugate gradient (CG) algorithm that searches along the conjugate directions. It generally converges faster than those searches along the steepest decent directions. Vlahogianni et al. [98] used a scaled CG algorithm to train an SHL-MLP for traffic flow prediction. The results show that the scaled CG algorithm converges smoothly. van Lint et al. [201] used Bayesian regularised Levenberg-Marquardt algorithm (LM-BR) to train a state-space ANN (the network is not a typical MLPs but can be considered as a variant of MLPs). LM-BR was first proposed by Mackay [130] to address the risk of overfitting using Levenberg-Marquardt algorithm. Chan et al. [23] also used two Levenberg-Marquardt algorithms to train an SHL-MLPs for traffic flow prediction.

Evolutionary learning algorithms also have been used to optimise the converging training of MLPs. Annunziato et al. [9] compared two evolutionary learning algorithms, Partial Emulation and Chaotic Populations, with the traditional back-propagation approach to train an SHL-MLPs for short-term urban traffic flow prediction. The results show that the two evolutionary learning approaches are slightly better than the BP approach.

Genetic algorithms (GAs) is a very popular choice in the area of computational intelligence. That is mainly because they use heuristic to search optimal solution, and they can be used when gradients are not available. John Holland stated in his book [80] that GA can perform a robust search by implicitly sampling hyperplane partitions of a high-dimensional search space. A hyperplane can be first encoded by bit strings, known as '*chromosomes*'. Then, GA converges to an optimal solution by applying '*selection*' (based on fitness evaluation), '*reproduction*', '*crossover*', and '*mutation*'. For the more detailed information about GAs, please refer to [80, 215].

Many studies can be found using GAs to train MLPs. Vlahogianni et al. [206] used GA to train an SHL-MLPs for traffic flow prediction. Three training strategies are introduced in their study: (1) use the back-propagation approach for the network training and use GA to tune the model hyperparameters including the learning rate, the momentum term, and the number of hidden neurons; (2) use the back-propagation approach with an adaptive learning rate for the network training and use GA to tune the number of hidden neurons; (3) use scaled conjugate gradient descent algorithm for the network training and use GA to tune the number of hidden neurons. Their study results show that the 2nd and the 3rd strategies converge faster than the 1st one, and all the genetically trained SHL-MLPs outperform an ARIMA model. Khosravi [98] used GA to train a MLPs for both bus travel time prediction and highway travel time prediction. A novel point of their study is that a prediction interval based cost function is used to replace the classic error based cost function for the training. Several prediction interval based cost functions are investigated. The results show that the prediction interval based training approaches converge faster, and high-quality prediction uncertainty estimation can be provided. Note that the computational cost of evolutionary learning algorithms is usually more expensive than that for the back-propagation approaches that utilise gradient steepest decent algorithms.

Recurrent Neural Networks

Besides MLPs, many RNNs also have been developed for traffic and travel time prediction. RNNs have '*memory*'. They can utilise their *internal memory neurons* to process arbitrary sequences of inputs, thus makes them capable of learning temporal sequence [128]. There are three major types of RNN used in the existing traffic and travel time prediction studies [128]:

1. **Elman neural network (Elman NN)** is also known as the simple RNN [42]. It feeds back the activations from the previous set of hidden neurons to

its inputs. The activations can be seen as the state of a network. The network is able to incorporate the previous state to model a temporal process (or time series). Several studies can be found applying Elman NNs for traffic prediction [91, 6]. van Lint et al. [198] proposed a state-space RNN (SSNN) to predict travel time for highways. The proposed SSNN can be seen as a variant of Elman NN where the hidden neurons represent the state of individual road links. Thus, each hidden neuron is just fed with the data associated with the link it represents (i.e., the input neurons and hidden neurons are not fully connected). The data consists of mean speeds and flow rates measured at the detector stations connected to the link, during the time period [t-1, t]. Inflow and outflow data can be included, depending on whether the link is connected to an on-ramp or an off-ramp. Further studies on SSNN can be found in [201, 199, 121, 197]. Some studies show that incorporating techniques such as EKF [121] or Bayesian committee [197] can improve the training quality and prediction performance for SSNNs. They also make SSNNs more feasible for online prediction tasks.

2. **Time-delay neural network (TDNN)** feeds back the previous inputs into the current inputs, which makes the inputs a time series. Hence, a TDNN can be considered as a nonlinear multivariate autoregression model. And it becomes more suitable for multivariate time series prediction. Early discussion in this chapter have shown that GAs are capable to provide robust converging training for MLPs. They can also be used to optimise the training process for TDNNs [71]. Lingras et al. [118] applied GA to train a TDNN for inter-city traffic flow prediction. The results show that the genetically trained TDNN outperforms a non-genetically trained TDNN. Abdulhai et al. [2] implemented a TDNN to predict traffic flow and density for highways. In their study, GA is used to determine the temporal lag dynamically when spatial neighbourhood is taken into account. The results show that their TDNN outperforms an SHL-

MLPs when only spatial neighbourhood is incorporated. Zhong et al. [241] also implemented a TDNN to predict traffic flow. In their study, GA is used as a data filtering method that aims to find the best inputs (i.e., the most correlated w.r.t. the observed outputs). The results show that the TDNN outperforms a locally weighted linear regression model. Shen et al. [176] investigated three types of RNNs including TDNN (with only local feedback loops), the generalised SSNN (with only global feedback loops) and Elman NN (with both local and global feedback loops) for travel time prediction for highways. Different temporal lags are examined with these RNNs. The results show that all the RNNs outperform a MLPs. TDNN turns out to be the best among the three types of RNNs.

3. **Nonlinear autoregressive with exogenous inputs models (abbreviated as NARXs)** have been widely applied in structural engineering. NARXs can be thought of as the ANN version of the generalised time series models which can exogenously take time series inputs. A NARXs feeds back the network using the estimates of the output, rather than the outputs of hidden neurons as those for SSNNs [231]. Zeng et al. [231] proposed a time-delayed state-space neural network (TDSSNN) for travel time prediction for highways. The TDSSNN can be considered as a modified NARX. The main differences between the TDSSNN and traditional NARXs are: (1) the TDSSNN feeds back the network using the outputs of hidden neurons but not the output from the output neurons; (2) the TDSSNN only allows one time-unit delay, whereas the feedback layer can take more than one time-unit delay in NARXs. Different input delay settings were examined in this study. The results show that the TDSSNN approach outperforms the TDNN approach for all cases. Wang et al. [210] proposed a space-time delay neural network (STDNN) to predict travel time for urban roads. The STDNN have additional hidden layer in which each hidden neuron is *the pseudo-adjacency* of the target spatial location. A spa-

tial operator is introduced to the STDNN to calculate the weighted sum of the inputs where each weight represents the strength (i.e., the level of correlation, either 1 or 0). The resulted sum later is used to calculate the output at the output layer. The STDNN is trained using Levenberg-Marquardt algorithm. One novel feature of the STDNN is that it can be easily extended to multiple output architecture. Thus, multi-spatial prediction can be performed with only one run (i.e., we can perform travel time predictions for multiple road links simultaneously). Their study results show that the STDNN outperforms a ARIMA model and a SARIMA model.

4. **Long short-term memory neural network (LSTMNN)** was initially introduced by Hochreiter and Schmidhuber [79]. In LSTMNN, a hidden layer is a memory block which contains a memory cell, an input gate and an output gate that controls the input and output activations into the block, and a forget gate, respectively. The core of the memory cell is a recurrently self-feed linear unit, namely Constant Error Carousel (CEC), and the activation of the CEC which represents the state of the cell. The multiplicative input and out gate can learn to open or close based on the state of the cell. Thus, LSTMNN can solve the problem of vanishing error by '*remembering*' the network error (i.e., remaining the error constant). The forget gate is able to reset the memory if the information flow is out of date, and replace the CEC weight with the multiplicative forget gate activation. Ma et al. [128] argued that traditional RNNs such as Elman NNs, TDNNs and SSNNs have two main problems when performing traffic prediction tasks: (1) they are not able to handle time series data with long time lags; (2) the predetermined time lags for RNNs may not be optimal. Hence, they implemented an LSTMNN to address these problems [128]. Their study results show that the LSTMNN approach outperforms the traditional RNN approaches.

The main differences between SSNNs, TDNNs and NARXs can be generalised to

the problem that how the state of a temporal or temporal-spatial stochastic process should be modelled and used in the network. Networks that have the capability to model temporal-spatial process usually have better performance. However, they are more difficult to implement and train, which is mainly due to the vanishing gradient and exploding gradient problems [79]. LSTMNNs can be used to address these problems [128]. In general, RNNs are more adaptive to the changes of traffic conditions when compared with MLPs.

Deep Neural Networks

A deep neural network (DNN) is referred to as the ANN architecture that is able to perform deep learning. The deep learning is a particular kind of machine learning that can be regarded as the study of models that either involve a great amount of composition of learned functions or learned concepts that is more than traditional machine learning approaches [64]. The main difference between DNNs and traditional '*shallow*' ANNs is that traditional '*shallow*' ANNs need only a few steps to find proper synaptic weights while DNNs require many. Various types of DNNs have been proposed including deep feedforward neural networks (i.e., MLPs with multiple hidden layers), convolutional neural networks (CNNs), and RNNs (with much more recurrences) etc. For the more detailed discussion on DNNs, please refer to [170, 64].

DNNs have been applied to many areas including computer vision, speech and audio processing, natural language processing, video games, search engines, online advertising, finance, robotics, bioinformatics, and chemistry. Several studies have been found apply DNNs for traffic prediction. Lv et al. [125] implemented a stacked autoencoders (SAEs) to predict traffic flow for highways. A SAEs is built by stacking multiple layers of autoencoders where each autoencoder layer is trained to reconstruct its inputs [205]. In their study, a logistic regression layer is placed on top of the SAEs so that the resulting model is able to perform the prediction task. Their SAEs is trained in two steps: (1) the model is first trained layer by layer in a bottom-up fashion using a greedy layerwise unsupervised learning algorithm; (2)

the model parameters are fine-tuned layer by layer in a top-down fashion using the back-propagation approach as for MLPs [14]. Different topologies are configured for different prediction horizons. Only traffic volume data is used as the inputs for their study. The results show that their DNN approach outperforms a MLPs, a random walk prediction model, a SVR model, and a radial basis function neural network. Wang et al. [208] implemented a variant of CNN, namely error-feedback recurrent CNN (eRCNN) to continuously predict traffic speed using the speed information in taxis' GPS data for urban roads. The eRCNN contains five layers: an input layer that transforms the speed data into a spatial-temporal input matrix, a convolution layer and a pooling layer that extracts more representative features from the inputs, an error-feedback recurrent layer that recognises the prediction errors using the predictions from previous periods, and an output layer that uses a modified rectified linear neuron to predict traffic speed. With the the recurrent layer, the eRCNN is able to model the abrupt changes in traffic speeds caused by some emerging traffic events (e.g., the morning peaks and traffic accidents). The eRCNN is pre-trained using clustered speed data at first. Then, it is trained using a mini-batch stochastic gradient descent algorithm. Their study results show that the eRCNN outperforms some state-of-the-art approaches including ARIMA [3], SVR [219], and SEAs [125]. Ma et al. [127] proposed an interesting idea of applying CNN for traffic flow prediction. In their study, the traffic flow of a road network is first converted to a set of images where each image represents a daily time-space traffic speed. Then a CNN is implemented so that traffic features can be extracted using the convolutional and pooling layers and traffic speed can be predicted using a fully connected layer. The back-propagation approach is used to train their CNN. The results show that the CNN approach outperforms some other ANN approaches including MLPs, RNN, LSTMNN and SAEs.

Hybrid DNN architectures also have been proposed for traffic flow prediction. Ma et al. [129] combined SSNN and conditional Restricted Boltzmann Machine (RBM) (RBM-SSNN) to learn traffic congestion pattern for large-scale urban road network

using taxis' GPS data. RBM is an energy based network in which a probability density function is used to determine whether a process unit in the network should be 'turned on' or 'turned off'. The probability density function can be modified as a conditional probability density function based on the previous state of the network. Thus, combining SSNN and RBM grants the capability of learning complex spatial-temporal time series. A mini-batch stochastic gradient descent algorithm is used to train the RBM-SSNN. The results show that the RBM-SSNN is able to predict the revolution of the congestion in a test bed road network with more than 500 links. Wu et al. [220] combined CNN and LSTMNN to perform traffic flow prediction for highways. The hybrid DNN contains a one dimensional CNN, two LSTMNNs, and a regression layer. The CNN is used to capture the spatial features of the traffic flow because it is good at handling data representation with a locality structure. The two LSTMNNs are used to capture the short-term, daily and weekly features of the traffic flow. All these features are sequentially concatenated into a feature vector, which is used in the regression layer for traffic flow prediction. The hybrid DNN is trained using a first-order gradient-based optimisation algorithm as is introduced in [102]. The results show that the hybrid DNN approach outperforms MLPs, LSTMNN, SAEs, and gradient boosting regression tree. A similar study can be found in [229]. Jia et al. [93] investigated the performance of two DNN approaches, deep belief network and LSTMNN, for traffic flow prediction. The rainfall intensity data is added as an input feature. The results show that the rainfall intensity data is beneficial to the prediction with both DNN approaches. The LSTMNN approach is overall the best.

Until the time I finished this thesis, no studies have been found using DNNs for travel time prediction in the literature. It is not difficult to extend these ideas for travel time prediction. However, it should be noted that whether to implement a DNN for traffic or travel time prediction really depends on if the prediction problem requires deep learning. For travel time prediction problems, it is not necessary to have very large training data with large feature dimensionality. In this case, applying

DNNs over the problems can be an overkill w.r.t. the computational cost and it may not yield satisfying results. Another issue is the interpretability. Since the resulted models are way more complex than shallow ANNs, it is even more difficult to examine the causal relationship between the input features and the output. The main challenge for DNNs is the training. It is more difficult and computationally more expensive to train DNNs than shallow ANNs. To reduce the training difficulty, the mini-batch stochastic gradient descent algorithm is usually applied. In this case, a large training data is first split into a number of small batches, then the less expensive standard gradient descent based back-propagation approach can be performed. Another issue regarding DNNs is about tuning the hyperparameters. The traditional cross-validated grid search approach is not suitable for the task due to its low efficiency and poor capability of capturing the hierarchical relations between hyperparameters, more efficient approaches such as sequential model based optimisation should be considered.

3.3.6 Kernel Methods

Recall the general expression of the linear regression model in Equation (3.10) and the OLS estimation in Equation (3.11), if $\mathbf{X}^T\mathbf{X}$ is singular, the pseudo-inverse can be used. Then, we can find $\boldsymbol{\beta}$ that satisfies Equation (3.11) with the least norm. Alternatively, we can trade off the size of the norm against the loss. This approach is known as the ridge regression (RR). In RR, we search the best $\boldsymbol{\beta}$ by solving the following optimisation problem [174, pp.31]:

$$\min_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}, \mathcal{D}) = \min_{\boldsymbol{\beta}} \lambda \|\boldsymbol{\beta}\|^2 + \sum_{i=1}^{N} (y_i - g(\mathbf{x}_i))^2 \tag{3.41}$$

where $\mathcal{L}$ is the loss function in which the trade-off between the norm (the first term) and the loss (the second term) can be controlled by the weight factor λ ($\lambda > 0$); $g(\mathbf{x}_i)$ is the predicted output given $\mathbf{x}_i$. Balancing the trade-off between the norm

and the loss is also known as the *regularisation*. $\mathcal{L}$ thus is also referred to as the least squares with $l2$-norm regularisation. Thus, we can solve $\boldsymbol{\beta}$ by taking the derivative of $\mathcal{L}$ w.r.t. $\boldsymbol{\beta}$ [174, pp.31]:

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^T\mathbf{y} \tag{3.42}$$

where $\mathbf{I}_d$, with a size of $d \times N$, d is the dimensionality of the input space and N is the number of instances in the dataset. The computation complexity for solving $\boldsymbol{\beta}$ is $\mathcal{O}(N^3)$. Alternatively, we can rewrite Equation (3.42) to [174, pp.31]:

$$\boldsymbol{\beta} = \mathbf{X}^T\boldsymbol{\alpha} \tag{3.43}$$

where

$$\begin{aligned}\boldsymbol{\alpha} &= \lambda^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{y} \\ &= (\mathbf{G} + \lambda\mathbf{I}_d)^{-1}\mathbf{y}\end{aligned} \tag{3.44}$$

Here, $\mathbf{G}$ is a Gram matrix where each entry $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Solving $\boldsymbol{\alpha}$ involves solving N linear equations with N unknowns. The resulting prediction function becomes [174, pp.31-32]:

$$g(\mathbf{x}) = \sum_{i=1}^{N} \boldsymbol{\alpha}_i \langle \mathbf{x}_i, \mathbf{x} \rangle = \mathbf{y}^T(\mathbf{G} + \lambda\mathbf{I}_d)^{-1}\mathbf{k} \tag{3.45}$$

where $\mathbf{x}$ is the general form of any input in $\mathbf{X}$; $\mathbf{k} = \langle \mathbf{x}_i, \mathbf{x} \rangle$. If $d > N$, it becomes more efficient to solve our linear problem using Equation (3.43) than using Equation (3.42). In this case, the prediction on a new test input only requires the inner products between the training data and the new test input [174, pp.32-33].

To address the nonlinearity in the data, the multi-dimensional input data is

mapped to a high-dimensional feature space so that linear regression techniques such as RR can be performed on the feature space:

$$\phi : \mathbf{x} \in \Re^d \mapsto \phi(\mathbf{x}) \in F \subseteq \Re^h \tag{3.46}$$

where ϕ is the mapping; F is the feature space. The input data in F becomes $\{\phi(\mathbf{x}_i)\}_{i=1}^N$ of dimension h. Although we can use Equation (3.42) to solve $\boldsymbol{\beta}$, the computational cost can be very expensive if N reaches few thousands. Alternatively, we can use Equation (3.43) so that all the information needed is just the inner products between two data points in F. In this case, each entry in $\mathbf{G}$ becomes $\mathbf{G}_{ij} = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$, the rows of $\mathbf{X}$ becomes the feature vectors $\{\phi(\mathbf{x}_i)^T\}_{i=1}^N$, and $\mathbf{k} = \{\langle\phi(\mathbf{x}_i), \phi(\mathbf{x})\rangle\}_{i=1}^N$. Assume the computational complexity of ϕ is $\mathcal{O}(h)$, the complexity for the inner products $\langle\phi(\mathbf{x}_i), \phi(\mathbf{x})\rangle$ is still $\mathcal{O}(h)$. Thus, the overall computational complexity for solving α is $\mathcal{O}(N^3 + N^2h)$ and the computational complexity for the prediction on a new test input is only $\mathcal{O}(Nh)$. The inner products can be computed more efficiently as a direct function of the input features, without explicitly computing the mapping ϕ. The function that performs such direct computation is called *kernel function* [174, pp.33-34]. A kernel function can be defined as follows [174, pp.34].

Definition 9 *A **kernel function** or **kernel** k is a function that for all $\mathbf{x}$, $\mathbf{x}' \in \mathbf{X}$ that satisfies:*

$$k(\mathbf{x}, \mathbf{x}') = \langle\phi(\mathbf{x}), \phi(\mathbf{x}')\rangle \tag{3.47}$$

where ϕ is a mapping from $\mathbf{X}$ to an (inner product) feature space F.

The above mapping using kernel function is also known as *the kernel trick*. Methods that utilise the kernel trick are generally known as kernel methods. In machine learning, kernel methods are commonly used for pattern analysis because they are very good at learning nonlinear decision boundary.

Because α in Equation (3.44) is not sparse, the prediction on a new test input requires the evaluation of the kernel with every training instances. If the training data is large, the response time can be slow [174, pp.234]. This led to the development of support vector regression (SVR) with loss functions that are able to use sparse data for the inference. SVR is one of the most widely used kernel methods for regression problems. SVR can be differentiated in terms of the loss function it uses. For example, ϵ-SVR uses ϵ-insensitive loss function; ν-SVR introduces the parameter ν to the loss function to control the fraction of the support vectors.

The very first application of SVR for travel time prediction was probably done by Wu et al. [219]. In their study, ϵ-SVR is used to predict travel time on three relatively long-stretch roads. A linear kernel function is selected with $\epsilon = 0.01$ and $C = 1000$[4]. Vanajakshi et al. [204] implemented an SVR model to perform short-term travel time prediction. The radial basis function (RBF) is used as the kernel function. The results show that the SVR model outperforms two naïve methods and an ANN. Castro-Neto et al. [21] implemented an online SVR [126] to predict traffic flow. In their study, RBF is also used as the kernel function. The results show that the online SVR approach outperforms three other approaches including ANN, Gaussian maximum likelihood (GML), and exponential smoothing (ES), at some vehicle detection stations under atypical traffic conditions such as holidays and incidents. Wang et al. [143] implemented a ν-SVR model to predict bus travel time. In their study, road section number, road section length, the number of intersections, historical bus travel time, and departure time at the departure station are used as the input features. RBF is also selected as the kernel function with $\epsilon = 1$ and $C = 256$. Chen et al. [223] incorporated SVR with KF to predict bus rapid transit (BRT) vehicle travel time. In their hybrid approach, an SVR model is first fitted to predict initial travel times. Then, the predicted travel times are corrected recursively using KF. Wang et al. [211] implemented an SVR model to perform short-term

[4] C is the control factor in the loss function that controls the trade-off between the norm and the performance loss

traffic speed prediction for urban highways. In their study, traffic speed is assumed nonlinear, non-stationary and chaotic. Hence, the dimensionality of the input space is first identified based on the phase space reconstruction theory. A novel non-stationary kernel is developed using the Mexican-hat wavelet function. The results show that the proposed hybrid approach outperforms the standard SVR approach. A recent study [140] has been found implemented an SVR model to predict bus travel time. In that study, GA is used to automate the model hyperparameters tuning for SVR.

There are other types of kernel methods that have been found in the literature including ridge regression (RR) [74] and Gaussian process regression (GPR) [89, 222, 115]. One unique feature about GPR is that it is a full Bayesian framework so that the prediction uncertainty (or the prediction interval) estimation can be provided. Also, GPR does not require a time-consuming validation procedure as for ANNs and SVR. GPR is also considered as an ensemble method which assembles latent prediction functions as a Gaussian distribution.

Kernel methods can flexibly utilise different kernel functions (or even customised kernel functions) to capture local and global trend in data. They are considered more robust on handling sparse data when compared with the TSA approaches. The performance of kernel methods very much relies on the selection of kernel function(s) and the quality of model hyperparameters tuning. The resulted models are often difficult to interpret.

3.3.7 Decision Tree Ensembles

A decision tree (or tree) partitions a training data into a set of disjoint regions and then fits a simple model (e.g., linear regression model) to each region. Recall that we have data $\mathcal{D}$ consisting of N input vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ of dimension d ($d \geq 1$) and the corresponding output vector $\mathbf{y} = \{y_i\}_{i=1}^{N}$, $\mathbf{X} \in \Re^d$, $\mathbf{y} \in \Re$. Assume we first partition $\mathcal{D}$ into M disjoint regions $R_1, R_2, ..., R_M$, we can model the output as a

constant c_m in each region [73]:

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbf{I}(\mathbf{x} \in R_m) \tag{3.48}$$

We can solve c_m using LSE. In this case, the best estimation for c_m is just the mean of y in region R_m. However, finding binary partition is computationally infeasible using LSE. Thus, a greedy search algorithm is applied to find the best input variable with the best splitting point. Given a splitting variable j and its corresponding splitting point s, we get two disjoint regions $R_1(j,s) = \{\mathbf{X}|\mathbf{X}_j \leq s\}$ and $R_2(j,s) = \{\mathbf{X}|\mathbf{X}_j > s\}$. Then j and s can be solved by solving the following optimisation problem [73]:

$$\min_{j,s} \left[\min_{c_1} \sum_{\mathbf{x}_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{\mathbf{x}_i \in R_2(j,s)} (y_i - c_2)^2 \right] \tag{3.49}$$

The inner minimisation above can be solved as [73]:

$$\hat{c}_1 = \arg\operatorname{ave}(y_i|\mathbf{x}_i \in R_1(j,s)) \; \textit{and} \; \hat{c}_2 = \arg\operatorname{ave}(y_i|\mathbf{x}_i \in R_2(j,s)) \tag{3.50}$$

The above procedure is recursively performed for each region till a stop (e.g., reaches the maximum tree depth, or reaches the minimum sample size in leaf node) is met. Since a very large tree may overfit the given data, a post-pruning process is often required. One post-pruning technique is called *cost-complexity pruning* which aims to govern the trade-off between the model complexity and the goodness of data fitting. There are various way to implement a single tree model including the iterative dichotomiser 3 (ID3) [158], C4.5, C5.0, and the classification and regression tree (CART). Most tree implementations in the literature prefer binary splits over multiway splits. That is mainly because: (1) multiway splits fragment the data too fast which results in insufficient data in the next tree level; (2) multiway splits always can be done by a series of binary splits [73].

Single tree models are sensitive to the data: small changes in the data can result in very different tree structures which may lead to very different prediction results. This structural instability is the hierarchical nature of the growing procedure, which can hardly be avoided even with a smarter split strategy [73]. Therefore, single tree models have high variance. To address the drawback of single tree models, tree ensembles have been introduced to reduce the model variance. A tree ensemble assembles multiple trees where each tree is grown to a given data set drawn from training data with or without replacement.

There are mainly two types of tree assembling strategies: *bagging* and *boosting*. With a bagging strategy, a tree ensemble is a collection (or a 'bag') of single tree models where each tree is grown to a bootstrap sample drawn from the training data with replacement. To make prediction with the ensemble, we can just average out the outputs of all the trees in the ensemble. Such tree ensemble is also known as '*bagged trees*'. The general form of bagged trees can be expressed as:

$$F(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x}) \tag{3.51}$$

where B is the number of trees; T_b is the bth tree in the ensemble. If trees in the ensemble are deep enough, the bias of the ensemble can be kept as low as that for individual trees. The most famous tree ensembles with bagging strategy are random forests (RF) [19]. For the more detailed discussion about RF, please refer to [19].

Because trees in the ensemble of bagged trees are grown independently, each instance in the training data is equally likely to be picked. Boosting strategies focus on the instances that are not learned well using the prior ensemble. Thus, the chance that a particular instance being picked for the next boosting iteration depends on the performance of the existing trees on that instance. With a boosting strategy, a

tree ensemble is built in an additive expansion form:

$$F(\mathbf{x}) = \sum_{m=1}^{M} T_m(\mathbf{x}) \tag{3.52}$$

where M becomes the number of boosting iterations. At each boosting iteration m, the ensemble can be expressed as:

$$F(\mathbf{x})_m = F(\mathbf{x})_{m-1} + \gamma_m T_m(\mathbf{x}) \tag{3.53}$$

where γ_m is the boosting coefficient. γ_m can be solved by minimising the following objective function:

$$\min_{\gamma} \sum_{i=1}^{N} \mathcal{L}(y_i, F_{m-1}(\mathbf{x}_i) + \gamma T_m(\mathbf{x}_i)) \tag{3.54}$$

where $\mathcal{L}$ can be any differentiable loss function (e.g., squared loss, absolute loss, Huber loss). Regularisation term such as $l1$-norm and $l2$-norm also can be added to the above objective function to govern the trade-off between the model complexity and the goodness of data fitting. Since learning γ_m involves F_{m-1}, the growing of T_m thus relies on the previously built ensemble. Adaptive boosting regression trees (ABRT) [51] and gradient boosting regression trees (GBRT) [52] are two popular boosting tree ensembles. For the more detailed discussion about ABRT and GBRT, please refer to [51, 50] and [52, 53], respectively.

Tree ensembles are becoming sought after in recent years mainly due to their appealing performances in the KDD cups[5]. However, they have only received limited attention in the area of traffic and travel time prediction. Leshem et al. [110] implemented an adaptive boosting ensemble to classify traffic conditions for urban roads using loop detectors data. In their study, RF ensembles are used as the weak learners instead of decision trees. Day-of-week, loop detector passing time, green

[5]The KDD cup is the world's leading annual data mining and knowledge discovery competition organised by ACM Special Interest Group on Knowledge Discovery and Data Mining.

light duration, vehicles counts per signal cycle, and occupancy are used as the input features. The results show that missing data has a little impact on the prediction performance. Hamner [70] implemented an RF ensemble to predict travel time. In his study, a modular based model fitting approach is applied. That is, the training and test set are first partitioned into several subsets based on the pre-identified traffic contexts; then a 100-tree RF ensemble is fitted to each subset. To make the prediction on a test input, the corresponding model is selected based on the traffic context of the test input. Zhang et al. [235] applied RF and GBRT to predict travel times for highways with short prediction horizons (5 to 30 minutes ahead). Except for those commonly used date-time indicators (e.g., departure time interval index, day-of-week, week, month), travel time observations from previous departure time intervals and their differences are used as the input features. The results show that the travel time observation from last departure time interval is the most important feature. Their results also show that the GBRT approach yields slightly better results than the RF and ARIMA approaches. Gal et al. [54] compared several tree ensembles including extremely randomised trees, RF, ABRT, GBRT and the combined versions with the snapshot method for bus travel time prediction using bus trip log data. Their study results show that the performances of the RF and GBRT models fluctuate from time to time in a day which is probably due to the variation of traffic flow. We [114] also applied GBRT to perform travel time prediction for freight transportation using vehicles' trajectory data. The result shows that GBRT is able to handle sparse trip data. The results are appealing.

Tree ensembles have some remarkable advantages: (1) they work very well with mixed-type input features; (2) they are robust to outliers and insensitive to monotone feature transformations; (3) they can handle redundant features and missing values; (4) they are fast to build and easy to scale; (5) the resulted models are 'white-boxes' so that they are easy to interpret. Thus, tree ensembles are considered the most qualified 'off-the-shelf' data-driven methods [73]. However, fitting a tree ensemble can be slow, depending on the training data size. To reduce the model

fitting time, we can fit trees in parallel, but such parallelisation strategy is not easy to apply for boosting tree ensembles.

3.3.8 Distance Based Methods

Learning is about generalising from training data to unseen data by exploiting the similarity between them [47]. Kernel methods use kernel matrix to describe the similarity. For tree ensembles, data in one node are considered similar to each other. Such similarity also can be measured by '*the distance*' between them. In this section, some popular distance based data-driven methods are discussed.

K-nearest Neighbours Algorithms

K-nearest neighbours algorithm (k-NN) is one of the most popular distance-based methods due to its simplicity and good performance. k-NN is mainly used for classification or as a smoothing function for regression analysis. Recall that we have data $\mathcal{D}$ consisting of N input vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ of dimension d ($d \geq 1$) and the corresponding output vector $\mathbf{y} = \{y_i\}_{i=1}^N$, $\mathbf{X} \in \Re^d$, $\mathbf{y} \in \Re$. In this case, y_i is the class label corresponding to the input $\mathbf{x}_i$. To predict the class label given a new test input $\mathbf{x}_*$, we first find k training instances that are closest to $\mathbf{x}_*$. Then, the class is temporally labelled as the class of the majority. We can use Euclidean distance (i.e., the Minkowski distance of order 2) to measure the distance between $\mathbf{x}_*$ and a neighbouring training instance $\mathbf{x}_k$:

$$Dis(\mathbf{x}_*, \mathbf{x}_k) = \sqrt{\sum_{d=1}^{D} (\mathbf{x}_k^d - \mathbf{x}_*^d)^2} = \sqrt{(\mathbf{x}_k - \mathbf{x}_*)^T (\mathbf{x}_k - \mathbf{x}_*)} \tag{3.55}$$

We can also use other distance metrics (e.g., Manhattan distance, Hamming distance) to measure the distance, depending on the type of input features. Cross validation (CV) is usually used to fine a good setting of k. To apply k-NN for regression problems, local estimation method (LEM) is needed. LEM can be simply

a mean function or a linear regression model.

Several studies have been found using k-NNs for traffic and travel time prediction. Smith et al. [177] applied k-NN to predict traffic flow. You and Kim [227] used k-NN to find the most similar cases in historical data in their hybrid travel time prediction model. Bajwa et al. [13] used k-NN to predict travel time for highways. In their study, k is optimised using GA. The results show that the k-NN approach performs well for peak prediction. Robinson et al. [166] investigated the performance of k-NN for urban roads travel time prediction. Different choices of k, different distance metrics, and different local estimation methods are examined in this study. The results show that (1) k-NN is not particularly sensitive to the choice of distance metric; (2) median and local weighted scatter plot smoothing regression are better than mean and linear regression for LEM; (3) the choice of k is correlated with the historical data size. The results also show that the k-NN approach outperforms linear regression and ANN. Zhang et al. [233] applied k-NN for traffic flow prediction. A weighted average function is selected for LEM.

There are several other studies modify the standard k-NN for improve the prediction performance. Kim et al. [100] incorporated a pattern recognition technique into k-NN to predict traffic flow. In their study, k-NN is modified to search the most similar time series patterns rather than search the most similar training instances. The pattern is constructed based on a series of past observations. The results show that the modified k-NN outperforms the standard k-NN, and increasing k can gradually improve the prediction performance. Jiwon et al. [142] also proposed a pattern-matching k-NN to predict travel time for highways. In their study, the data is collected from a Vehicle Detection System (VDS) and an Automatic Toll Collection System (ATCS). The proposed k-NN has two pattern-matching steps: (1) search the most similar cases by matching the current estimated travel times to the historical travel times estimated using the ATCS data; (2) search the best matched case by matching the most similar cases from Step (1) to the historical VDS (occupancy) data. The results show that making use of both the ATCS and

VDS data is beneficial to the prediction for both short and long-stretch roads. Hong et al. [81] proposed a hybrid multi-metric k-NN (HMMKNN) to predict traffic flow. In their hybrid k-NN, the weights in LEM are estimated using a Gaussian kernel. Unlike the standard k-NN that the distance metric is a fixed static metric, their k-NN uses a dynamic distance metric approach. That is, the distance metric is solved (dynamically) by minimising a cost function which regularises the penalties for small and large distance data instances. This learning approach is also known as *the large margin nearest neighbour*.

k-NNs are very easy to use and fast to run. No training process is needed. But they are very sensitive to the quality of the data. The size of data also can have a significant impact on the performance. Although several studies have shown that k-NN can outperform ANN, most of these ANNs are single hidden layer MLPs without careful configuration and validation.

Distance Based Clustering Algorithms

Clustering algorithms are widely used for unsupervised learning[6]. A good clustering algorithm can partition data into coherent groups or clusters that two data instances from the same cluster have more in common than two data instances from different clusters [47]. The K-means clustering algorithm (K-means) [131] is one of the most popular distance based clustering (or partitional clustering) algorithms. In K-means, a cluster can be represented as the mean (or centroid) of the data instances assigned to the cluster. Let $\mathcal{D}$ denote the data consisting of N input vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ of dimension d ($d \geq 1$) but no corresponding output vector, $\mathbf{X} \in \Re^d$. K-means can be summarised as follows [167]:

1. Randomly initialises K clusters (or vectors) $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_K$.

2. For each data instance $\mathbf{x}_i$, find $\boldsymbol{\mu}_k$ that is closest to $\mathbf{x}_i$. The distance can be

[6]Unsupervised learning algorithms tend to draw inferences from data that has no output vector but only input vectors.

measured using e.g., Euclidean distance metric
$\sqrt{(\mathbf{x}_i - \boldsymbol{\mu}_k)^T(\mathbf{x}_i - \boldsymbol{\mu}_k)}$.

3. If $\mathbf{x}_i$ is assigned to cluster k, $z_{i,k} = 1$, otherwise $z_{i,k} = 0$.

4. If all the cluster assignments remain unchanged, the algorithm stops.

5. Update each cluster $\boldsymbol{\mu}_k$: $\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} z_{i,k} \cdot \mathbf{x}_n}{\sum_{i=1}^{N} z_{i,k}}$.

6. Return to 2.

Several studies have been found apply K-means for traffic and travel time prediction. Sohr et al. [181] used K-means for traffic speed prediction. In their study, the smoothed traffic speed data is first partitioned into six clusters using K-means, based on time-to-time and day-to-day traffic speed behaviours. Then, the centroids of the clusters are used for the prediction. Nath et al. [146] proposed a customised K-means for travel time prediction. In their approach, only two clusters are formed. The Manhattan distance is used as the distance metric to measure the distance between data instances and centroids of the two clusters. The novelty of their K-means approach is that the centroids of the two clusters are fixed before assigning data instances to the clusters. In this case, the travel time prediction based on two clusters is more stable when compared with the standard K-means. Elhenawy et al. [41] applied K-means for travel time prediction. In their study, both the training and test data are first partitioned into five clusters using K-means. Then, a GA based model is fitted to each cluster and is used to predict travel times on the corresponding partitioned test data. Other distance based clustering algorithms such as Gaussian mixture [226] and Fuzzy c-means clustering [230] also have been applied for traffic and travel time prediction. However, they are much less popular than K-means.

Distance based clustering algorithms are also very easy to use and fast to run. However, their performances are often not very satisfying when compared with other

types of data-driven methods such as k-NNs, SVR and ANNs. Thus, these algorithms are often used to partition data into several clusters so that context-aware dynamic travel time prediction methods can be developed. One shortcoming of distance based clustering algorithms is that the random initialisation of centroids yields a different result for each run. This issue can be addressed by fixing the centroids before assigning training instances to the clusters. However, this solution can greatly reduce the generality of the algorithms.

3.3.9 Hybrid Approaches

Generally, a hybrid approach combines two or more learning algorithms. Various hybrid methods have been proposed for traffic and travel time prediction. Sometimes, a traffic flow data can be a very complex time series. It can be very difficult to handle using single-model TSA approaches. As a result, many hybrid TSA approaches have been developed. One popular approach is combining an ARIMA model with a GARCH model [95, 191, 67, 238]. In this case, the ARIMA model is used to model the conditional mean, and the GARCH model is used to model the conditional variance of the given time series. SARIMA model can be used as the ARIMA model if there is periodic variability in the time series. Other hybrid TSA approaches also have been found in the literature, e.g., the VAR-GARCH approach [221], the ARIMA-SV approach [236].

KF based approaches are easy to implement and fast to run, which makes them feasible for online prediction tasks. However, the performances of these approaches are often not very appealing. That is mainly because the KF gain based observation prediction model is too simple and not robust to outliers. Thus, many studies use more advanced learning algorithms such as SVR or ANNs to predict the observations in KF [29, 223, 228, 122].

Traffic or travel time prediction for large-scale road network often requires a large amount of data. Having too many input features significantly increases the

computational cost, and it does not always yield a better result due to the interactive and combinatorial effects of input features. Thus, several studies [220] utilise ANNs (e.g., CNNs, RNNs) to select more representative input features and then feed them into a regression model to perform prediction tasks. The resulted approaches are usually referred to as the hybrid ANN approaches.

Hybrid approaches also have been found to develop adaptive (or context-aware) travel time prediction models [27, 104, 242, 41]. In these approaches, the training data is first partitioned into several clusters using distance based clustering algorithms. Then, a prediction model is fitted to each cluster using advanced learning algorithms such as ANNs. To make the prediction on a test input, the best prediction model is selected based on the similarity measure between the test input and the centroids of those clusters.

Many studies incorporated an advanced learning algorithm to improve the quality of model training. For example, several studies have been found use KF to solve state-space models [121, 175, 119]. In this case, the state-space model parameters and the training errors are treated as the state and observations of the training process, respectively. Other studies have been found use GA to improve the converging training for ANNs [2, 241, 123, 206, 98]. The results show that the genetically trained ANNs outperform the ANNs trained using the traditional gradient based back-propagation approach.

Several studies focus on combining the results of multiple single models of different learning algorithms. Such combination can be done using simple techniques such as weighted mean or weight average [137, 139], or more advanced techniques such as Bayesian analysis [240, 196]. The main difference between this type of hybrid approach and those reviewed earlier is that the learning algorithms are not coupled to each. Thus, they are more flexible to use. Some researchers also refer this type of hybrid approach as the ensemble approach. Note that these ensembles are different from those natural-born ensembles in Section 3.3.7. The later ones assemble multiple single models of the same type. The overall bias and variance have been taken

into account when designing the assembling strategies.

In general, a hybrid approach tends to make up for the shortcomings of a single-model approach by incorporating multiple learning algorithms, which makes the learning process more complicated and more difficult to implement.

3.4 Travel Time Prediction for Freight Transportation

The existing travel time prediction studies can be mainly summarised into the following three types:

1. Aggregated departure travel time prediction studies which mainly use traffic flow data (e.g., traffic speed, flow rate, occupancy) as the inputs.

2. Aggregated departure travel time prediction studies which mainly use probe vehicle data (mainly trajectory data) as the inputs.

3. Individual departure travel time prediction studies for a particular vehicle type, e.g., bus.

Freight transportation has been rapidly developed due to the rise of consumption-driven lifestyle and the acceleration of economic globalisation in recent years. For freight transportation service providers, accurate travel time prediction not only helps make better transportation service tracking, but also helps make better service planning and scheduling. There is, however, limited research in present focus on travel time estimation and prediction for freight transportation, which may be quite different from the passenger vehicles. Zhao et al. [239] estimated container truck travel time using a univariate statistical distribution. Wang et al. [213] proposed a hybrid approach to predict truck travel time on highways. In their study, the traffic volumes and trucks' trajectory data are utilised to generate traffic density data. The traffic density data is first partitioned into several clusters using K-means. A

mean speed is estimated for each cluster. Then, link travel time can be calculated by dividing the link length by the corresponding mean speed estimate. At last, the travel time of the target road segment can be calculated by summing all the link travel times. The results show that the proposed hybrid approach outperforms two delay model based approaches.

These limited studies are relatively simple and lack of comparison with more advanced data-driven methods. Many freight transportation service providers own a huge amount of trajectory data of their fleets, but they are far away from being effectively utilised. Very few studies provide in-depth discussions on how freight vehicles' trajectory data can be utilised for travel time prediction for freight transportation, which can be very useful for fleet task scheduling and performance evaluation. As was mentioned in [94], freight vehicles' trajectory data has not received adequate attention from researchers.

3.5 Summary

In this chapter, we provide a comprehensive review on the existing travel time prediction methods. These methods can be classified into model based methods and data-driven methods. Model based methods require pre-defined models based on some assumptions of traffic behaviours. Explicitly modelling the relation between latent influencing factors and observed travel times can be difficult in many cases, and the model performance can be poor due to inaccurate assumptions. Although these models may be calibrated to reproduce many traffic situations, they require data assimilation techniques to synchronise the internal model variables (e.g., densities or other time-varying parameters or inputs) with the real-life data [202]. The synchronisation process can be difficult to achieve due to the data availability issue. On the contrary, data-driven methods require no pre-defined models, and the model (both the model structure and parameters) can be learned only use data. Various types of data-driven methods have been reviewed. Both the advantages

and disadvantages are discussed. Table 3.2 summarises the the characteristics of these data-drive methods. The characteristics are chosen based on Table 10.1 in [73, p.351].

Table 3.2: Characteristics of different data-driven methods*. Key: P = poor, F = fair, G = good.

Characteristics	LR	L-TSA	NL-TSA	KF	ANN	Kernel	DTE	DIST
Handling of multi-dimensionality inputs	F	P	P	P	G	G	G	F
Natural handling of mixed-type inputs	F	P	P	P	P	P	G	P
Handling of missing values	P	P	P	P	P	F	G	G
Robustness to outliers in inputs	P	P	P	P	P	F	G	F
Insensitive to monotone transformations of inputs	P	P	P	P	P	P	G	P
Scale to large data	P	P	P	P	F	P	G	P
Ability to deal with irrelevant inputs	P	P	P	F	P	P	G	P
Ease of implementation	G	G	F	G	F	F	F	G
Ease of training difficulty	G	G	F	G	P	F	F	G
Interpretability	G	F	F	P	P	P	G	G
Predictive power	P	P	F	F	G	G	F	P
Fast to run	G	G	F	G	F	F	F	G

* LR - linear regression model; L-TSA - linear time series analysis approach; NL-TSA - nonlinear time series analysis approach; KF - Kalman filter based approach; ANN - artificial neural network; Kernel - kernel method; DTE - decision tree ensembles; DIST - distance based methods.

A good data-driven method should not only have good prediction power but also is easy to implement, fast to run and has good model interpretability. The review in this chapter has shown that ANNs is a very popular choice for travel time prediction in the recent decade due to their good prediction powers. However, training an ANN can be difficult and time-consuming. Also, most of the developed ANNs are black boxes which are difficult to interpret. Thus, exploring the causal effects between the

inputs and output becomes challenging. DNNs have even better prediction powers than 'shallow' ANNs. But training a DNN is computationally more expensive. It can be an overkill if the given travel time prediction problem does not require deep learning at all.

Ensemble methods assemble multiple single models to make predictions. They are generally more robust than single-model approaches, which is mainly because that both the bias and the variance of the model can be structurally reduced. The assembled single models can be of different types or the same type. They can be fitted independently, or dependently w.r.t each other. Decision tree ensembles assemble multiple decision trees where each tree is grown to a dataset that is drawn from training data with or without replacement. They work very well with multi-dimensional mixed type inputs; they are robust to outliers, redundant inputs and missing values; and the resulted models are easy to interpret. Although they are very popular in the data mining community due to their appealing performances in many world-famous data mining competitions, they have not received adequate attention in the area of traffic and travel time prediction.

Despite the comprehensive review on the data-driven methods for travel time prediction. This chapter also provides an overview of travel time prediction studies for freight transportations. Although freight transportation plays a very significant role in the development of economy nowadays, there is, however, insufficient research in travel time prediction for freight transportation. Also, many freight transportation service providers own a huge amount of trajectory data of their fleets, but they are far away from being effectively utilised.

In conclusion, there are great potentials to:

1. perform travel time prediction for freight transportation using ensemble methods, by utilising freight vehicles' trajectory data;

2. develop an in-depth understanding of how the selected ensemble methods can be effectively applied for general travel time prediction problems.

Chapter 4

Freight Vehicle Travel Time Prediction by Sparse Gaussian Process Regression

4.1 Introduction

Various types of data-driven travel time prediction methods have been discussed in the last chapter. Many studies adopted ANNs due to their appealing prediction performance. However, these ANNs often require careful data preprocessing (e.g., feature selection, monotonic feature value transformation, imputation) and model fitting processes. The model fitting process can be very time-consuming (with traditional cross validated grid search approach), depending on the structural complexity of the given network and the amount of the data used for training. Also, most of these ANN approaches are single-model approaches which potentially have a large variance. One way to address this large variance issue is using ensemble approaches.

Gaussian process regression (GPR) is a Bayesian ensemble which assembles latent functions as a Gaussian distribution (i.e., Gaussian distribution over functions) with a continuous domain, e.g., time or space. GPR has been widely used in the

areas such as facial recognition [30, 72] and kinematic control [65, 209, 132]. GPR models is very powerful on modelling the data with complex correlations. Like other kernel methods, GPR models can be optimised exactly given the values of their hyperparameters, which yields a good balance between the data fitness and smoothing. Thus, they perform very well on small datasets (e.g. 1000 or less) because of this well-tuned smoothing. Few studies have been found apply the full GPR (i.e., the model is fitted using full training set) for traffic or travel time prediction. The main problem of the full GPR is that the inference becomes too costly if it scales to large datasets (i.e., > 2000). In this chapter, a sparse GPR (SGPR) is implemented to predict travel time for freight vehicles. That is, instead of using full training set for model fitting, only a small part of the training set is used. As a result, the computational cost can be significantly reduced.

The rest of this chapter is organised as follows: Section 4.2 provides an overview of GPR and SGPR; Section 4.3 introduces how the data is prepared; Section 4.4 explains the experiments including input features, tasks, performance evaluation metric, and configuration for SGPR; Section 4.5 provides the results and analysis; the discussions and conclusions are presented in Section 4.6.

4.2 Methodology

4.2.1 Gaussian Process Regression

Travel time prediction problem can be seen as a regression problem. Assume we have a dataset $\mathcal{D}$ consisting of N input vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ of dimension d ($d \geq 1$) and the corresponding output vector $\mathbf{y} = \{y_i\}_{i=1}^{N}$; $\mathbf{X} \in \Re^d$; $\mathbf{y} \in \Re$. Our task is finding the latent travel time function f that $y_i = f(\mathbf{x}_i) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. GPR projects the original input space to a higher dimensional function space and assumes $f_i \doteq f(\mathbf{x}_i)$ is a random variable corresponding to the case $(\mathbf{x}_i, y_i)$. Let

$\mathbf{f} = \{f_i\}_{i=1}^N$, GPR specifies the prior distribution over $\mathbf{f}$ as:

$$\mathbf{f} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}) \tag{4.1}$$

where $\mathbf{m} = [m(\mathbf{x}_1), m(\mathbf{x}_2), ..., m(\mathbf{x}_N)]^T$ is the vector of mean functions and $K = K(\mathbf{X}, \mathbf{X})$ is the $N \times N$ covariance matrix which can be expressed as:

$$K(\mathbf{X}, \mathbf{X}) = \begin{bmatrix} k(\mathrm{x}_1, \mathrm{x}_1) & \cdots & k(\mathrm{x}_1, \mathrm{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathrm{x}_N, \mathrm{x}_1) & \cdots & k(\mathrm{x}_N, \mathrm{x}_N) \end{bmatrix} \tag{4.2}$$

where $k(\mathrm{x}, \mathrm{x}')$ is the covariance (or kernel) function that describes the similarity between any pair of data points. The formal definition of GP can be found in [161].

Definition 10 *A **Gaussian process (GP)** is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

For notational simplicity, we set $m(\mathbf{x}) = 0$. Hence, a GP is fully specified by K. The prior distribution also can be written as $p(\mathbf{f}|\mathbf{X})$ because it is aware of the locations (indices) of $\mathbf{X}$ while $\mathbf{y}$ is not observable. Given $\mathbf{y}$, the posterior distribution can be obtained by applying Bayes' theorem:

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) = p(\mathbf{f}|\mathcal{D}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{f})p(\mathbf{f}|\mathbf{X})}{p(\mathbf{y}|\mathbf{X})} \tag{4.3}$$

where $p(\mathbf{f}|\mathbf{X})$ is the prior distribution; $p(\mathbf{y}|\mathbf{X}, \mathbf{f})$ is the likelihood which is the probability of having $\mathbf{y}$ given $\mathbf{X}$ and $\mathbf{f}$; $p(\mathbf{y}|\mathbf{X})$ is the marginal likelihood of $\mathbf{y}$ which can be expressed as:

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{X}, \mathbf{f})p(\mathbf{f}|\mathbf{X})d\mathbf{f} \tag{4.4}$$

where $\boldsymbol{\theta}$ is the vector of hyperparameters (i.e., the parameters of those covariance functions in K).

The prior distribution is a Gaussian distribution which can be calculated using [161, pp.18-19]:

$$\log p(\mathbf{f}|\mathbf{X}) = -\frac{1}{2}\mathbf{f}^T K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{N}{2}\log 2\pi \tag{4.5}$$

The log marginal likelihood can be expressed as follows [161, pp.18-19]:

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T[K + \sigma_\epsilon^2 I]^{-1}\mathbf{y} - \frac{1}{2}\log|[K + \sigma_\epsilon^2 I]| - \frac{N}{2}\log 2\pi \tag{4.6}$$

where I is an identity matrix of size $N \times N$. The only term that involves the observed outputs is the data-fit term $-\frac{1}{2}\mathbf{y}^T[K + \sigma_\epsilon^2 I]^{-1}\mathbf{y}$. $\frac{1}{2}\log|[K + \sigma_\epsilon^2 I]|$ is the complexity penalty term which depends only on the covariance function and the inputs. $\frac{N}{2}\log 2\pi$ is a normalization constant. The optimal values of $\boldsymbol{\theta}$ can be obtained by maximising the log marginal likelihood (or minimising the negative log marginal likelihood), where both the data fitness and the model complexity can be taken into account. To predict the outputs of N_* new test cases $\mathbf{X}_*$, let $\mathbf{f}_*$ denote the vector of predicted outputs for $\mathbf{X}_*$, we first compute the $N + N_*$ dimensional joint prior $p(\mathbf{f}_*, \mathbf{y})$, which can be expressed as [161, pp.15]:

$$p(\mathbf{f}_*, \mathbf{y}) = N\left(0, \begin{bmatrix} K + \sigma_\epsilon^2 I & K_* \\ K_*^T & K_{**} \end{bmatrix}\right) \tag{4.7}$$

where K_* and K_{**} denote $K(\mathbf{X}, \mathbf{X}_*)$ and $K(\mathbf{X}_*, \mathbf{X}_*)$, respectively. The posterior distribution over the joint functions can be obtained by conditioning the joint prior distribution because we need to restrict this joint prior distribution to contain only the functions that agree with the observed data. The posterior distribution can be

expressed as [161, pp.16]:

$$f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}(\bar{\mathbf{f}}_*, \mathbb{V}[\mathbf{f}_*]) \text{ where}$$

$$\bar{\mathbf{f}}_* := K_*^T[K + \sigma_\epsilon^2 I]^{-1}\mathbf{y} \tag{4.8}$$

$$\mathbb{V}[\mathbf{f}_*] := K_{**} - K_*^T[K + \sigma_\epsilon^2 I]^{-1}K_* \tag{4.9}$$

If $N_* = 1$ (i.e., only one test case), the prediction model can be rewritten by replacing K_{**} with $k(\mathbf{x}_*, \mathbf{x}_*)$ in Equation (4.9). Note that the predicted variance $\mathbb{V}[\mathbf{f}_*]$ in Equation (4.9) only depends on the inputs. This variance is the difference between the prior covariance matrix and the product of the other three covariance matrices. Computing $\bar{\mathbf{f}}_*$ and $\mathbb{V}[\mathbf{f}_*]$ requires the inversion of $\mathbf{K}$ (using Cholesky decomposition), which has the computational complexity of $\mathcal{O}(N^3)$. As a result, the inference using GPR can be slow if n is more than a few thousands. Also, For detailed information about how $\bar{\mathbf{f}}_*$ and $\mathbb{V}[\mathbf{f}_*]$ can be computed, please refer to Algorithm 2.1 in [161, pp.19].

4.2.2 Sparse Gaussian Process Regression

The GPR in the previous section is also known as the full GPR as it uses full training data. The main issue for the full GPR is its high computational cost. To overcome this limitation, many researchers have suggested using a subset of training data to approximate the exact prior distribution instead of using full training data. This approach is known as the *sparse* approximation. The GPR with the sparse approximation is also known as the sparse GPR (SGPR). The computational cost of SGPR is a lot cheaper than the full GPR.

First, we introduce an *inducing set* consisting of m inducing inputs $\mathbf{X}_u$ and the corresponding output vector $\mathbf{u} = \{u_i\}_{i=1}^m$. The joint distribution $p(\mathbf{f}, \mathbf{f}_*)$ can be

obtained by integrating out $\mathbf{u}$ from $p(\mathbf{f}, \mathbf{f}_*, \mathbf{u})$ [160]:

$$p(\mathbf{f}, \mathbf{f}_*) = \int p(\mathbf{f}, \mathbf{f}_*, \mathbf{u}) d\mathbf{u} = \int p(\mathbf{f}, \mathbf{f}_*) p(\mathbf{u}) d\mathbf{u} \tag{4.10}$$

where $p(\mathbf{u}) = \mathcal{N}(0, K(\mathbf{X_u}, \mathbf{X_u}))$.

The above one is the exact expression for the joint prior. SGPR approximates the exact joint prior by assuming that $\mathbf{f}$ and $\mathbf{f}_*$ are *conditionally independent* given $\mathbf{u}$. Thus, we have [160]:

$$p(\mathbf{f}, \mathbf{f}_*) \simeq q(\mathbf{f}, \mathbf{f}_*) = \int q(\mathbf{f}|\mathbf{u}) q(\mathbf{f}_*|\mathbf{u}) d\mathbf{u} \tag{4.11}$$

where $q(\mathbf{f}|\mathbf{u})$ and $q(\mathbf{f}_*|\mathbf{u})$ are the approximate *training conditional* and *test conditional* of the exact conditionals. They are known as the approximate *inducing conditionals*. The exact inducing conditionals can be expressed as follows:

$$p(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{u}} K_{\mathbf{uu}}^{-1} \mathbf{u}, K - Q) \tag{4.12}$$

$$p(\mathbf{f}_*|\mathbf{u}) = \mathcal{N}(K_{*\mathbf{u}} K_{\mathbf{uu}}^{-1} \mathbf{u}, K_{**} - Q_{**}) \tag{4.13}$$

where $K_{\mathbf{u}}$, $K_{\mathbf{uu}}$ and $K_{*\mathbf{u}}$ denote $K(\mathbf{X}, \mathbf{X_u})$, $K(\mathbf{X_u}, \mathbf{X_u})$ and $K(\mathbf{X}_*, \mathbf{X_u})$, respectively. The notation $Q_{\mathbf{ab}}$ is the approximation of true $K_{\mathbf{ab}}$, which can be expressed as:

$$Q_{\mathbf{ab}} := K_{\mathbf{au}} K_{\mathbf{uu}}^{-1} K_{\mathbf{ub}} \tag{4.14}$$

Note that $\mathbf{f}$ and $\mathbf{f}_*$ are indirectly related through $\mathbf{u}$. Quionero-candela et al. [160] explained that the different computationally efficient algorithms proposed in the literature corresponds to different *additional assumptions* of the two approximate inducing conditionals of the integral in Equation (4.11). In general, we only have to invert $K_{\mathbf{uu}}$ of size $m \times m$ in SGPRs. Thus, the computational complexity for the model training can be reduced to $\mathcal{O}(Nm^2)$.

Seeger at al. [171] proposed a sparse greedy approximation method for GPR. In

their study, the inducing set is selected from training data, and the true likelihood $p(\mathbf{y}|\mathbf{u})$ is approximated as:

$$q(\mathbf{y}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{u}}K_{\mathbf{uu}}^{-1}\mathbf{u}, \sigma_\epsilon^2 I) \tag{4.15}$$

This approximation approach is also known as the deterministic training conditional (DTC) approximation because it yields the deterministic training conditional and exact test conditional [160]:

$$q_{DTC}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{u}}K_{\mathbf{uu}}^{-1}\mathbf{u}, 0) \tag{4.16}$$

$$q_{DTC}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u}) \tag{4.17}$$

Thus, the approximated joint distribution over $\mathbf{f}$ and $\mathbf{f}_*$ can be expressed as [160]:

$$q_{DTC}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(0, \begin{bmatrix} Q + \sigma_\epsilon^2 I & Q_* \\ Q_*^T & K_{**} \end{bmatrix}\right) \tag{4.18}$$

Note that the DTC approximation uses the exact test conditional. Thus, it helps reduce the unreasonable prediction variance caused by the selection of inducing set. The posterior can be expressed as follows [160]:

$$q_{DTC}(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(Q_*^T[Q + \sigma_\epsilon^2 I]^{-1}\mathbf{y}, K_{**} - Q_*^T[Q + \sigma_\epsilon^2 I]^{-1}Q_*) \tag{4.19}$$

Snelson et al. [179] proposed another SGPR approach. In their study, the inducing set is no longer a subset of training data but a pseudo dataset. The likelihood is given by the posterior, and parameterised by the pseudo inducing set. The approximation of the exact likelihood becomes [179]:

$$q(y|\mathbf{u}) = \mathcal{N}(K_{\mathbf{u}}K_{\mathbf{uu}}^{-1}\mathbf{u}, \Lambda + \sigma_\epsilon^2 I) \tag{4.20}$$

where $\Lambda = diag(\lambda)$; $\lambda = K - Q$; K and Q are the exact and approximate covariance

matrix of the training data respectively. In this case, the approximation of the true likelihood takes into account the difference between the true covariance matrix and approximated covariance matrix.

Unlike DTC, no deterministic relation between $\mathbf{f}$ and $\mathbf{u}$ is imposed in this approach. It proposes an approximation to the training conditional as a further independent assumption. This approach is also known as the fully independent training conditional (FITC) approximation. The posterior is expressed as follows [160]:

$$q_{FITC}(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(Q_*^T[Q_{\mathbf{uu}} + \Lambda']^{-1}\mathbf{y}, Q_*^T[Q_{\mathbf{uu}} + \Lambda']^{-1}Q_*) \tag{4.21}$$

where $\Lambda' = diag(\lambda + \sigma_\epsilon^2 I)$. Inversion of Λ' is much easier and the computational complexity of the model training is also $\mathcal{O}(Nm^2)$.

Since $\mathbf{X_u}$ parameterises the approximated likelihood in both the DTC and FITC cases, $\mathbf{X_u}$ thus can be jointly learned with the hyperparameters $\boldsymbol{\theta}$ and the free noise scale σ_ϵ. The feasibility of the joint optimisation over $(\mathbf{X_u}, \boldsymbol{\theta}, \sigma_\epsilon)$ has been proven in two studies [171, 179]. Titsias [190] argued that because the joint prior distribution changes due to the change of the inducing set, it makes continuous optimisation of the likelihood w.r.t. $\mathbf{X_u}$ unreliable to approximate the exact GP. And since the likelihood is strongly parameterised with $\mathbf{X_u}$, overfitting may happen when jointly optimising over $(\mathbf{X_u}, \boldsymbol{\theta}, \sigma_\epsilon)$. Thus, he proposed a variational learning approach for GPR. In his method, the posterior is directly approximated as [190]:

$$\begin{aligned} p(\mathbf{z}|\mathbf{y}) \simeq q(\mathbf{z}) &= \int p(\mathbf{z}|\mathbf{u})p(\mathbf{f}|\mathbf{u})\phi(\mathbf{u})d\mathbf{f}d\mathbf{u} \\ &= \int p(\mathbf{z}|\mathbf{u})\phi(\mathbf{u})d\mathbf{u} \\ &= \int q(\mathbf{z}, \mathbf{u})d\mathbf{u} \end{aligned} \tag{4.22}$$

where $\mathbf{z}$ denotes any finite set of the latent function outputs; $\phi(\mathbf{u}) = p(\mathbf{u}|y)$. The above approximation is based on the assumption that any $\mathbf{z}$ is conditionally independent from $\mathbf{f}$ given $\mathbf{u}$. In practice, it is difficult to find such $\mathbf{u}$ that are sufficient

statistics, thus $q(\mathbf{z})$ is only an approximation of $p(\mathbf{z}|\mathbf{y})$. In this case, $\phi(\mathbf{u})$ can be considered as a 'free' variational Gaussian distribution. Then the task becomes specifying ϕ and $\mathbf{X_u}$ to form the variational distribution $q(\mathbf{f})$ and the exact posterior $p(\mathbf{f}|\mathbf{y})$. Equivalently, it can be achieved by minimising a distance between the joint true posterior $p(\mathbf{f}, \mathbf{u}|\mathbf{y})$ and the joint variational posterior $q(\mathbf{f}, \mathbf{u})$. Such distance is also known as the Kullback-Leibler (KL) divergence. The minimisation of the KL divergence $KL(q(\mathbf{f}, \mathbf{u})||p(\mathbf{f}, \mathbf{u}|\mathbf{y}))$ can be done by maximising of the following variational lower bound of the true log marginal likelihood [190]:

$$F_v(\mathbf{X_u}, \phi) = \int p(\mathbf{f}|\mathbf{u})\phi(\mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{u})}{\phi(\mathbf{u})} d\mathbf{f} d\mathbf{u} \tag{4.23}$$

The lower bound can be first optimised by analytically solving for the optimal choice of ϕ. The bound after the first-step optimisation becomes [190]:

$$F_v(\mathbf{X_u}) = \log[\mathcal{N}(0, \sigma_\epsilon^2 I + Q)] - \frac{1}{2\sigma_\epsilon^2} Tr(\tilde{K}) \tag{4.24}$$

where $\tilde{K} = K - Q$. The bound contains an extra regularisation trace term $\frac{1}{2\sigma_\epsilon^2} Tr(\tilde{K})$ when compared to the approximated likelihood forms in the DTC and FITC approaches. In this case, finding the optimal value of F_v is a multi-objective optimisation task, i.e., maximising the log likelihood and minimising the trace term Tr. Tr is the total variance of the conditional prior $p(\mathbf{f}|\mathbf{u})$ which corresponds to the squared error in the training data given $\mathbf{u}$. When $Tr = 0$, we have $K = Q$, hence $\mathbf{u}$ is fully able to reproduce the full GP. Thus, $(\mathbf{u}, \boldsymbol{\theta}, \sigma_\epsilon^2)$ can be jointly continuously optimised by continuously maximising the lower bound [190]. The approximation is also known as the variational DTC (VDTC) approximation. The computational cost for the predictive posterior is also $\mathcal{O}(Nm^2)$.

Note that, the VDTC approximation is based on the assumption that $\mathbf{X_u}$ is selected by applying a gradient based optimisation algorithm. If the input space is very large, the input space for $\mathbf{X_u}$ also becomes very large and the kernel function

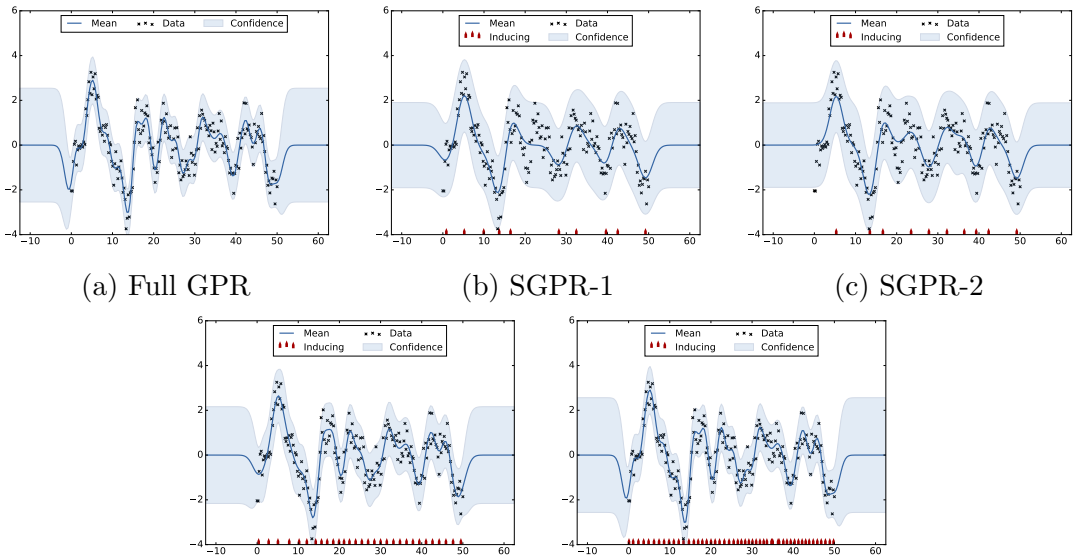
may not be differentiable w.r.t. the inputs. In this case, we can select $\mathbf{X_u}$ from the training inputs and then apply the VDTC approximation. In this study, the VDTC approximation is selected when implementing the SGPR models.

4.2.3 Inducing Set Initialisation for Sparse Gaussian Process Regression

In the VDTC approximation, when the input space is not very large, the method randomly selects $\mathbf{X_u}$ from training data input (indices). Figure 4.1 shows the example of a full GPR model and several SGPR models that are fitted on a pseudo dataset that consists of 50 input data $\mathbf{x} = \{x_i\}_{i=1}^{50}$ and the corresponding output vector $\mathbf{y} = \{y_i\}_{i=1}^{50}$. The output vector is randomly sampled from a multivariate Gaussian distribution that $\mathbf{y} \sim \mathcal{N}(0, K(\mathbf{x}, \mathbf{x}))$. The squared exponential kernel is used to construct the covariance matrix K. The inducing sets are randomly initialised. It can be seen that two SGPR models, SGPR-1 and SGPR-2, are significantly underfitted due to the poor locations of the inducing inputs. The length scales for the kernel functions in the these two SGPR models (1.684 for SGPR-1, 1.558 for SGPR-2) are larger than that for the full GPR model (1.234). This is because the KL divergence requires a larger length scale to improve the quality of the approximation.

The inducing set size can also affect the model quality. This is because using more inducing data potentially increases the chance of pinning down more good locations. As are shown in Figure 4.1d and Figure 4.1e, the SGPR models with relatively larger inducing set are less underfitted than those with smaller inducing set. But it is not to say that larger inducing set always yield a better model. The model quality can be fairly affected by the quality of the data, the selection of kernel functions and the optimisation algorithm. Also, setting a larger inducing set can significantly increase the computational cost.

Several studies have been found using k-means to initialise the inducing inputs for SGPR [78, 55, 77]. k-means++ is an improved version of k-means which augments



(a) Full GPR (b) SGPR-1 (c) SGPR-2

(d) SGPR-3 (e) SGPR-4

Figure 4.1: The examples of full GP and SGPR models.

a randomised feeding technique to improve the performance of k-means [11]. In our study, three inducing set initialisation methods are examined: random selection, k-means, and k-means++.

4.3 Data

The Port of Ningbo-Zhoushan (CNZOS) is the world's busiest port in terms of cargo tonnage. It is located in Ningbo and Zhoushan, on the coast of the East China Sea in China. Container trucks play a significant role in the intermodal freight transportation in CNZOS. Each intermodal container truck is equipped with a GPS module, which allows sending the real-time location of the truck to CNZOS's data center. As a result, a huge amount of trajectory data have been accumulated, but they are far from being effectively utilised. In this study, three road segments are selected from the intermodal drayage road network. The descriptive information of these road segments are shown in Table 4.1. The trajectory data used in this study are collected between 16th March and 30th April 2015. It has more than 7

Table 4.1: Selected road segments for the study.

Route Name	Description	Direction	Length (m)
R1	a road segment of 2nd Tong Dao	west→east	7187
R2	a road segment of 2nd Tong Dao	east→west	5847
R3	a road segment of G329	east→west	7993

million trajectory reports, each of which contains the latitude and longitude of the reported location, the instantaneous speed, the moving direction, and the date-time. The sampling rate of the trajectory data is 30 seconds. Since the traffic flow data that owned by traffic management authorities is not accessible to us. The trucks' trajectory data becomes the only data for this study. Thus, the trajectory data are used to: (1) identify all the trips for each road segment; (2) estimate travel times of all the trips for each road segment; (3) and compose input features.

4.3.1 Trip Identification and Travel Time Estimation

There is no 'off-the-shelf' trip log data available for the study. Hence, the first task is identifying all the trips for each road segment. In this study, a road segment is defined as follows:

Definition 11 *A **road segment** is part of an arterial road or national expressway that consists of multiple road links and signalised or non-signalised traffic intersections.*

A trip is defined as follows:

Definition 12 *A **trip** starts when a truck enters the road segment, and it ends when it exits the road segment.*

Because trajectory point rarely happens to be the start point or the end point of a road segment, the trip identification thus is mainly about the estimation of the entry and exit time of the road segment. A method is developed for this study to estimate the entry and the exiting time of a trip using the trajectory data. The concept is shown in Figure 4.2.

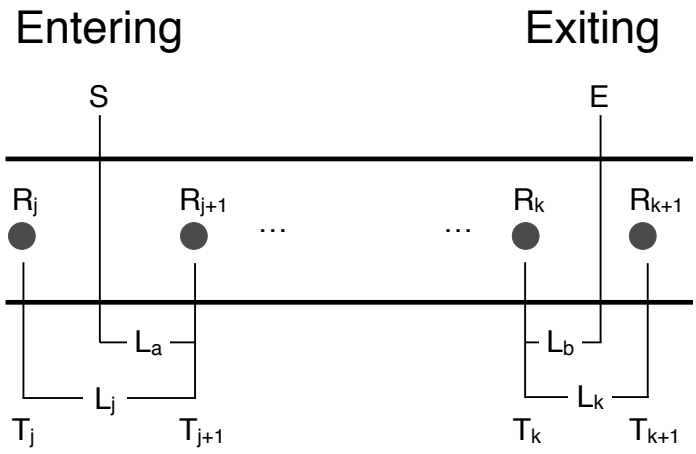


Figure 4.2: Estimation of the entry and exit time of a trip.

For each road segment, there is a start point S and an end point E. The length of the road segment is the travelled distance between S and E following the trajectory. The geographic coordinates of S and E are determined using Google Earth[1]. The length of the road segment is also measured in Google Earth. Suppose we want to identify a trip for a road segment, we observed a set of consecutive trajectory reports, denoted as R_j, R_{j+1}, ..., R_k, R_{k+1}, the corresponding locational reporting times are T_j, T_{j+1}, ..., T_k, and T_{k+1}, respectively. Let TT_a denote the travel time between S and R_{j+1}, and TT_b denote the travel time between R_k and E, the entry time and exit time of a trip can be expressed as follows:

$$T_{entry} = T_{j+1} - TT_a \tag{4.25}$$

$$T_{exit} = T_k + TT_b \tag{4.26}$$

We assume that within a short period, travel times of a container truck are equally distributed per distance unit (i.e., per meter). Thus, TT_a is proportional to the travel time between R_j and R_{j+1} and TT_b is also proportional to the travel time

[1]Google Earth is a computer program developed and maintained by Google LLC. It renders a three-dimensional representation of Earth based on satellite imagery.

between R_k and R_{k+1}. Let L_j, L_a, L_k, and L_b be the distance between R_j and R_{j+1}, S and R_{j+1}, R_k and R_{k+1}, and R_k and E respectively, the entry and exit time in Equation (4.25) and Equation (4.26) become:

$$T_{entry} = T_{j+1} - (T_{j+1} - T_j)\frac{L_a}{L_j} \tag{4.27}$$

$$T_{exit} = T_k + (T_{k+1} - T_k)\frac{L_b}{L_k} \tag{4.28}$$

Then, the travel time between S and E can be simply calculated as:

$$TT = T_{exit} - T_{entry} \tag{4.29}$$

Note that the assumption of identical distribution of travel times over distance may not be valid if there are traffic intersections near S and E. More advanced method is needed to take into account the queuing behaviours in this case. In our case, there are no intersections near S and E.

4.3.2 Abnormal Trip Filtration

Although the above method can identify all the trips for each road segment and intercept the trajectory reports of each trip, some abnormal trajectory behaviours have been found in the extracted trips:

1. Vehicles made one or more stops at some locations for more than 5 minutes before reaching the destination. The field study has shown that these places are either gas stations, road-side garage or locations near some container yards.

2. Two consecutive trajectory reports are too far away that the expected mean speed is faster than 120km/h.

3. Most of the trajectory reports are missing in the trip.

The trips in the first case should be considered a multi-stop trip which is no longer a valid trip in our study. In the second and third cases, the trajectory data are

corrupted due to either software or hardware errors (including the communication network failures), which can not be used for the later study. Hence, an electronic fence is developed for this study. A set of rules is defined for the abnormal trip filtration:

1. A trip that has one or more trajectory reports outside the fence should be filtered out.

2. A trip in which any pair of two consecutive reports has a distance larger than 1000 meters should be filtered out.

3. A trip in which any pair of two consecutive reports has a time gap larger than the sampling rate (30 seconds) should be filtered out.

4. A trip that has estimated travel time larger than $(L/\bar{v}_{min})$ should be filtered out. Here L is the length of a given road segment, $\bar{v}_{min} = 5$km/h. That means the average speed should be faster than 5km per hour.

After applying the electronic fence with the above filtration rules, the number of valid trips for R1, R2 and R3 are 1143, 1643 and 1320, respectively.

4.4 Experimentation

4.4.1 Prediction Tasks

Two types of prediction tasks are used for this study: prediction before departure (BDP) and prediction for ongoing trips (OGP). BDP predicts travel time before a trip starts. Because there is no other real-time information available, BDP generalises future travel times only based on historical data. OGP predicts travel time for an ongoing trip before it ends. OGP can be performed continuously as long as the trip is not ended. Real-time information is usually incorporated into the model to enhance the prediction power. For each road segment, the last 200 trips are used for the performance evaluation, and the rest trips are used for training.

4.4.2 Features

Three types of input features are used in this study: date-time indicator, historical descriptive statistic and real-time data. A date-time indicator describes the date-time relevant information such as departure time, day-of-week, month, year, or holiday. Date-time indicators helps to find the most similar cases in the historical data. Since there are no public holidays in the test data and the time coverage is only one and a half month, it does not seem necessary to have the month, year, and holiday identifiers in the feature columns. Thus, departure time and day-of-week are the only date-time indicators for this study.

A historical descriptive statistic is a descriptive statistic (e.g., mean, median, percentiles or range measures) on the historical data, given a departure time interval. In this study, a historical mean travel time profile is built for each truck. Because the trip data is temporally sparse, we set the aggregation time window to 15 minutes for the calculation of the mean travel times in each profile. As a result, each profile is comprised of $(60/15) \times 24 = 96$ mean travel times where each corresponds to a departure time interval. Unfortunately, some profiles are not fully filled due to the sparsity of the trip data. Hence, certain rules are applied to ensure that every profile is fully filled. For each road segment, given a truck and a departure time interval t:

1. If no trip is found, the mean travel time of the trips from all the other trucks that have departure time within t should be used.

2. If no trip is found given t and all the other trucks, the mean travel time of $t - 1$ (i.e., previous departure time interval) should be used.

Date-time indicators and historical descriptive statistics can be used for BDP, but the prediction performance is usually not good enough, particularly if the training data is temporally sparse. Hence, real-time data should be considered to incorporate into the model. Travel time estimations or observations from previous departure time intervals are usually used as the (lagging) real-time data. However,

such data can be missing due to the sparsity of the trip data. In this study, each truck is equipped with a GPS device, which allows the truck to send a trajectory report to the data server every 30 seconds. Thus, we can perform OGP by incorporating the received trajectory reports up to the time when a prediction is performed. Speed information is proven to be a good type of input feature for traffic and travel time prediction in the literature. Although a trajectory report contains speed information, this speed information only describes the instantaneous speed and can not represent the mean speed. In this study, the mean speed sequence (MSS) is introduced as the real-time input features for OGP. Assume a trip starts at time t_0, we want to perform the OGP T minute after the trip starts, the MSS can be expressed as

$$MSS_T = \{\hat{v}_i\}_{i=1}^{M} \tag{4.30}$$

where M is equivalent to the number of trajectory reports received within the first T minute and $\hat{v}_i$ is the ith estimated mean speed:

$$\hat{v}_i = \frac{\hat{L}_i}{t_i - t_0} \tag{4.31}$$

where $\hat{L}_i$ is the ith travelled distance, estimated using the location information (i.e., the geographic coordinates) from the received trajectory reports. For example, assume we want to perform the OGP for a trip 2 minutes ($T = 2$) after it starts, as a trajectory report is received every 30 seconds, the MSS contains four values which correspond to the mean speed estimate of the first $\frac{1}{2}$minute, 1 minute, $1\frac{1}{2}$minutes and 2 minutes, respectively.

4.4.3 Performance Evaluation Metrics

Root mean squared error (RMSE) and mean absolute percentage error (MAPE) are selected as the performance evaluation metrics. The general forms of RMSE and

MAPE are shown in Table 4.2. Since an SGPR model may yield a different result for every run due to the inducing set selection process, each prediction task is replicated for 30 rounds, and the means and the standard errors are reported.

4.4.4 SGPR Configuration and Implementation

A kernel function defines the similarity between two data inputs. Various types of kernel functions can be used for GPR including squared exponential kernel (SE), rational quadratic kernel (RQ), periodic kernel (PER), and linear kernel (LIN). SE is a very popular kernel in the existing studies. It presents a smooth transition between two neighbouring data inputs. RQ can be seen as an infinite sum of SE with multiple length scales [161]. The data visual analysis has shown that neither peak lumps nor linear uptrend or downtrend can be clearly identified for all the road segments. Hence, it is not necessary to use PER or LIN, or making a composited kernel out of them in this study. Thus, RQ with automatic relevance determination (RQ_{ARD}) is selected as the kernel function. RQ_{ARD} can be expressed as follows:

$$RQ_{ard}(\mathbf{x}, \mathbf{x}') = \sigma^2(1 + \frac{\|\mathbf{x} - \mathbf{x}'\|^T \mathbb{L}^{-1} \|\mathbf{x} - \mathbf{x}'\|}{2\alpha})^{-\alpha} \tag{4.32}$$

where σ^2 is the variance w.r.t. the function mean; α is the shape parameter; $\mathbb{L}$ is a diagonal matrix of the squares of the length scales where each length scales corresponds to an input dimension.

All the SGPR models are implemented using GPy [188], which is a cross-platform Gaussian processes framework developed and maintained by the machine learn-

Table 4.2: The general forms of RMSE and MAPE.

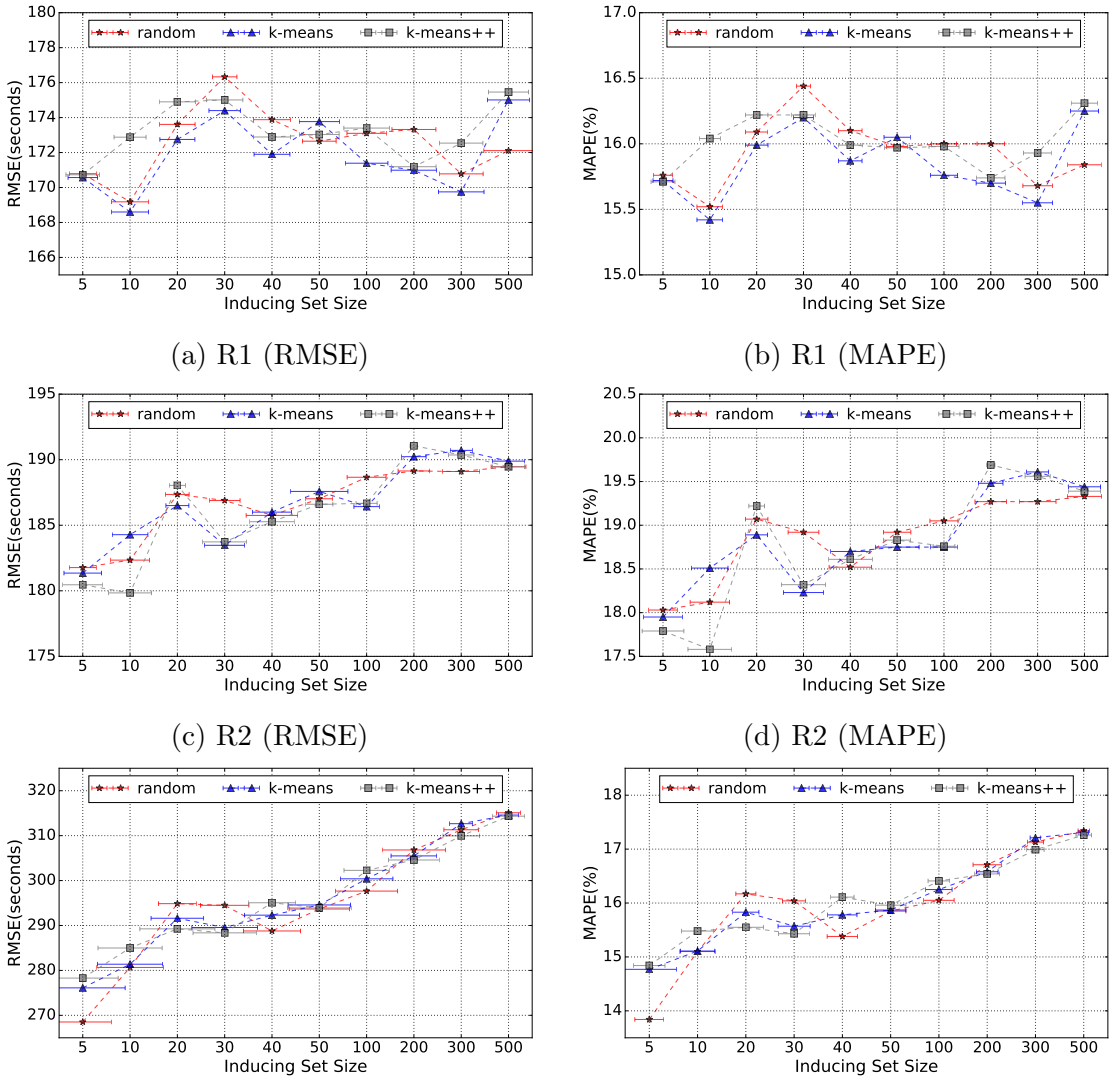
RMSE	MAPE (%)
$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$	$\frac{100}{n}\sum_{i=1}^{n}\lvert\frac{\hat{y}_i - y_i}{y_i}\rvert$

ing group from the University of Sheffield. The limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS) is selected as the optimisation algorithm to jointly optimise ($\mathbf{X_u}$, $\boldsymbol{\theta}$, σ_n^2) with the maximum iteration of 300.

4.5 Results And Analysis

4.5.1 Prediction Before Departure

Thirty SGPR models are fitted using ten different inducing set sizes and the three initialisation methods to perform the prediction before departure (BDP). Figure 4.3 shows the BDP results of the SGPR models. The horizontal lines that lie on the symbols represent the variations of the results. The variations are scaled for display purpose. In the results of R1 (see Figure 4.3a and Figure 4.3b), the errors generally increase when the inducing set size increases from 5 to 30; then they decline with some fluctuations when the size increases from 30 to 300; and a rise occurs when the size increases from 300 to 500. In the results of R2 (see Figure 4.3c and Figure 4.3d) and R3 (see Figure 4.3e and Figure 4.3f), the errors generally increase with the increase of inducing set size. The results of R1 also show that using k-means for the inducing inputs initialisation generally yields a better result than the other two methods for R1. However, this finding is not consistently observed for R2 and R3. All the initialisation methods yield fast convergence. Using k-means and k-means++ does not yield faster convergence than using the random selection. Table 4.3 shows the results of the full GPR models. It can be seen that the errors are larger for the full GPR models than those for the SGPR models. Theoretically, when the quality of the training data is decent, a SGPR model is more likely to yield a better result with the increase of inducing set size. But if the training data is not very good (i.e., contains too many poor data points), increasing the inducing set size may also increase the chance of selecting more poor inducing data. Thus, the results reveal that the input features used for the BDP tasks are not good enough to generalise



(a) R1 (RMSE)

(b) R1 (MAPE)

(c) R2 (RMSE)

(d) R2 (MAPE)

(e) R3 (RMSE)

(f) R3 (MAPE)

Figure 4.3: The BDP results of the SGPR models.

Table 4.3: The BDP results of the full GPR models.

R1		R2		R3	
RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)
181.55	17.12	193.29	20.03	320.47	17.71

future travel times and capture the nature of correlation in travel times.

4.5.2 Prediction for Ongoing Trips

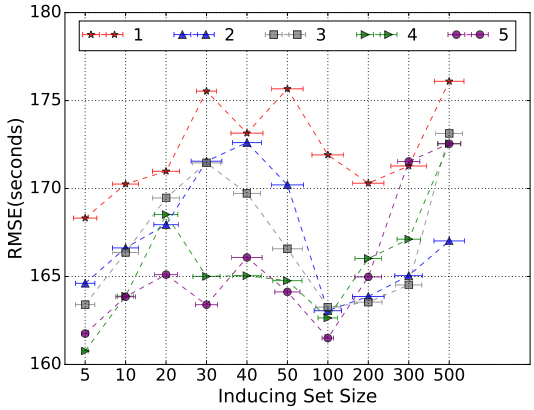
Table 4.4 shows the results of the full GPR models for the prediction for ongoing trips (OGP). The first column T indicates how soon (in minutes) the prediction is performed after a trip starts. It can be seen that adding MSS to the feature columns is beneficial to the prediction for R2 and R3, but not for R1.

Table 4.4: The OGP results of the full GPR models.

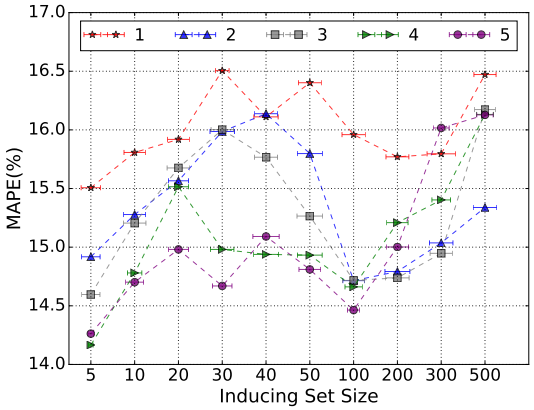
T (min)	R1		R2		R3	
	RMSE (sec)	MAPE (%)	RMSE (sec)	MAPE (%)	RMSE (sec)	MAPE (%)
1	178.99	16.90	188.97	19.09	319.96	17.61
2	179.81	16.97	184.55	18.48	314.72	17.17
3	179.61	16.95	176.61	17.48	309.52	16.41
4	178.56	16.86	169.90	16.44	307.35	16.14
5	178.30	16.83	159.99	14.45	306.09	15.81

Figure 4.4 and Figure 4.5 show the OGP results of the SGPR models for R1 and R2, respectively. It can be seen that the errors generally decrease with the increase of T, particularly when the inducing set size is below 100. Similar to the BDP results, the 'increase-decrease-increase' variation pattern can be observed for all cases. But the variance w.r.t the change of the inducing set size become much smaller when $T = 4$ and $T = 5$. Figure 4.6 shows the OGP results of the SGPR models for R3. It can be seen that the errors still generally decrease with the increase of T. Increasing the inducing set size in the case of R3 have more significant impact on the prediction accuracy than those in the cases of R1 and R2, which indicates that the training data for R3 is still not good enough to generalise the travel times for the test trips. Further experiment results show that removing the mean speed estimates from the first 3 minutes does not affect the OGP results for $T = 4$ and $T = 5$.

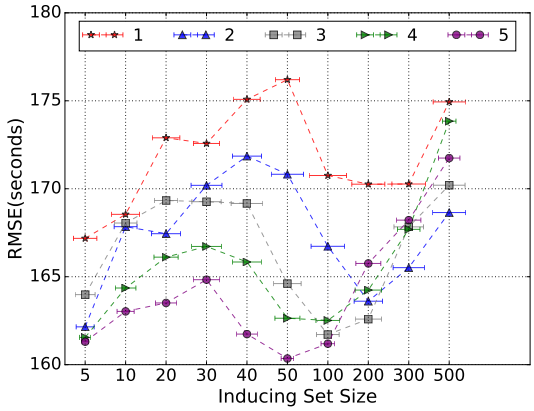
Note that although increasing T can be beneficial to the prediction, it also reduces the value of the prediction because we want to predict travel time far earlier before a trip ends. Although the quality of the training data can be improved by adding MSS in some cases, it is still not good enough for the prediction. In this case, information such as load, task type (e.g., pickup, drop-off), and task urgency



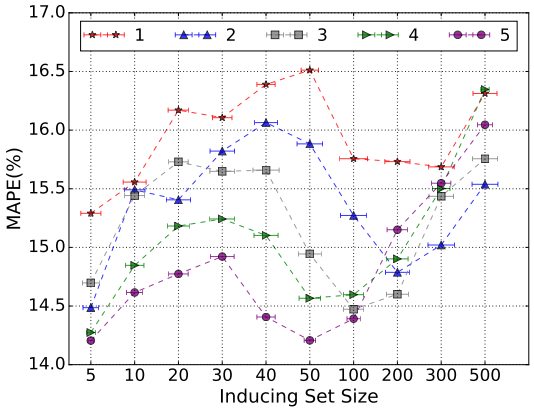
(a) SGPR-R (RMSE)



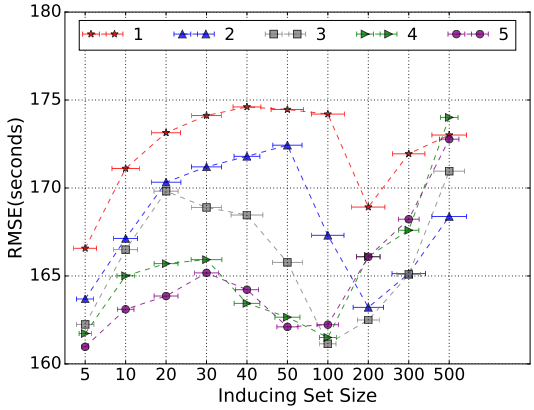
(b) SGPR-R (MAPE)



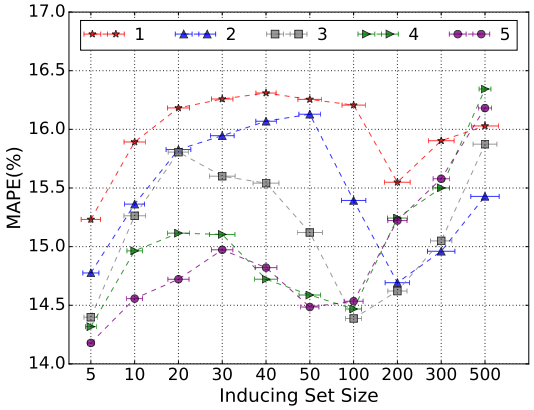
(c) SGPR-KM (RMSE)



(d) SGPR-KM (MAPE)



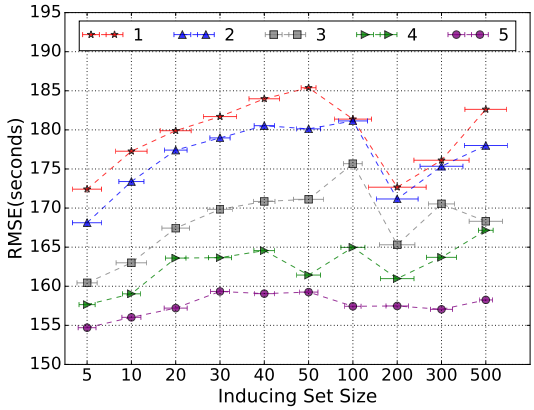
(e) SGPR-KP (RMSE)



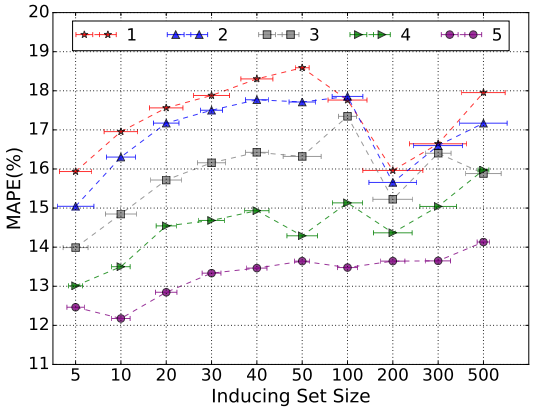
(f) SGPR-KP (MAPE)

Figure 4.4: The OGP results of the SGPR models for R1[1].

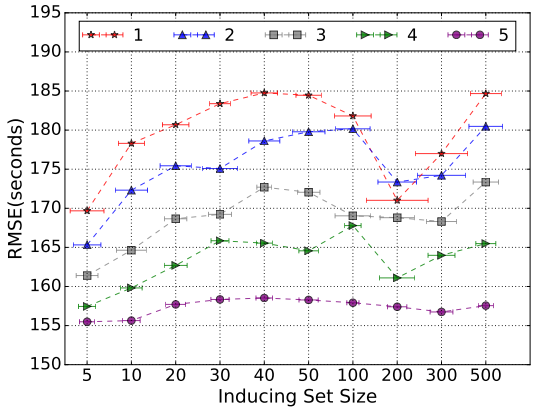
[1] SGPR-R: SGPR with random selection; SGPR-KM: SGPR with k-means; SGPR-KP: SGPR with k-means++.



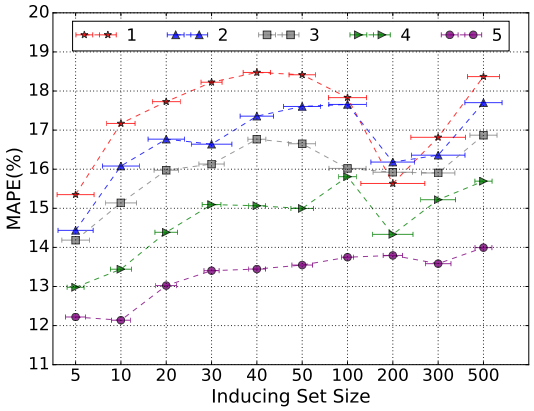
(a) SGPR-R (RMSE)



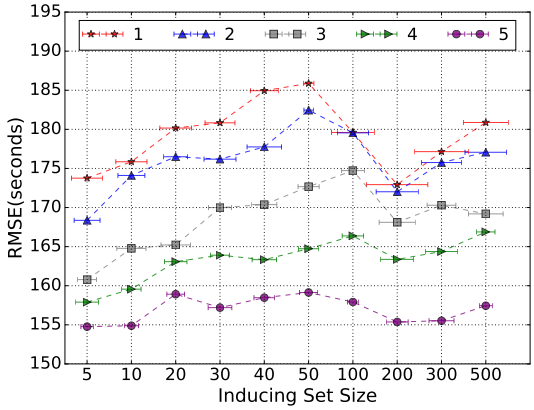
(b) SGPR-R (MAPE)



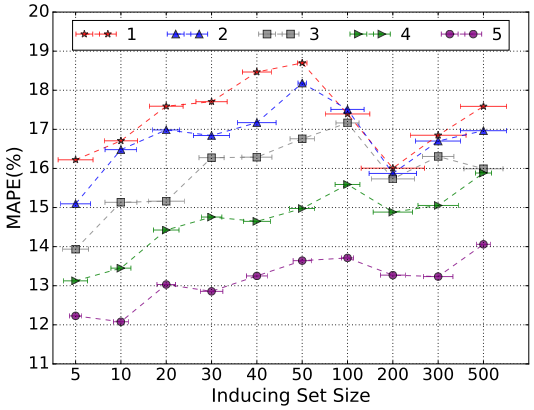
(c) SGPR-KM (RMSE)



(d) SGPR-KM (MAPE)



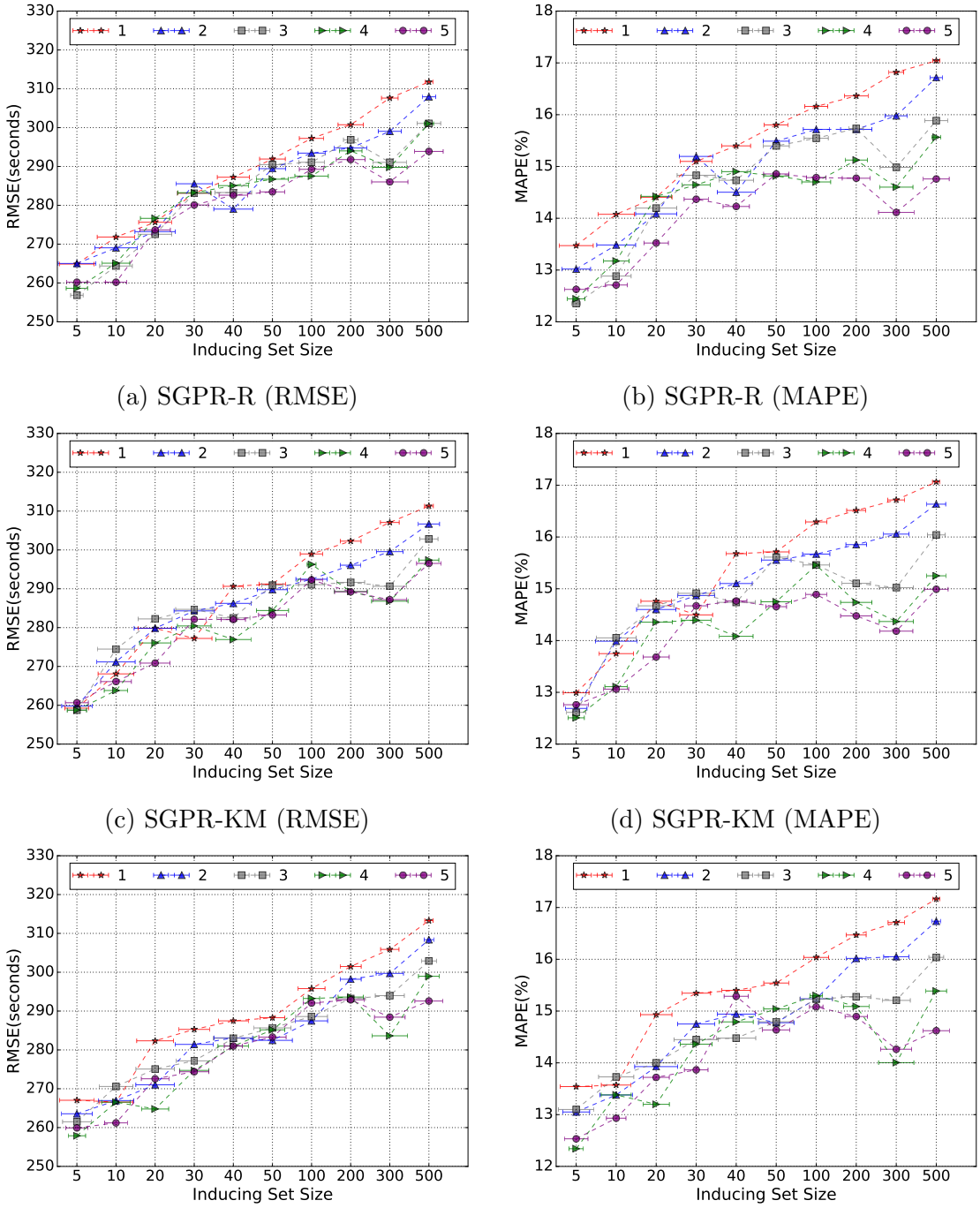
(e) SGPR-KP (RMSE)



(f) SGPR-KP (MAPE)

Figure 4.5: The OGP results of the SGPR models for R2[1].

[1] SGPR-R: SGPR with random selection; SGPR-KM: SGPR with k-means; SGPR-KP: SGPR with k-means++.



(a) SGPR-R (RMSE)

(b) SGPR-R (MAPE)

(c) SGPR-KM (RMSE)

(d) SGPR-KM (MAPE)

(e) SGPR-KP (RMSE)

(f) SGPR-KP (MAPE)

Figure 4.6: The OGP results of the SGPR models for R3[1].

[1] SGPR-R: SGPR with random selection; SGPR-KM: SGPR with k-means; SGPR-KP: SGPR with k-means++.

should be collected and incorporated into the model.

4.5.3 Comparative Study

The GPR models are also compared with the models implemented using multi-layer perceptrons (MLPs) and support vector regression (SVR). The MLPs used for this study is a single hidden layer MLPs. A first order gradient based optimisation algorithm, namely 'Adam' [103], is used as the optimisation algorithm for the converging training of the MLPs. All the MLPs and the SVR models are fitted using 5-fold cross validation. Both the BDP and OGP tasks are performed. The inducing set size is set to 5 and k-means is used as the inducing set initialisation method for the SGPR models. Table 4.5 and Table 4.6 show the BDP and OGP results, respectively. It can be seen that the SGPR models outperform the MLPs and SVR models for both the BDP and OGP tasks. The results also show that adding MSS can be beneficial to the prediction with the MLPs and the SVR models too; and the errors generally decrease with the increase of T in the case of R2 and R3. Since SGPR does not require time-consuming cross validated grid search to fit the models, the model fitting times are significantly shorter than those for MLPs and SVR.

Table 4.5: The BDP results of the full GPR models, the SGPR models, the SVR models, and MLPs.

Model	R1		R2		R3	
	RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)
SGPR-5-KM	170.57±6.21	15.72±0.85	181.35±7.94	17.95±1.65	276.11±17.93	14.77±2.24
GPR	181.55	17.12	193.29	20.03	320.47	17.71
SVR	180.88	17.06	240.24	22.14	333.54	18.11
MLP	181.68	17.10	246.44	22.53	340.98	17.60

4.6 Discussions and Conclusions

This chapter we presented the travel time prediction for container trucks using SGPR, with trucks' trajectory data. Three road segments were selected from the

Table 4.6: The OGP results of the full GPR models, the SGPR models, the SVR models, and MLPs.

T	Model	R1		R2		R3	
		RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)
1	SGPR-5-KM	167.18±5.77	15.29±0.92	169.68±7.62	15.35±1.63	259.27±12.28	12.99±1.28
	GPR	178.99	16.90	188.97	19.09	319.96	17.61
	SVR	180.88	17.06	238.88	21.93	335.85	18.29
	MLP	182.24	17.19	241.41	21.63	333.15	17.32
2	SGPR-5-KM	162.15±4.44	14.49±0.72	165.32±6.23	14.44±1.55	259.81±15.54	12.69±1.08
	GPR	179.81	16.97	184.55	18.48	314.72	17.17
	SVR	180.88	17.06	239.93	22.06	332.73	18.04
	MLP	181.84	17.15	241.98	22.25	335.12	17.62
3	SGPR-5-KM	163.98±4.92	14.70±0.77	161.40±4.92	14.19±1.22	258.76±9.78	12.61±0.96
	GPR	179.61	16.95	176.61	17.48	309.52	16.41
	SVR	180.88	17.06	237.11	21.78	333.12	18.05
	MLP	182.33	17.05	236.61	21.55	328.09	16.83
4	SGPR-5-KM	161.55±2.76	14.27±0.41	157.45±3.88	12.98±0.75	258.74±9.89	12.51±0.81
	GPR	178.56	16.86	169.90	16.44	307.35	16.14
	SVR	180.91	17.06	233.59	21.46	334.26	18.09
	MLP	181.51	17.14	234.90	21.84	319.59	16.44
5	SGPR-5-KM	161.31±1.59	14.21±0.25	155.49±3.42	12.22±0.88	260.66±11.73	12.76±1.25
	GPR	178.30	16.83	159.99	14.45	306.09	15.81
	SVR	181.19	17.10	229.09	20.97	332.58	17.93
	MLP	182.19	17.19	205.55	19.14	329.87	17.25

intermodal drayage road network of the Port of Ningbo-Zhoushan, China. All the trips were extracted by mining the trajectory data for each road segment. The travel times were estimated using a proportional distance-based method.

Two types of prediction task were performed: prediction before departure (BDP) and prediction for ongoing trips (OGP). Two date-time indicators (i.e., departure time, day-of-week) and the historical mean travel time were used as the input features for BDP. The results are not very satisfying. This is mainly because the input features used for BDP are not good enough to generalise the travel times for the test trips. Because each truck is equipped with a GPS module, which allows the truck to send a trajectory report every 30 seconds to a data server, thus, we can estimate the mean speed every time a new trajectory report is received. The resulted series of mean speed estimates is called the mean speed sequence (MSS). The MSS was then added to the feature columns to perform the OGP tasks. The results show that adding MSS to the feature columns is beneficial to the OGP prediction for R2

and R3. Analytically, MSS can be seen as the real-time data which reveals how fast a truck is on the travelled part of the target road segment. If the future traffic remains stationary, MSS can be used to infer how fast a truck may go on the rest part of the road segment. However, if a non-recurring event occurs or the queue at an intersection is relatively long on the untravelled part of the road segment, MSS can be less useful.

The SGPR models were also compared with the MLPs and the SVR models. The results show that the SGPR models outperform the MLPs and the SVR models. Because our data is temporally sparse and noisy, if an algorithm is not able to handle the data redundancy and the outliers well, the model that fully relies on the entire training data may yield a worse result than the one that partially relies on the training data with careful selection. Unlike MLPs and SVR, SGPR builds a model based on the joint distribution of both the training data and test data, in which the test data is conditionally independent from the training data via the inducing set. In this case, the negative impact of 'bad' data points can be reduced by using a small inducing set. The difference in performance between the SGPR models and other types of models might be shortened if more useful input features (e.g., traffic flow data, weather, load) are incorporated. Unfortunately, the relevant data is not available for this study. Because SGPR does not require time-consuming cross validated grid search to tune the model hyperparameters, the model fitting time is significantly shorter for SGPR than that for MLPs and SVR. Since the predictive posterior can be recursively updated by feeding new data, SGPR is also feasible for online applications.

In general, the inducing set initialisation method does not have significant impact on the model performance. However, the selection of inducing set size can have a significant impact on the performance, particularly if the quality of the data is not fairly distributed over time. More case studies will be carried out in the future to investigate the robustness of the performance for SGPR.

Chapter 5

Freight Vehicle Travel Time Prediction by Decision Tree Ensembles

5.1 Introduction

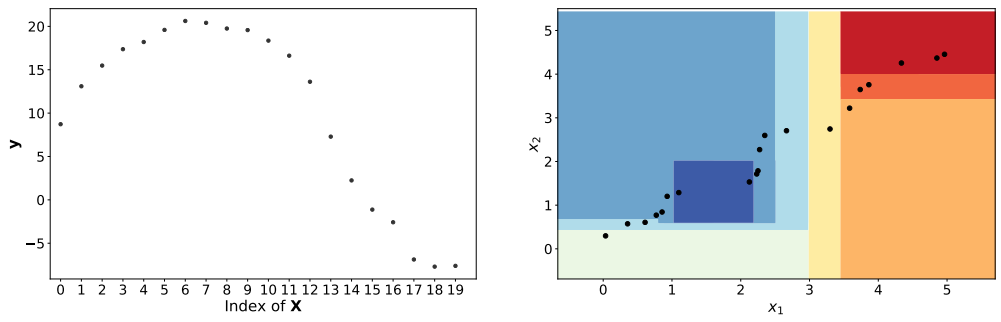
In Chapter 4, sparse Gaussian process regression (SGPR) was used to perform the travel time prediction for container trucks with the trucks' trajectory data. The results show that the SGPR approach outperforms the MLPs and SVR approaches. Although the SGPR approach requires no lengthy model fitting process, and yields satisfying prediction results, the model variance is still relatively large due to the inducing set selection. Furthermore, it still requires careful feature selection and feature (value) transformation. Decision tree (or tree) ensembles (e.g., random forests, gradient boosting regression trees) are becoming more and more popular in the machine learning community due to their appealing performances in some world-class data mining competitions. Several studies [110, 136, 73] have shown that they can work directly with mixed-type multi-dimensional input features; they are insensitive to the monotonic feature transformation, and they are more robust to outliers and

missing values. In this chapter, we continue to examine the travel time prediction performance for the container trucks, by the three tree ensembles, adaptive boosting regression trees, random forests, and gradient boosting regression trees. The same datasets are used as those for Chapter 4.

The rest of this chapter is organised as follows: Section 5.2 provides a brief introduction of the three tree ensembles; Section 5.3 explains the experiments including data, prediction tasks, input features, and performance evaluation metric; Section 5.4 provides the results and analysis; the discussions and conclusions are presented in Section 5.5.

5.2 Methodology

A single decision tree model partitions training data into a set of disjoint regions where each region is fitted using a simple model. Figure 5.1a shows an example of a small pseudo training data. The data consists of 20 input vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{20}$ of two feature variables (x1, x2) and the corresponding output vector $\mathbf{y} = \{y_i\}_{i=1}^{20}$, $\mathbf{X} \in \Re^2$, $\mathbf{y} \in \Re$. We grew a tree to this given dataset where the tree is implemented using the classification and regression trees algorithm (CART) [18]. Figure 5.1b shows the in-sample decision surface for the partitions over the data. The dark blue region in the figure contains two data points which correspond to the two data points (i.e.,



(a) Data (b) Decision surface

Figure 5.1: The example of decision tree learning for a regression problem.

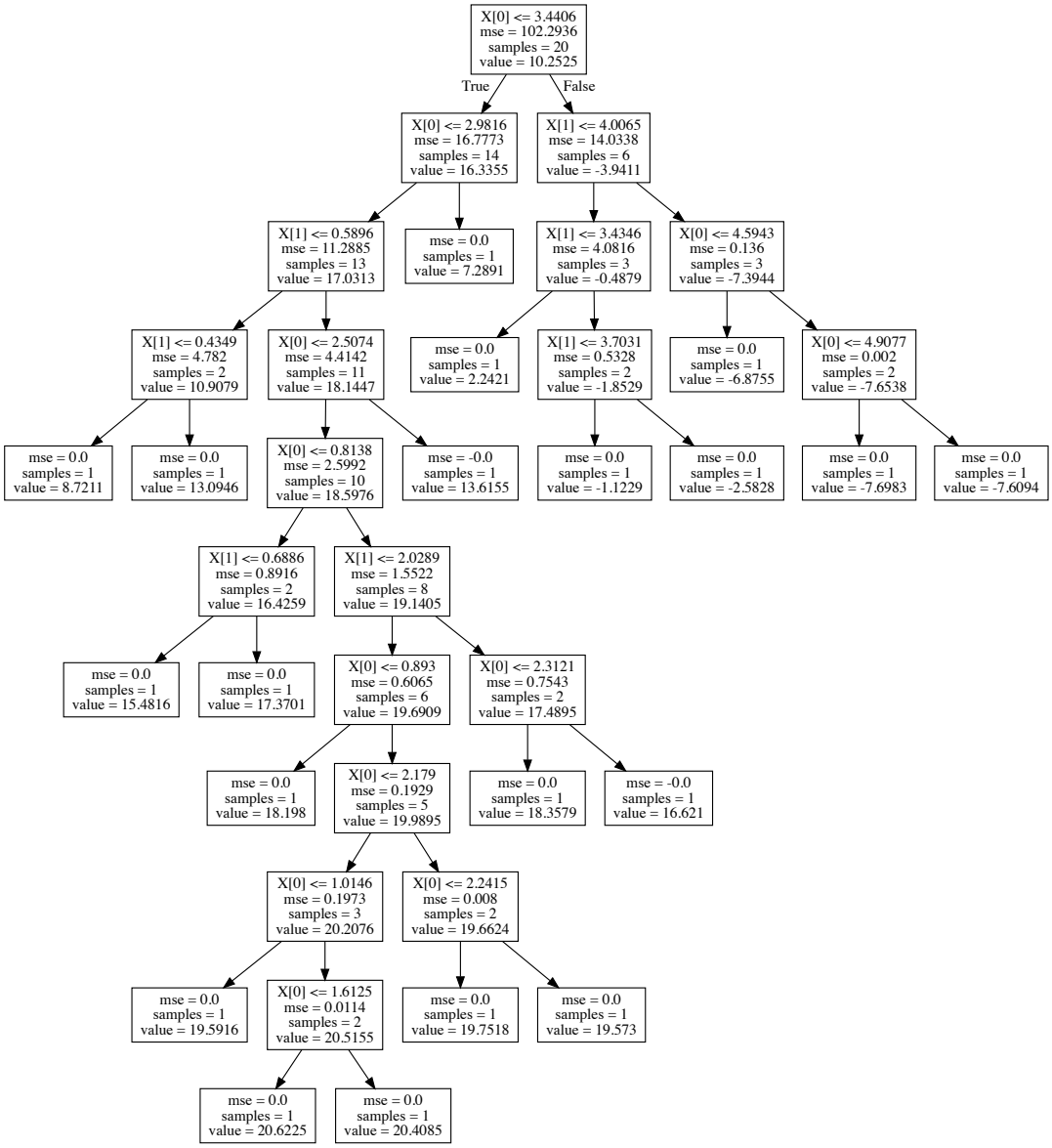


Figure 5.2: The interpretable view of the grown decision tree to the data in Figure 5.1a.

indices 6 and 7) at the peak in Figure 5.1a; the dark red region contains three data points which correspond to the last three data points in Figure 5.1a. Figure 5.2 shows the grown tree. It can be seen that the tree recursively makes binary splits until a stop is met, i.e., reaching the maximum depth of the tree (set to 50 in this

example) or reaching the minimum leaf node sample size (set to 1 in this example).

A single tree is sensitive to the data: a small change in the training data may result in a different tree structure which can lead to a different prediction result. This structural instability is the hierarchical nature of the growing procedure, which can hardly be avoided even with a smarter split strategy [73]. Therefore, single tree models are not robust to the changes of data and potentially have high variance. To address the drawback of single tree models, tree ensembles have been introduced to reduce the model variance. A tree ensemble assembles multiple trees where each tree is grown to a given data which can be drawn from the training data with or without replacement. Tree ensembles can be differentiated by the way they assemble trees including bagging, boosting or hybrid approaches. Three popular tree ensembles are selected for this study: random forests (RF), adaptive boosting regression trees (ABRT) and gradient boosting regression trees (GBRT). We now give a more detailed account of how these tree ensembles work.

5.2.1 Random Forests

Assume we have a dataset $\mathcal{D}$ consisting of N input vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ of dimension d ($d \geq 1$) and the corresponding output vector $\mathbf{y} = \{y_i\}_{i=1}^N$; $\mathbf{X} \in \Re^d$; $\mathbf{y} \in \Re$. RF assembles multiple trees where each tree is grown to a bootstrapped sample $\mathcal{Z}^*$ of size N that is randomly drawn from $\mathcal{D}$ with replacement. This strategy is called *bagging*. The idea of the RF is summarised in Algorithm 1.

Because each tree is identically distributed, the expectation of B trees' average is the same as the expectation of any of them. Thus, to make the prediction on a test case $\mathbf{x}_*$, we can simply average the prediction results of B trees in the ensemble:

$$\hat{f}(\mathbf{x}_*) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x}_*) \tag{5.1}$$

The performance of a RF model can be gradually improved by growing more

Algorithm 1 Random forest for regression [73, pp.588]

1: Initialise the number of trees, B.
2: For $b = 1$ to B

a) Draw a bootstrapped sample $\mathcal{Z}^*$ of size N from $\mathcal{D}$ with replacement.

b) Grow a tree T_b to $\mathcal{Z}^*$, by recursively repeating the following steps for each terminal node of the tree, until a stop is met:

I. Select m variables (i.e., input features) at random from d features.

II. Pick the best feature(s)/split-point(s) among the above m selected features.

III. Split the sample into two child nodes.

3: Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.

trees [73]. However, in some cases, growing too many trees only increases the computational cost but with very little or even no performance gain. Thus, fitting an RF model usually requires a trade-off between the number of trees and the computational budget (e.g., time, memory). Hastie et al. [73] reported that the performance of the RF model can be improved by applying maximum depth control for regression problems. Thus, two parameters are proposed to control the tree growth: the maximum tree depth (d_{max}) and the minimum leaf node samples (s_{min}). A tree should stop growing whenever its depth reaches the d_{max} or the sample size in leaf node reaches the s_{min}. Leo Breiman [19] suggested $m = d/3$ and $s_{min} = 5$ for regression problem in general. It is suggested that the best values of B, m, d_{max} and s_{min} should be obtained using cross validation. Using bootstrapped samples leaves us with a bunch of data that are not used to grow trees. These left-out data are known as the out-of-bag (OOB) data. In this case, the validation of a tree can be performed using the OOB data within the model fitting process. Thus, no separated validation process is needed. Since trees in an RF ensemble are fitted independently, the model fitting process can be performed in parallel.

5.2.2 Adaptive Boosting Regression Tree

Unlike the bagging strategy that each training example is equally likely to be picked, the *boosting* strategy focuses on the training examples that are not learnt well. As a result, the probability of a particular training example being in a training data for tree growth is no longer identical to others but depends on the performance of previously built ensemble on that example. Adaptive boosting regression trees (ABRT) is a popular type of boosting tree ensemble proposed by Drucker [40]. The basic idea of ABRT is summarised in Algorithm 2.

To make the prediction on a new test input $\mathbf{x}_*$, we can calculate the weighted median of the outputs of all the trees in the ensemble. In Algorithm 2, β_m is the measure of confidence in T_m. Low β_m means high confidence in T_m. Updating the weight depends on the performance loss. Smaller loss yields more weight reduction thus makes a training example less likely to be picked for the next boosting iteration. M is the maximum boosting iterations which can be predefined by users. $\mathcal{L}$ can be

Algorithm 2 Adaptive boosting regression trees for regression.

1: Initialise the maximum boosting iterations, M.
2: Assign a weight for each training example $w_i = 1$ for $i = 1, 2, ..., N$.
3: Initialise the probability for each training example $p_i = w_i / \sum w_i$, $i = 1, 2, ..., N$.
4: For $m = 1$ to M:
   a) Pick training examples based on their probabilities to make the training data $\mathcal{Z}^*$.
   b) Grow a tree T_m to $\mathcal{Z}^*$.
   c) Compute the weighted average error for the mth iteration: $\bar{e}_m = \sum_{i=1}^{N} p_i \cdot \mathcal{L}(\hat{y}_i, y)$.
   d) If $\bar{e}_m < 0.5$, boosting stops, go to step 5.
   e) Compute $\beta_m = \dfrac{\bar{e}_m}{1 - \bar{e}_m}$.
   f) Update the weight $w_i \longleftarrow w_i \cdot \exp[1 - \mathcal{L}(\hat{y}_i, y)]$, $i = 1, 2, ..., N$.
5: Output the ensemble of trees $\{T_m\}_{m=1}^{M}$.

any differentiable loss function (e.g., absolute loss, squared loss, exponential loss).

Increasing M can help reduce the training error, but too large M can increase the risk of overfitting. In Algorithm 2, a threshold value (0.5) is defined so that if the reduction of the weighted average error is below 0.5, the boosting should be stopped [40]. We can also use the maximum tree depth (d_{max}) and the minimum leaf node samples (s_{min}) to control the tree growth. In general, trees in ABRT ensemble are shallower than those in RF ensembles.

5.2.3 Gradient Boosting Regression Trees

Gradient boosting regression trees (GBRT) is another popular type of boosting tree ensemble. The idea of GBRT is summarised in Algorithm 3. The second line initialises the optimal constant model, which is a single terminal node tree. $\mathcal{L}$ can be any differentiable loss function, e.g., squared error loss, absolute error loss, Huber M-regression loss [87]. γ parameterises the split feature variables and the split points for the trees in the ensemble. Line 3(a) computes the negative gradient of the $\mathcal{L}$ for the given training data. At line 3(b), a tree partitions the input space into J_m

Algorithm 3 Gradient boosting regression trees for regression [73, pp.361].

1: Initialise the maximum boosting iterations, M.
2: Initialise $f_0(\mathbf{x}) = \arg\min_\gamma \sum_{i=1}^{N} \mathcal{L}(y_i, \gamma)$.
3: For $m = 1$ to M:

a) For i = 1, 2, ..., N compute

$$r_{im} = -[\frac{\partial \mathcal{L}(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}]_{f=f_{m-1}}$$

b) Grow a tree to the targets r_{im} giving terminal regions R_{jm}, j = 1, 2, ..., J_m.

c) For j = 1, 2, ..., J_m compute

$$\gamma_{jm} = \arg\min_\gamma \sum_{\mathbf{x}_i \in R_{jm}} \mathcal{L}(y_i, f_{m-1}(\mathbf{x}_i) + \gamma)$$

d) Update $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \sum_{j=1}^{J_m} \gamma_{jm} I(\mathbf{x} \in R_{jm})$

4: Output $\hat{f} = f_M(\mathbf{x})$.

disjoint regions, then it is fitted to the target r_{im}. J_m is usually treated as a meta-parameter of the entire boosting procedure, which can be adjusted to maximise the estimated performance for the given data [73]. Line 3(c) finds the best constant j_m for each region. Line 3(d) updates the ensemble by adding a newly fitted tree $\sum_{j=1}^{J_m} \gamma_{jm} I(\mathbf{x} \in R_{jm})$. In this study, the absolute error loss is selected as the loss function because it is more robust to the changes of the data.

The maximum boosting stages M and the number of terminal nodes J_m are the two basic tuning parameters for GBRT. J_m controls the size of trees in the ensemble. Hastie et al. [73] suggested that the value of J_m should be between 4 and 10. The control of the tree size in the ensemble also can be achieved by setting the maximum tree depth or the minimal leaf node samples as those for RF. Friedman [52] introduced the *learning rate* ν $(0 < \nu < 1)$ to scale the contribution for each tree. The step 3(d) in Algorithm 3 becomes:

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu \cdot \sum_{j=1}^{J_m} \gamma_{jm} I(\mathbf{x} \in R_{jm}) \tag{5.2}$$

Smaller ν yields larger training risk given M. Thus, both M and ν are able to control the prediction risk on the training data. However, decreasing ν may result in the increase of M, i.e., there exists a trade-off between ν and M. Friedman [53] also suggested to use a fractional training data (without replacement) to grow the next tree. The fraction of the training data is controlled by the parameter η. Their empirical study shows that using fractional training data not only reduces the computational cost but also yields better prediction result [53, 73].

5.2.4 Hyperparameter Tuning for Tree Ensemble Models Using Bayesian Optimisation

A hyperparameter is usually referred to as a configuration of a learning method that is external to the model and whose value can be learned via data. A typical

way of tuning hyperparameters for tree ensembles usually requires a nested cross validation. That is, the inner loop performs a k-fold cross validation while the outer loop retrieves a hyperparameter query from a predefined domain grid. This nested cross validation is also known as cross validated grid search (CVGS). CVGS is very ineffecient because it checks every query in the grid, being unable to capture the hierarchical dependencies between parameters. Although such hierarchical dependencies can be built into the grid by experts, this makes the tuning process non-automated.

Bayesian optimisation (BO) has received more attention in recent years due to its capability of joint optimisation of design choices including model hyperparameter tuning [172]. In BO, we are interested in finding the maximum (or minimum) of a non-specifiable function $f(\mathbf{x})$ on some bounded $\mathbf{X} \in \Re^d$, d is the dimensionality of $\mathbf{X}$. BO specifies a probabilistic model for $f(\mathbf{x})$ and then exploits the model to determine where in $\mathbf{X}$ to next evaluate the function, while integrating out uncertainty. The key concept is to utilise all the information from previous evaluations and not just rely on local gradient and Hessian approximation. As a result, BO is able to find the maximum (or minimum) of difficult non-convex functions with relatively few evaluations [180]. The BO procedure is shown in Algorithm 4 [172]. For model hyperparameter tuning problems, BO first specifies a prior distribution (e.g., a Gaussian process prior) over the hyperparameter query space. The posterior

Algorithm 4 Bayesian Optimisation

1: Initialise the maximum optimisation iterations, M.
2: For $m = 1$ to M:
   a) Find $\mathbf{x}_{m+1}$ by optimising the acquisition function α
   $$\mathbf{x}_{m+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_m)$$
   b) Query objective function to get y_{m+1}
   c) Augment data $\mathcal{D}_{m+1} = \{\mathcal{D}_m, (\mathbf{x}_{m+1}, y_{m+1})\}$
   d) Update statistical model
3: Output $\mathbf{x}$ that has the best y (i.e., maximum or minimum value).

distribution then can be computed by applying Bayes' theorem, conditioning on the corresponding observed validation scores. The next query (to be evaluated) can be determined by maximising an acquisition function which utilises the information of the posterior distribution (e.g., the mean and variance).

Reguler BO approaches explore the parameter space sequentially. In high dimensional and or otherwise complex problems, the number of required evaluations can be notably large. In this case, it may take a lot of time to complete the optimisation procedure. Therefore, many batch BO approaches [12, 61, 180] were proposed to make the evaluations of f simultaneously on batches of data. The batch BO approaches require the modelling of the interaction between the different evaluations in a batch, which can be computationally expensive. Gonzalez et al. proposed a batch BO approach, namely BBO-LP, that selects batches of data points by an iterative *maximisation-penalisation* loop around the acquisition function [63]. In their BBO-LP, the latent function f that maps inputs and outputs is assumed to be a Lipschitz continuous function, which is a common assumption in global optimisation. This assumption allows the authors to place bounds on how far the optimum of f is from a certain location. There are two steps repeated for all the data points in a batch: the maximisation step and the penalisation step. In the maximisation step, the acquisition function is maximised using the same way as the non-batch case. In the penalisation step, the output of the acquisition function is reduced using a local penalisation in the neighbourhood around the location of the most recent maximum of the acquisition function. For a more detailed discussion of BBO-LP, please refer to [63].

Table 5.1 lists the hyperparameters to be tuned for the RF and the GBRT models, respectively. Gaussian process (GP) is selected to specify the prior distribution, with the automatic relevance determination squared exponential kernel [161]. Although Gonzalez et al. [63] reported that using the upper confidence bound as the acquisition function yielded slightly better result than that using the expected improvement, the upper confidence bound requires additional tuning parameter to

Table 5.1: The tuning hyperparameters for RF, ABRT, and GBRT.

Method	Parameter	Description	Type	Domain
RF	B	Number of trees	discrete, integer	$[1, 500]$
	d_{max}	Maximum depth	discrete, integer	$[1, 500]$
	k[1]	Maximum features	discrete, integer	$[1, d]$[2]
ABRT	d_{max}	Maximum depth	discrete, integer	$[1, 500]$
	k	Maximum features	discrete, integer	$[1, d]$
	ν	Learning rate	discrete, real	0.01, 0.02, ..., 0.9
	M	Boosting stages	discrete, integer	500
GBRT	d_{max}	Maximum depth	discrete, integer	$[1, 500]$
	ν	Learning rate	discrete, real	0.01, 0.02, ..., 0.09
	η	Subsampling fraction	discrete, real	0.1, 0.2, ..., 0.9
	k	Maximum features	discrete, integer	$[1, d]$
	M	Boosting stages	discrete, integer	500

[1] k is the maximum number of input features used for tree splits. Applying k aims to reduce the training risk of overfitting.
[2] d is the dimensionality of the input features.

Table 5.2: The BBO-LP configuration.

Prior distribution	Gaussian process
Acquisition function	Expected improvement
Optimiser	Limited-memory Broyden-Fletcher-Goldfarb-Shanno
Initial query size	10
Initial selection	Random selection
Batch size	10
Iterations	10

balance the exploitation against the exploration, which increases the complexity of the optimisation process. Hence, the expected improvement is used as the acquisition function in this study. The initial query selection is set to 10 as is suggested in [63]. In general, the more batches are collected, the better the search result is. However, collecting too many batches is counterproductive regarding the performance. Our trial study shows that ten batches with the size of 10 queries are good enough for the tuning task. Thus, we set both the batch size and the number of iterations to 10, which results in 100 queries. The BBO-LP configuration is summarised in Table 5.2.

All the tree ensemble models are implemented using the popular Python open-

source machine learning toolkit scikit-learn [154]. All the tuning experiments are implemented using GPyOpt [189], a Python open source library that was developed and maintained by the machine learning group of the University of Sheffield, UK.

5.3 Experimentation

5.3.1 Data and Prediction Tasks

The same data is used for this study as for Chapter 4 (see Table 5.3). The two types of prediction tasks, prediction before departure (BDP) and prediction for ongoing trips (OGP), are also selected for this study. For each road segment, the last 200 trips are used for performance evaluation, and the remaining trips are used for training.

Table 5.3: Datasets.

Dataset	Road Segment	Direction	Length(m)	Trips
R1	a road segment of 2nd Tong Dao	west→east	7187	1143
R2	a road segment of 2nd Tong Dao	east→west	5847	1643
R3	a road segment of G329	east→west	7993	1320

5.3.2 Features

In Chapter 4, three types of input features were used: date-time indicator, historical descriptive statistic, and real-time data. Departure time and day-of-week are the only date-time indicators for both the BDP and OGP tasks because of the relatively short span of time in the data. The historical mean travel time was used as the historical descriptive statistic. The BDP results show that using the departure time, day-of-week and historical mean travel time are not good enough to generalise travel times for the test trips. Because each truck is equipped with a GPS module, which allows the truck to send a trajectory report every 30 seconds to a data server. Thus, we can estimate the mean speed every time a new trajectory report is received. The

resulted series of mean speed estimates is called the mean speed sequence (MSS). MSS can be seen as the real-time data which reveals how fast a truck is on the travelled part of the target road segment. If the future traffic remains stationary, MSS can be used to infer how fast a truck may go on the remaining part of the road segment. The results show that adding MSS to the feature columns is beneficial to the OGP prediction for R2 and R3.

The main drawback of MSS is that it predicts the future mean speed of a truck based on the mean speed behaviours on the travelled part of the target road segment. If a non-recurring event occurs or the queue at an intersection is relatively long on the untravelled part of the road segment, MSS becomes less useful. To address the limitation of MSS, the historical MSS (HMSS) is introduced in this study. Instead of focusing on the first T minutes after a trip starts, HMSS focuses on the last T' minutes before a trip ends. To compose a HMSS of the last T' minutes, we first compute the MSS in the last T' minutes for each trip in the training data:

$$MSS_{T'} = \{\hat{v}_i\}_{i=1}^{M} \tag{5.3}$$

where $\hat{v}_i$ is the last ith mean speed estimates:

$$\hat{v}_i = \frac{\hat{L}_i}{t_{end} - t_i} \tag{5.4}$$

where t_i is the corresponding instant time of the last ith trajectory report; t_{end} is the end time of the trip; $\hat{L}_i$ is the estimation of travelled distance in the last $(t_{end} - t_i)$ minutes; M is equivalent to the number of trajectory reports received within the last T' minute. Then, the HMSS can be obtained by simply averaging out all the MSSs of the last T' minutes. Because our trip data is temporally sparse, similar rules are applied (see Section 4.4.2) to make sure that every trip has a corresponding HMSS.

5.3.3 Performance Evaluation Metrics

Root mean squared error (RMSE) and mean absolute percentage error (MAPE) are selected as the prediction accuracy evaluation metrics. Each prediction task is replicated for 30 times, the means and the standard errors are reported.

5.4 Results and Analysis

5.4.1 Prediction Before Departure

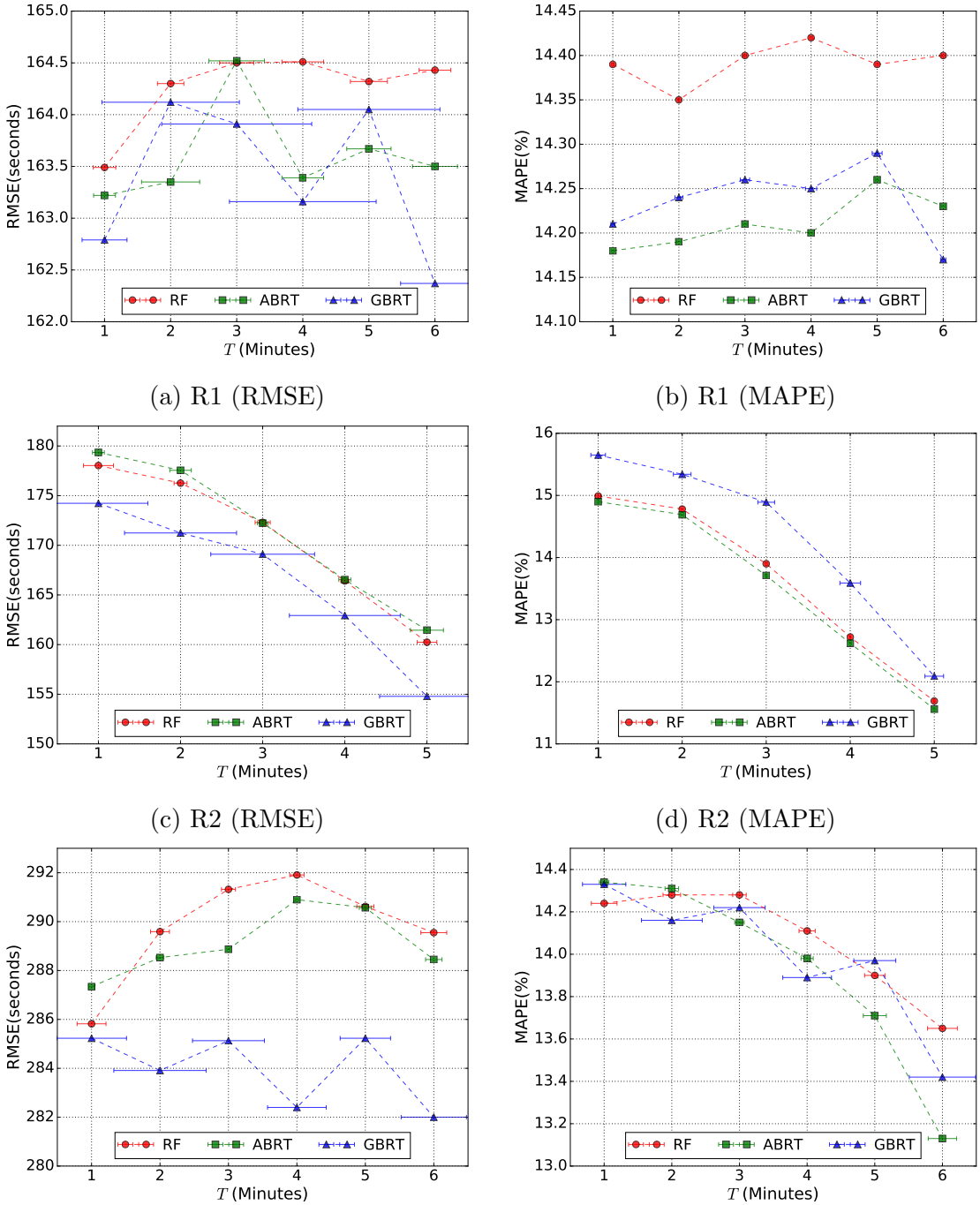
Table 5.4 shows the results of the predictions before departure (BDP) by the three types of tree ensemble models. It can be seen that the performances of the three types of tree ensemble models are very similar for R1. For R2, the RMSE results show that the GBRT model is slightly better than the others, while the MAPE results show that the ABRT model is slightly better than the others. For R3, the RF model is overall the best but with relatively larger variance. With insufficient features, the difference in the performance between bagging and boosting strategies is not practically significant. Further analysis on the tree size of those tree ensemble models shows that the trees are generally shallower in the boosting tree ensemble models than those in the RF models.

Table 5.4: The BDP results of the tree ensemble models.

Method	R1		R2		R3	
	RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)
RF	163.80±0.25	14.34±0.02	181.41±0.86	16.01±0.17	279.30±4.80	13.93±0.26
ABRT	163.69±1.17	14.37±0.17	182.54±0.89	15.98±0.21	283.38±1.45	14.12±0.19
GBRT	163.83±1.05	14.35±0.16	178.19±4.22	16.14±0.28	285.06±3.96	14.45±0.21

5.4.2 Prediction For Ongoing Trips

Figure 5.3 shows the OGP results of the tree ensemble models. The x-axis T indicates how soon (in minutes) the prediction is performed after a trip starts. The



(a) R1 (RMSE)

(b) R1 (MAPE)

(c) R2 (RMSE)

(d) R2 (MAPE)

(e) R3 (RMSE)

(f) R3 (MAPE)

Figure 5.3: The OGP results of the tree ensemble models.

number of mean speed estimates in MSS increases as T increases. The horizontal lines that lie on the symbols indicate the variations of the errors. The variations are

scaled for display purpose. For R1, the error fluctuates with the increase of T. Both the RMSE and MAPE results show that adding MSS is not beneficial to the prediction. The boosting tree ensemble (i.e., ABRT and GBRT) models yield slightly better results than the RF models. For R2, the error decreases with the increase of T. The RMSE results shows that the GBRT model is overall the best while the MAPE results show that GBRT is overall the worst. For R3, the RMSE results of the RF and ABRT models show that the error first increases when T increases from 1 to 4, then it decreases when T increases from 4 to 6. This 'rise-and-fall' variability pattern is not observable in the result of GBRT models. The MAPE results show that the error generally decreases with the increase of T for all the three types of tree ensemble models. Note that RMSE is sensitive to outliers. The differences between the RMSE and the MAPE results may indicate that there are large errors which can be '*over punished*' by squaring them during the RMSE calculation. The results also show that the GBRT models have relatively larger variance mainly because fitting a GBRT model requires more hyperparameters to tune and the 'subsampling' strategy is adopted to reduce the risk of overfitting.

In general, adding MSS into the feature columns is beneficial to the prediction for R2 and R3, but not for R1. Analytically, if a target road segment is relatively short (i.e., less than 10km) and the future traffic condition is stationary, MSS is able to infer how fast a truck may move on the untravelled part of the road segment, based the mean speed behaviours on the travelled part of the road segment. However, MSS cannot capture the interruption of the traffic caused by some unexpected events (e.g., incident, road work). Therefore, the historical MSS (HMSS) is incorporated into the model so that the possible speed behaviours on the untravelled part of the road segment can be learned based on the speed behaviours of some recent trips.

Table 5.5 shows the comparison results between with and without using the HMSS of the last 5 minutes. The OGP tasks are performed 3, 4 and 5 minutes after a trip starts for each road segment. It can be seen that the error is slightly reduced by adding HMSS for all the three types of tree ensemble models for R1 and R2. The

Table 5.5: The OGP results of the tree ensemble models with HMSS of the last 5 minutes.

Method	T	HMSS[1]	R1		R2		R3	
			RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)
RF	3	N	164.50±0.52	14.40±0.05	172.30±0.92	13.90±0.15	291.32±1.02	14.28±0.10
		Y	164.48±0.89	14.37±0.05	170.71±0.54	13.63±0.10	290.95±1.05	14.27±0.10
	4	N	164.51±0.63	14.42±0.05	166.41±0.49	12.72±0.06	291.91±1.04	14.11±0.12
		Y	164.48±0.89	14.36±0.05	164.09±0.91	12.42±0.12	291.98±1.22	14.09±0.11
	5	N	164.32±0.56	14.39±0.05	160.24±1.20	11.69±0.11	290.62±1.24	13.90±0.15
		Y	164.68±0.81	14.38±0.05	159.70±0.95	11.44±0.10	290.65±1.55	13.92±0.16
ABRT	3	N	164.52±0.84	14.21±0.06	172.23±0.64	13.71±0.09	288.87±0.23	14.15±0.07
		Y	163.11±0.71	14.20±0.05	169.98±1.51	13.31±0.31	288.93±0.21	14.14±0.07
	4	N	163.39±0.63	14.20±0.06	166.54±0.73	12.62±0.14	290.90±0.29	13.98±0.09
		Y	162.78±0.45	14.19±0.06	163.53±0.60	11.74±0.16	290.91±0.31	13.98±0.09
	5	N	163.80±0.25	14.26±0.05	161.45±2.01	11.56±0.19	290.57±0.89	13.71±0.17
		Y	162.69±0.53	14.19±0.05	158.02±0.63	10.24±0.14	290.50±1.04	13.71±0.18
GBRT	3	N	163.91±2.27	14.26±0.14	169.10±6.32	14.89±1.00	285.13±5.25	14.22±0.38
		Y	162.61±0.75	14.15±0.07	163.25±6.46	13.96±1.00	282.43±5.16	14.02±0.34
	4	N	163.16±2.22	14.25±0.17	162.93±6.75	13.59±1.23	282.40±4.29	13.89±0.36
		Y	162.92±1.15	14.16±0.10	152.81±4.74	11.65±1.01	284.94±2.95	14.11±0.33
	5	N	164.05±2.15	14.29±0.15	154.79±5.77	12.09±1.11	285.23±3.68	13.97±0.31
		Y	162.81±1.01	14.14±0.07	149.34±4.52	10.88±0.91	285.15±3.73	13.91±0.36

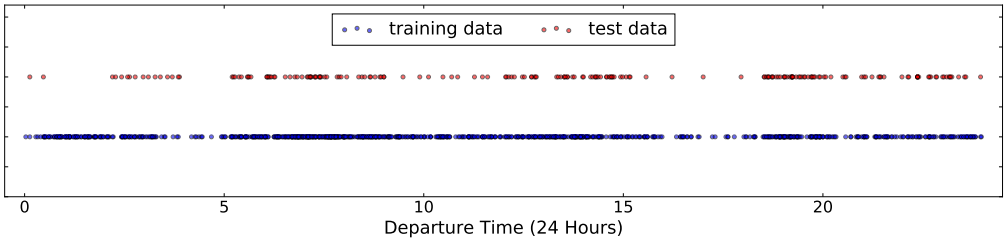
[1] N - HMSS is not used; Y - HMSS is used.

GBRT models receive more improvement than the others. For R3, adding HMSS does not yield any performance boost.

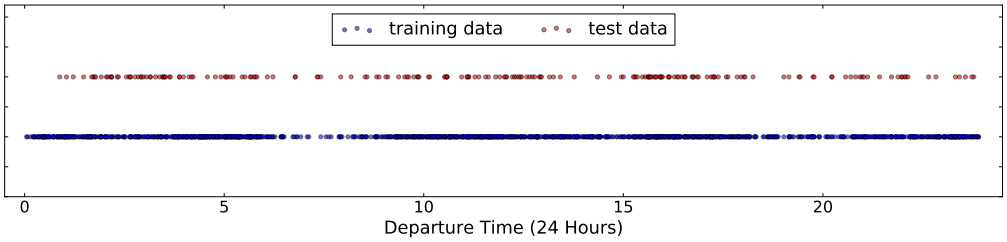
To understand why different performance gains are obtained, we looked at how well the training data covers the relevant prediction periods. Figure 5.4 shows that there are several trips in the test data whose departure times are not covered in the training data. The departure time coverage ratio (%) is then calculated for each dataset. The coverage ratio of a given dataset can be calculated by:

$$p_{coverage} = \frac{N_{covered}}{N_{test}} \times 100 \tag{5.5}$$

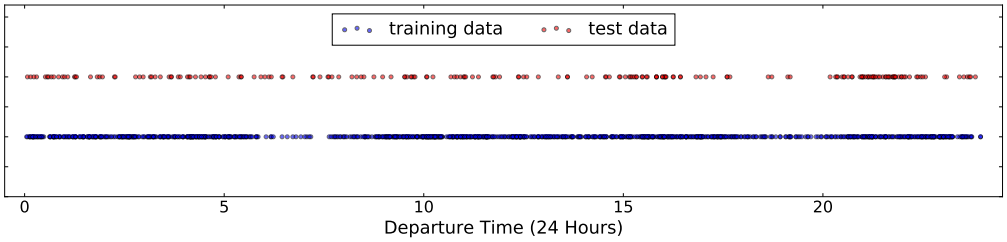
where $N_{covered}$ denotes the number of departure times covered in the training data (without duplication); N_{test} denotes the number of trips in the test data. As is shown in Table 5.6, R2 has the best coverage while R3 has the worst coverage. Further experiment results show that the error can be slightly reduced further by removing the trips whose departure times are not covered by the training data for all the three road segments. This result suggests that there is a relation between the effectiveness



(a) R1



(b) R2



(c) R3

Figure 5.4: The departure times covered in the training data and test data for each road segment.

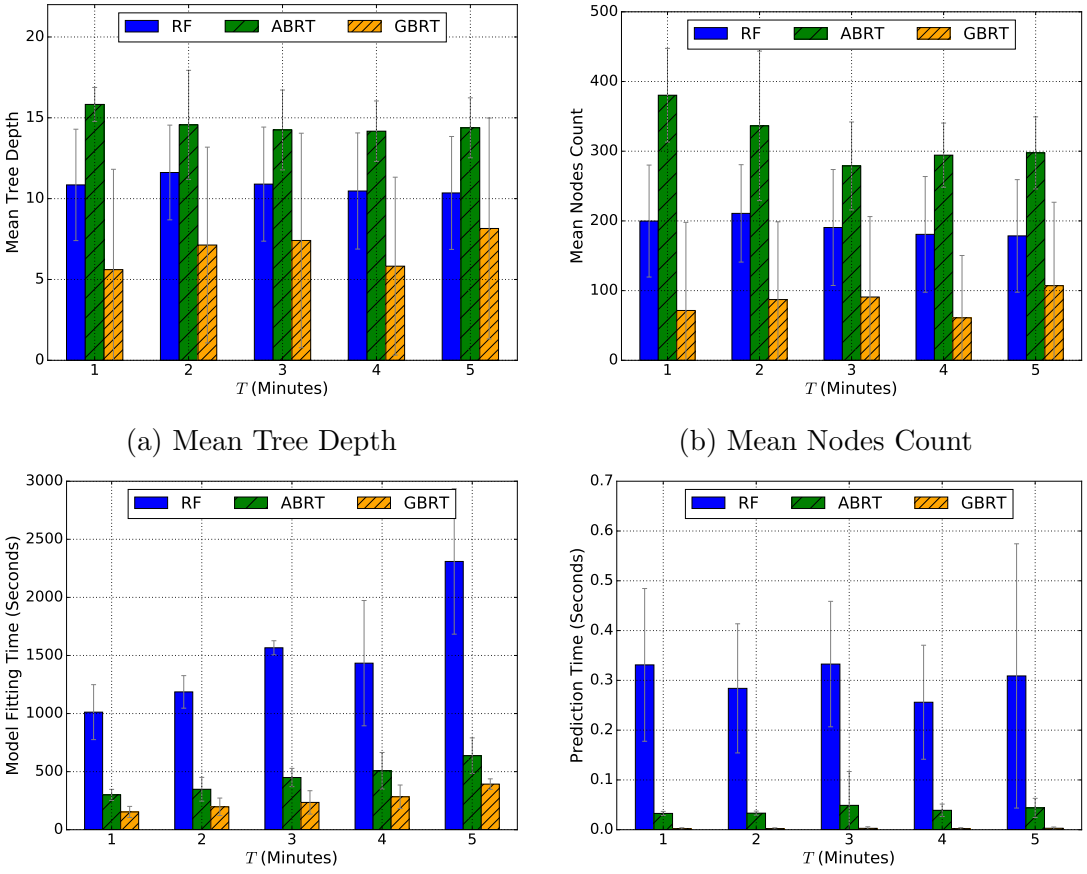
Table 5.6: The departure time coverage ratios for each road segment.

Dataset	Coverage Ratio (%)
R1	53%
R2	67%
R3	47%

of HMSS and the departure time coverage ratio that larger departure time coverage is more likely to yield a better result, with the help of HMSS.

5.4.3 The Computational Cost of The Tree Ensemble Models

Besides the prediction accuracy, the computational cost is also an essential aspect of the performance evaluation. The computational cost is evaluated from three aspects: tree size, model fitting time, and prediction time. The tree size is evaluated by measuring the mean tree depth and the mean nodes count in a tree ensemble model. Figure 5.5 shows the computational cost of the tree ensemble models for the OGP tasks for R1, without using HMSS. It can be seen that the trees in the GBRT models are much shallower than those in the RF and the ABRT models. Because



(a) Mean Tree Depth

(b) Mean Nodes Count

(c) Model Fitting Time

(d) Prediction Time

Figure 5.5: The computational cost analysis of the tree ensemble models in the OGP tasks for R1, without using HMSS.

the trees are generally much shallower in the GBRT models than in the RF and the ABRT models, the model fitting time and prediction time are also significantly shorter than those for the others. These findings are consistent with those for the other two road segments.

5.4.4 Comparative Study

Table 5.7: The comparison of the OGP results among the tree ensemble models, the SVR models, and the MLPs.

T	Model	R1		R2		R3	
		RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)	RMSE(sec)	MAPE(%)
1	RF	163.49±0.34	14.39±0.04	178.03±1.83	14.99±0.33	285.82±2.11	14.24±0.19
	ABRT	163.22±0.33	14.18±0.05	179.35±0.72	14.90±0.15	287.34±0.41	14.34±0.07
	GBRT	162.79±0.68	14.21±0.06	174.23±6.03	15.65±0.87	285.23±5.11	14.33±0.32
	SVR	180.88	17.06	238.88	21.93	335.85	18.29
	MLP	182.24	17.19	241.41	21.63	333.15	17.32
2	RF	164.30±0.40	14.35±0.02	176.26±0.79	14.78±0.15	289.59±1.38	14.28±0.13
	ABRT	163.35±0.88	14.19±0.07	177.55±1.31	14.69±0.20	288.53±0.57	14.31±0.10
	GBRT	164.12±2.08	14.24±0.12	171.24±6.82	15.34±1.05	283.91±6.75	14.16±0.45
	SVR	180.88	17.06	239.93	22.06	332.73	18.04
	MLP	181.84	17.15	241.98	22.25	335.12	17.62
3	RF	164.49±0.63	14.39±0.05	171.51±1.10	13.76±0.18	291.13±1.05	14.27±0.10
	ABRT	163.82±1.05	14.20±0.06	171.11±1.61	13.51±0.30	288.90±0.22	14.15±0.07
	GBRT	163.26±1.81	14.21±0.12	166.18±7.02	14.43±1.10	283.78±5.37	14.12±0.37
	SVR	180.88	17.06	237.11	21.78	333.12	18.05
	MLP	182.33	17.05	236.61	21.55	328.09	16.83
4	RF	164.50±0.77	14.39±0.06	165.23±1.38	12.57±0.18	291.94±1.14	14.10±0.12
	ABRT	163.09±0.63	14.20±0.06	165.03±1.65	12.18±0.46	290.91±0.30	13.98±0.09
	GBRT	163.04±1.78	14.20±0.15	157.87±7.72	12.62±1.49	283.67±3.90	14.00±0.36
	SVR	180.91	17.06	233.59	21.46	334.26	18.09
	MLP	181.51	17.14	234.90	21.84	319.59	16.44
5	RF	164.43±0.66	14.39±0.05	159.97±1.11	11.57±0.16	290.64±1.40	13.91±0.15
	ABRT	163.18±0.78	14.23±0.06	159.73±2.27	10.90±0.68	290.53±0.96	13.71±0.18
	GBRT	163.43±1.79	14.22±0.13	152.06±5.86	11.49±1.18	285.19±3.71	13.94±0.34
	SVR	181.19	17.10	229.09	20.97	332.58	17.93
	MLP	182.19	17.19	205.55	19.14	329.87	17.25

The tree ensemble models are also compared with the the SVR models and the MLPs fitted in Chapter 4. As is shown in Table 5.7, the tree ensembles models outperform the SVR models and the MLPs for all cases.

5.5 Discussions and Conclusions

This chapter presents the travel time prediction for container trucks using three tree ensembles, namely RF, ABRT and GBRT. The same prediction tasks and datasets were used as those for Chapter 4. All the tree ensemble models were fitted using a batch Bayesian optimisation algorithm.

Departure time, day-of-week, and historical mean travel time are used as the input features for the predictions before departure. The results show that the performances of the three types of tree ensemble models are very close. Next, the predictions for ongoing trips are performed 1, 2, 3, 4, and 5 minutes after a trip starts. The mean speed sequence (MSS) is added to the feature columns. The results show that the error generally decreases with the increase of T for R2 and R3, but not for R1. If a target road segment is relatively short (i.e., less than 10km) and the future traffic condition is stationary, MSS can infer how fast a truck may move on the untravelled part of the road segment, based the mean speed behaviours on the travelled part of the road segment. However, MSS cannot capture the interruption of the traffic caused by some unexpected events (e.g., incident, road work). Therefore, the historical MSS (HMSS) is incorporated into the model so that the possible speed behaviours on the untravelled part of the road segment can be learned based on the speed behaviours of some recent trips.

New OGP experiments were later conducted by adding the HMSS of the last 5 minutes into the feature columns. The results show that the errors are slightly reduced for R1 and R2, but not for R3. The analysis on the trips shows that R3 has more trips in the test data whose departure times are not covered in the training data. Further experiment results show that the error for R3 can be slightly reduced by removing those 'uncovered' trips. This result suggests that the departure time coverage ratio (in the test data) can have an impact on the effectiveness of HMSS.

The computational cost of the tree ensemble models is also evaluated in this study. The computational cost is evaluated in four aspects: the mean tree depth,

the mean nodes count, the model fitting time, and the prediction time. The results show that the trees in the GBRT models are much shallower than those in the RF models and the ABRT models. As a result, the model fitting time and prediction time are much shorter for GBRT than those for RF and ABRT. If the computational cost is a big concern for the application, GBRT is recommended. In general, fitting tree ensemble models is not very fast, particularly if the feature dimensionality becomes large.

The tree ensemble models are also compared with the SVR models and the MLPs on the OGP tasks. The results show that the tree ensemble models outperform the SVR models and the MLPs for all cases.

Chapter 6

An Investigation of Decision Tree Ensembles for Travel Time Prediction Using Various Feature Combinations

6.1 Introduction

In Chapter 5, three types of decision tree (or tree) ensemble models were implemented to predict travel time for container trucks. The results show that although our trip data are temporally sparse and noisy, these tree ensembles are still able to produce satisfying results and outperform MLPs and SVR. Several other studies have been found investigating tree ensembles under different experiment configurations such as different prediction horizons, different trip lengths, and different traffic conditions. As we know, features are important for travel time prediction. However, there is not enough discussion of the effect of various features on the travel time prediction performance in the literature. Although built-in feature importance analysis helps tree ensembles identify important features to a certain degree, im-

proper feature selection can have a negative impact on the performance due to the combinatorial effects of features. Therefore, this chapter investigates the travel time prediction performance by the two popular tree ensembles, random forests (RF) and gradient boosting regression trees (GBRT), using different feature combinations. We first conduct the investigation using pseudo travel time data generated using a pseudo travel time sampling algorithm (PTTS). PTTS allows generating travel time data using different noise processes so that we can study the prediction performance under the effect of different noises. Then, a case study is performed to verify the findings of the investigation using the artificially generated data. Besides, modular based model fitting strategies are examined in both cases to see if they are beneficial to the prediction.

The rest of this chapter is organised as follows: Section 6.2 introduces the pseudo travel time sampling algorithm for generating the artificial travel time data; Section 6.3 explains the experiments with the artificially generated data, including data preparation, input features, model fitting, tasks and performance evaluation metric; Section 6.4 presents the results and analysis; Section 6.5 provides a case study using real-world data to verify the earlier findings in this chapter; the discussions and conclusions are presented in Section 6.6.

6.2 Pseudo Travel Time Sampling

Real-world travel time data are often noisy. It is often difficult to separate noise from travel times due to the lack of understanding of the causal effects of the noises of the input features. Also, the noise can be either non-correlated or temporally correlated (i.e., each observation is not entirely independent of the previous observations and provides less information than an independent or non-correlated observation). Thus, a pseudo travel time sampling algorithm (PTTS) was developed for this study so that different types of noise in travel time prediction can be studied. Various synthetic travel time datasets can be generated and used for this study.

The idea of the PTTS algorithm is summarised in Algorithm 5. A user first specifies the sampling rate (in minutes) per day (R) and the number of days (N_{days}) to sample. Then the total number of (departure time) intervals per day can be calculated as $(60/R) \times 24$. For each day, the user is asked to define a morning peak and an afternoon peak which requires specifying the start interval index and the length of the peak (step 3a and 3b). The result of the definitions are the collections of interval indices for the morning and afternoon peak, respectively. Defining peaks in a day gives us time-of-day variability of travel times. The parameter *corr* determines if the noise should be temporally correlated or not (step 3c). If $corr = False$, the noise is non-correlated, and randomly sampled from a zero-mean Gaussian distribution with noise scale σ which can be specified by the user; if $corr = True$, the noise

Algorithm 5 The pseudo travel time sampling algorithm (PTTS).

1: Specify start date-time DT_0; sampling rate per day R (in minutes); number of days N_{days}; noise scale σ, *corr*.
2: Initialise the collection of travel times $\mathcal{D} = \{\}$
3: For $d = 1$ to N_{days}:

a) Define the morning peak for the dth day by specifying the start interval index I_{PKAM} and the length L_{PKAM}.

b) Define the evening peak for the dth day by specifying the start interval index I_{PKPM} and the length L_{PKPM}.

c) If $corr = False$, sample non-correlated noises for the dth day $\boldsymbol{\epsilon}$:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2)$$

else, sample correlated noises with predefined correlation matrix $\boldsymbol{A}$:

$$\boldsymbol{\epsilon}' \sim \mathcal{N}(0, \sigma^2)$$
$$\boldsymbol{\epsilon} = \boldsymbol{A} \cdot \epsilon'$$

d) Sample travel times for the non-peak and peak given I_{PKAM}, L_{PKAM}, I_{PKPM}, L_{PKPM} and $\boldsymbol{\epsilon}$. Assemble all the travel times for the dth day as $\boldsymbol{TT}_d$.

e) Augment data $\mathcal{D} \leftarrow \{\mathcal{D}, \boldsymbol{TT}_d\}$.

4: Output $\mathcal{D}$

becomes temporally correlated according to a correlation matrix $\boldsymbol{A}$. Then, we can sample travel times for the non-peak and peak, given the defined interval indices and sampled noises (step 3d). The above processes is repeated for N_{days} days. Note that we use the same noise scale for both the non-peak and peak so that we can compare the performance of the tree ensemble models over different traffic conditions. The user can also use different noise scales for different traffic conditions.

6.2.1 Noise

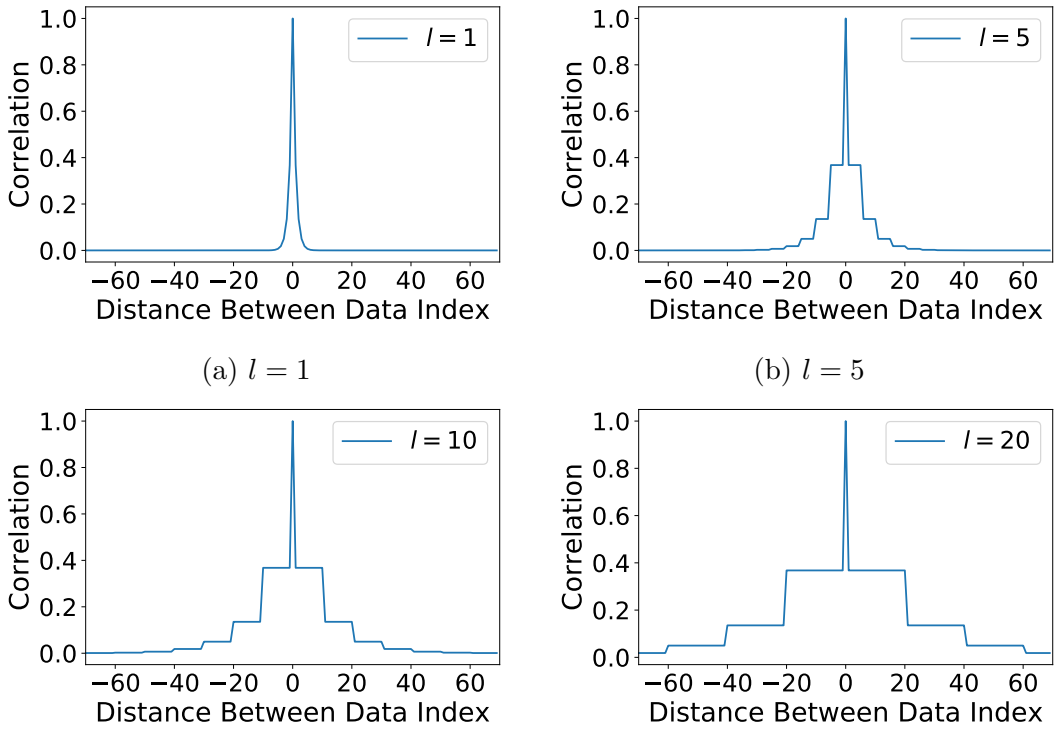
As is described at step 3c in Algorithm 5, a user can either make the noise non-correlated or temporally correlated. If the noise is non-correlated, we assume the noise follows an identical normal distribution, i.e., $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2)$; if the noise is temporally correlated, then the nature of this correlation need to be defined. In this study, the correlated noises will be sampled in three steps: (1) define a correlation matrix $\boldsymbol{A}$; (2) draw n i.d.d. noise $\boldsymbol{\epsilon}'$; (3) get correlated noise $\boldsymbol{\epsilon} = \boldsymbol{A} \cdot \boldsymbol{\epsilon}'$, where $\boldsymbol{A}$ is a $n \times n$ matrix which can be expressed as:

$$\boldsymbol{A} = \begin{bmatrix} C(\epsilon_1, \epsilon_1) & \cdots & C(\epsilon_1, \epsilon_n) \\ \vdots & \ddots & \vdots \\ C(\epsilon_n, \epsilon_1) & \cdots & C(\epsilon_n, \epsilon_n)) \end{bmatrix} \tag{6.1}$$

where $C(\epsilon, \epsilon')$ is the covariance function that is symmetric and positive semi-definite. There are various ways to define C. For this study, a simple stationary parametric covariance function - the exponential covariance function - is used. It can be expressed as:

$$C_{i,j} = \exp\left(\frac{-|i-j|}{l}\right) \tag{6.2}$$

where i, j are the data point index; l is the correlation length scale that can be specified by the user. As is shown in Figure 6.1, the length scale regularises the



(a) $l = 1$ (b) $l = 5$

(c) $l = 10$ (d) $l = 20$

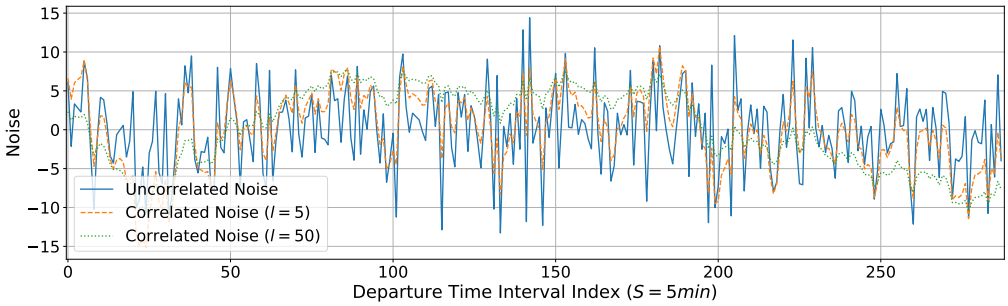
Figure 6.1: The exponential covariance functions with different length scales.

smoothness of the covariance function. The larger the length scale, the larger the correlation is between two data points.

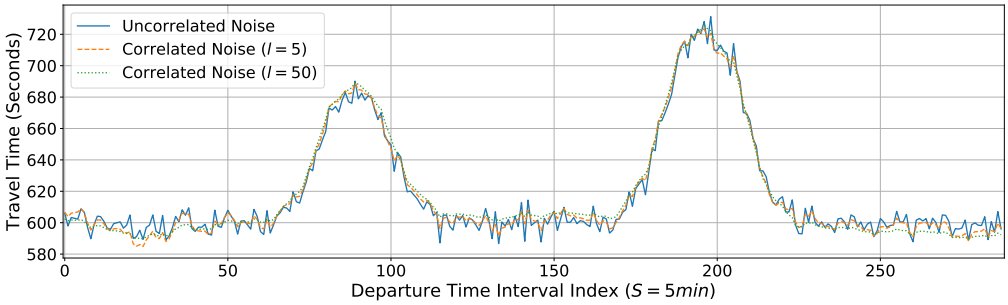
Figure 6.2a shows the example of one non-correlated noise and two correlated noises with different correlation length scales for a day. It can be seen that the non-correlated noise fluctuates around the mean zero. The two correlated noises are locally smoother than the non-correlated one, which results in smoother travel time trends as are shown in Figure 6.2b.

6.2.2 The Non-peak Travel Time Sampling

In this study, we assume the traffic in non-peak is under the free-flow condition where vehicles move at their *desired (free)* speeds. Given a departure time interval



(a) Noise.



(b) Single-day travel times.

Figure 6.2: The examples of single-day travel times with non-correlated and correlated noises.

t that is in a non-peak, the travel time can be modelled as:

$$TT_t = \boldsymbol{E}[TT_{free}] + \epsilon \tag{6.3}$$

where $\boldsymbol{E}[TT_{free}]$ is the expected free-flow travel time and ϵ is the i.d. noise which can be interpreted as the variance of travel time that is largely due to the difference in the choice of desired speed. The above equation also can be written as $TT_t \sim N(\boldsymbol{E}[TT_{free}], \sigma^2)$ if the noise ϵ is non-correlated and follows a normal distribution.

6.2.3 The Peak Travel Time Sampling

Given a departure time interval t that is in a peak, the travel time at t can be modelled as:

$$TT_t = TT_{t-1} + \nu_t + \epsilon \tag{6.4}$$

where TT_{t-1} is the travel time observed from the last departure time interval; ν_t is the noiseless difference between TT_t and TT_{t-1}; ϵ is the i.d. noise. Adding ν_t allows us to create a local linear trend for travel times, but the overall trend can be either linear or non-linear depending on how ν_t is defined.

A peak can be further divided into three sub-phases: free-flow to congestion, congestion, and congestion to free-flow. Travel time goes up in the free-flow to congestion phase and decreases in the congestion to free-flow phase. A long queue is usually found in the congestion phase in which most vehicles move approximately at the same speed. Such speed is known as the flow speed. The following piecewise-

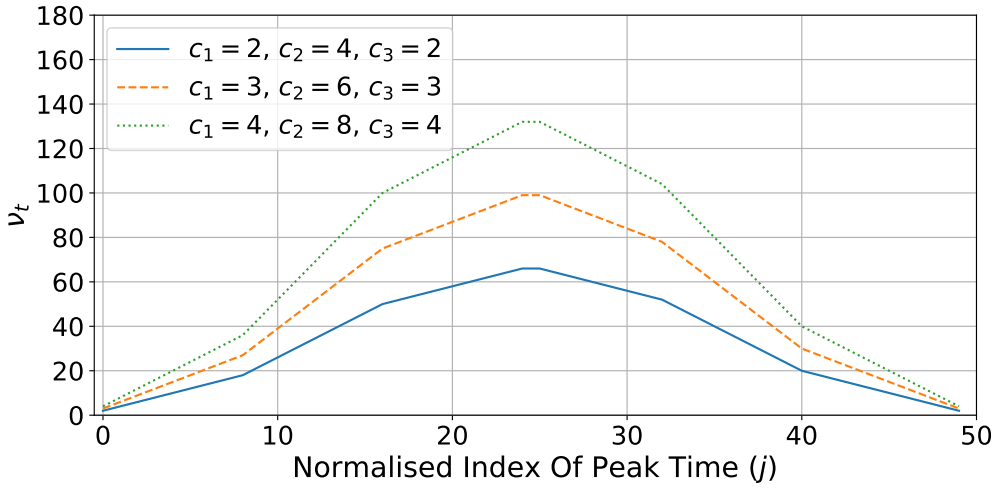


Figure 6.3: The examples of the ν_t function. The peak length: $L = 50$.

linear function is used to model ν_t for peak travel time for this study:

$$\nu_t = \begin{cases} c_1 & \text{if } 0 \leq j \leq \frac{L}{6} \text{ or } \frac{5L}{6} \leq j < L \\ c_2 & \text{if } \frac{L}{6} < j \leq \frac{L}{3} \text{ or } \frac{2L}{3} \leq j < \frac{5L}{6} \\ c_3 & \text{if } \frac{L}{3} < j < \frac{L}{2} \text{ or } \frac{L}{2} < j < \frac{2L}{3} \\ 0 & \text{if } j = \frac{L}{2} \end{cases} \tag{6.5}$$

where L is the length of the peak; $j = t - i_{start}$ is the normalised interval index; i_{start} is the start interval index of the peak; c_1, c_2 and c_3 are the user-defined constants that control the change rate of travel time in the peak. The traffic reaches the congestion phase in the middle of the peak. Here we assume most vehicles move at the speed of the congested traffic flow. Thus we set the change rate to zero. Figure 6.3 shows the examples of the ν_t functions with change rate configurations.

6.3 Experimentation

6.3.1 Datasets

To generate datasets for this study, we set the sampling rate $R = 5$ minutes. Hence, there are $(60/5) \times 24 = 288$ intervals per day where the interval index ranges from 0 to 287. DT_0 is set to '2016-02-01 00:00:00' and N_{days} is set to 35 days, which is equivalent to 5 weeks. The start interval indices and lengths for peaks are shown in Table 6.1. Eight ν_t functions are defined (see Table 6.2) so that we can create different peaks on different day-of-week (see Table 6.3). With these settings, both the time-of-day and day-of-week variability patterns can be created in each travel time dataset. Table 6.4 shows the configuration for noise sampling which helps us to sample travel times with different noise process. Each dataset contains 5 weeks of travel time data. The first 4 weeks of data is used for the training and the last one week data is used for the test. One-step-ahead prediction task is selected for

this study.

Table 6.1: The start interval indices and the lengths for peaks.

Peak	Setting	MON	TUE	WED	THU	FRI	SAT	SUN
Morning	Start Index	80	83	81	85	82	105	108
	Length	40	42	44	40	46	36	38
Afternoon	Start Index	190	193	188	191	189	207	211
	Length	44	46	48	44	50	40	42

Table 6.2: The ν_t functions with different parameter settings.

Function	Parameter Setting	Function	Parameter Setting
$\nu 1_t$	$c_1 = 1$, $c_2 = 3$, $c_3 = 1$	$\nu 5_t$	$c_1 = 3$, $c_2 = 6$, $c_3 = 3$
$\nu 2_t$	$c_1 = 2$, $c_2 = 4$, $c_3 = 2$	$\nu 6_t$	$c_1 = 3$, $c_2 = 9$, $c_3 = 3$
$\nu 3_t$	$c_1 = 2$, $c_2 = 6$, $c_3 = 2$	$\nu 7_t$	$c_1 = 2$, $c_2 = 12$, $c_3 = 2$
$\nu 4_t$	$c_1 = 2$, $c_2 = 8$, $c_3 = 2$	$\nu 8_t$	$c_1 = 3$, $c_2 = 18$, $c_3 = 3$

Table 6.3: The ν_t functions for peaks on each day-of-week.

Peak	MON	TUE	WED	THU	FRI	SAT	SUN
Morning	$\nu 3_t$	$\nu 5_t$	$\nu 6_t$	$\nu 4_t$	$\nu 7_t$	$\nu 1_t$	$\nu 2_t$
Afternoon	$\nu 4_t$	$\nu 6_t$	$\nu 7_t$	$\nu 5_t$	$\nu 8_t$	$\nu 2_t$	$\nu 3_t$

Table 6.4: The configuration of the noises sampling.

Noise Type	Configuration
Non-correlated	σ Values: $[1, 5, 10, 20, 50]$
Correlated	Length scale l, values: $[1, 5, 10, 20]$ σ, values: $[1, 5, 10, 20, 50]$

6.3.2 Features

In travel time studies, features can be generally categorised into four types: date-time indicator, historical descriptive statistics, real-time data, and features derived using other features. The descriptions of these feature types are shown in Table 6.5. Note that a derivative feature can be based on historical data, real-time data, or both of them. In the rest of this thesis, the abbreviations provided in Table 6.5 will be

used when referring to the different feature types. Table 6.6 shows the features that are selected for this study. Various feature combinations can be created out of these features for this study. Because we perform the prediction for a given departure

Table 6.5: General feature types.

Type	Abbr	Description
Data-time indicator	T	Date-time relevant input features, e.g., departure time, day-of-week, month, day-of-month, day-of-year, year.
Historical descriptive statistic	H	Historical descriptive statistic of a travel time sample, e.g., historical mean, median, percentiles, standard deviation, coefficient of variance.
Real-time data	R	Real-time traffic flow data (e.g., flow rate, speed, volume) or travel time observations from previous departure time intervals.
Derivative feature	D	Feature that is derived using other input features, e.g., predicted travel times or traffic flow, the difference between two consecutive travel time observations from previous departure time intervals.

Table 6.6: Features used in this study.

Type	Name	Description	Value Type
T	IND	Departure time interval index	discrete, integer
	DOW	Day-of-week	discrete, integer
H	MTT_t	Historical mean travel time of the current departure time interval, given the same DOW	continuous, real
R	TT_{t-1}	Travel time observation from the last departure time interval (1-step backward)	continuous, real
	TT_{t-2}	Travel time observation from the departure time interval 2-step backward	continuous, real
	TT_{t-3}	Travel time observation from the departure time interval 3-step backward	continuous, real
D	Δ TT_{t-1}	TT_{t-1} - TT_{t-2}	continuous, real
	Δ TT_{t-2}	TT_{t-2} - TT_{t-3}	continuous, real

date-time, features of the type T will be used in all the feature combinations.

6.3.3 Model Fitting Strategies

In many studies [104, 242, 41], the training data are first partitioned into several subsets using a clustering technique (e.g., k-means), then a model is fitted to each subset using advanced learning algorithm such as ANNs. Their results show that such hierarchical learning approaches are beneficial to travel time prediction. In this study, three model fitting strategies are examined, which are described as follows:

- **Regular approach (REG)** fits a tree ensemble model using the entire training data set. This approach results in one prediction model for all the test cases.
- **Modular approach 1 (MOD1)** first partitions the training data into three subsets according to their traffic states (i.e., non-peak, morning peak, afternoon peak), then a tree ensemble model is fitted to each subset. This approach results in three prediction models. To make the prediction on a test case, we first identify the traffic state of the test case; then the corresponding prediction model is selected to make the prediction.
- **Modular approach 2 (MOD2)** is similar to MOD1. The only difference is that the traffic state in a peak is further divided into free-flow to congestion (including congestion) and congestion to free-flow, without differentiating morning and afternoon peak. This approach results in four prediction models.

The hyperparameters for RF and GBRT are the same as those of Chapter 5. The batch Bayesian optimisation via local penalisation (BBO-LP) is also used to find the best values of the model hyperparameters. For more detailed description on the hyperparameters tuning approach, please refer to Section 5.2.4.

6.3.4 Performance Evaluation Metric

The root mean squared error (RMSE) is selected as the metric for the prediction performance evaluation. The general form of RMSE can be expressed as follows:

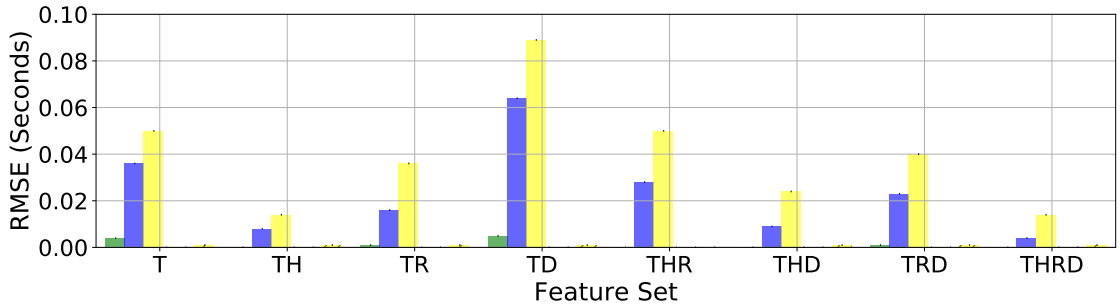
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{6.6}$$

where n is the number of test cases, y_i and $\hat{y}_i$ are the ith true and predicted travel time, respectively.

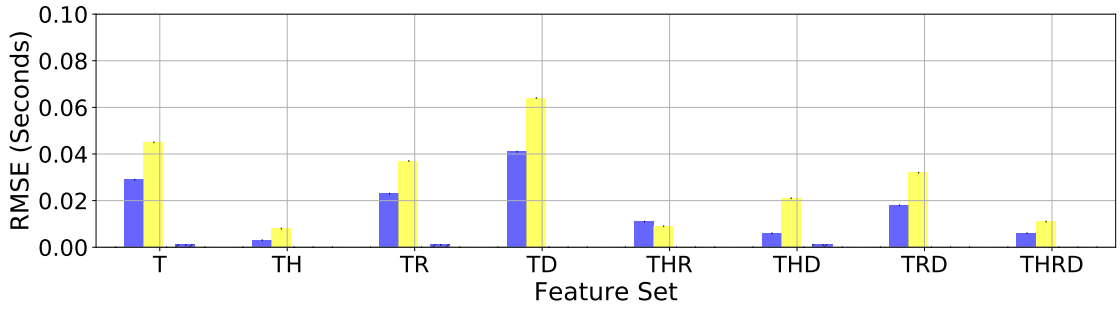
6.4 Results and Analysis

6.4.1 Results Using Noise-free Data

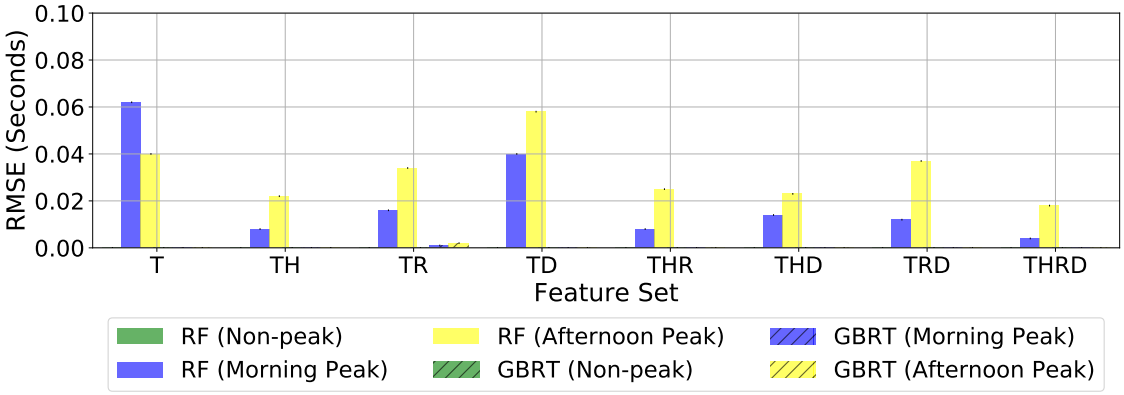
We first check the performance of noise-free travel time data. In this case, there is no variation of travel time for the given features of type T (departure time interval and day-of-week). Thus, learning the travel time function is about learning the constant mean $\boldsymbol{E}[TT_{free}]$ for non-peak and the ν_t functions for peak. Figure 6.4 shows the RMSE results using different feature combinations, with the three model fitting strategies. It can be seen that the overall performances are very appealing with the RMSE below 0.1 seconds for all cases. This type of performance is more or less expected because the data is 'perfect', without any noise. The GBRT models performed slightly better than the RF models. Further analysis result shows that this is due to the use of partial features for splits when fitting the RF models. Using all the features for splits yielded result as good as the GBRT models, but this can result in more similar trees thus increases the sensitivity of the tree ensemble to the noise of a single tree. For RF, the combinatorial features of type T and H (TH) turns out to be the best combination. Figure 6.4 also shows that errors are slightly smaller in the combinations with TH than those without. Since all the combinations have T



(a) REG



(b) MOD1



(c) MOD2

Figure 6.4: The RMSE results using the noise-free data.

by default, this indicates that the feature of type H, i.e., historical mean travel time, is beneficial to the prediction. For GBRT, all the combinations do equally well as the errors are all approximately equal to zero. Applying modular based model fitting approaches did reduce the error when TD or THR is used, but such improvement is not found in other combinations.

6.4.2 Results Using Data with Non-correlated Noises

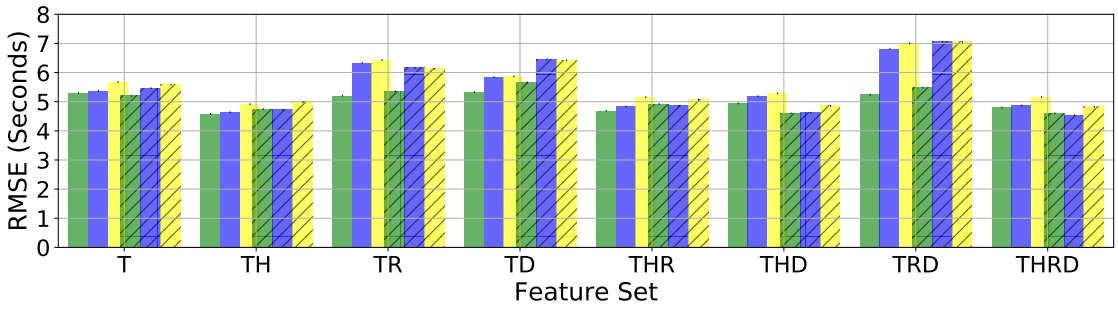
Next, the performance is examined using non-correlated noises of different scales. The results show that the performance of the RF and the GBRT models are very similar for all noise scales. When REG is used to fit models, it is difficult to tell which one is generally better than the other; when modular based approaches are used, GBRT performs slightly better than RF particularly for peak travel time prediction. Table 6.7 summarises the best combinations (smallest total RMSE) for each scenario (i.e., noise scale + fitting strategy + tree ensemble). For RF, both TH and THRD are considered the best combination as both result in the best performance the most often. For GBRT, TH turns out to be the best combination. As is shown in Figure 6.5, errors are slightly smaller in the combinations with the features of type H than those without for both RF and GBRT. The benefit of the features of type H can be consistently observed for all the other noise scales. That is similar to the finding using noise-free data that the historical mean travel time is beneficial to the prediction. Applying the two modular based model fitting approaches (MOD1 and MOD2) did not result in a performance boost, on the contrary, the errors got slightly increased in some cases.

Figure 6.6 and Figure 6.7 show the relation between the error and the involvement of the features of type H, R, and D. While Figure 6.6 shows the errors for non-peaks, Figure 6.7 shows the errors for morning peaks. The dashed lines in Figure 6.7 refer to the concurrence of multiple feature types. It can be seen that although for both RF and GBRT, having the historical mean travel time in the feature columns is beneficial to the prediction, combining the features of type R or D with the features of TH may increase the error. This result indicates that there is a negative combinatorial effect of the features of type H and type R or D. Such negative effect also can be observed in the result for peak time (see Figure 6.7). Although combining the features of type R or D with the features of TH may increase the error, combining the features of RD with the features of TH yields relatively more stable results than those using

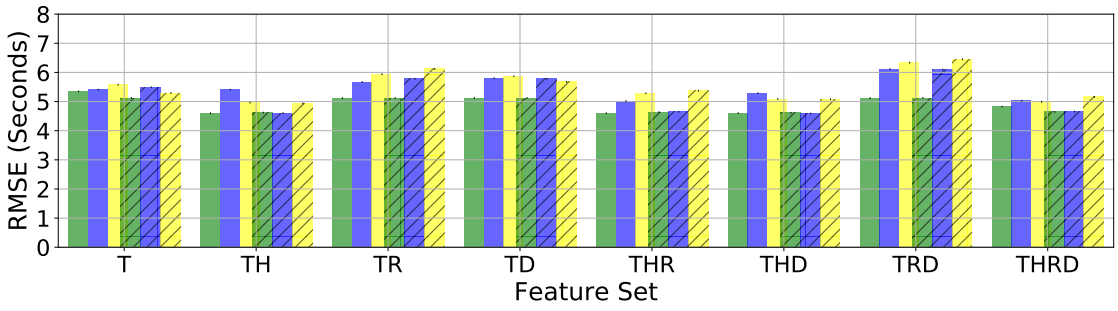
Table 6.7: The best feature combinations using the data with non-correlated noises.

Noise Scale (σ)	Strategy	Ensemble	Best	RMSE		
				Non-peak	Morning Peak	Afternoon Peak
1	REG	RF	THR	0.872	0.989	0.949
		GBRT	THD	0.892	1.046	1.017
	MOD1	RF	THRD	0.874	1.048	1.019
		GBRT	TH	0.950	1.042	1.091
	MOD2	RF	THRD	0.907	1.053	1.008
		GBRT	TH	0.871	1.032	0.976
5	REG	RF	TH	4.588	4.639	4.914
		GBRT	THRD	4.602	4.540	4.820
	MOD1	RF	THRD	4.830	5.028	4.993
		GBRT	TH	4.621	4.604	4.936
	MOD2	RF	THR	4.600	4.746	5.072
		GBRT	THR	4.622	4.648	5.199
10	REG	RF	TH	8.535	8.902	9.741
		GBRT	THR	8.580	8.871	9.764
	MOD1	RF	TH	8.909	9.128	10.193
		GBRT	THD	8.656	9.008	10.319
	MOD2	RF	TH	8.643	9.178	10.233
		GBRT	TH	8.616	8.889	9.910
20	REG	RF	TH	17.551	17.311	18.678
		GBRT	TH	17.499	17.327	18.249
	MOD1	RF	THD	17.639	18.861	19.898
		GBRT	TH	17.669	17.755	18.933
	MOD2	RF	THD	18.253	17.986	19.440
		GBRT	TH	17.700	17.818	18.840
50	REG	RF	THD	44.277	46.842	43.809
		GBRT	THD	44.506	46.863	43.573
	MOD1	RF	THRD	44.368	50.114	46.572
		GBRT	TH	44.465	49.457	43.521
	MOD2	RF	THRD	44.858	49.640	45.998
		GBRT	TH	44.464	47.072	44.197

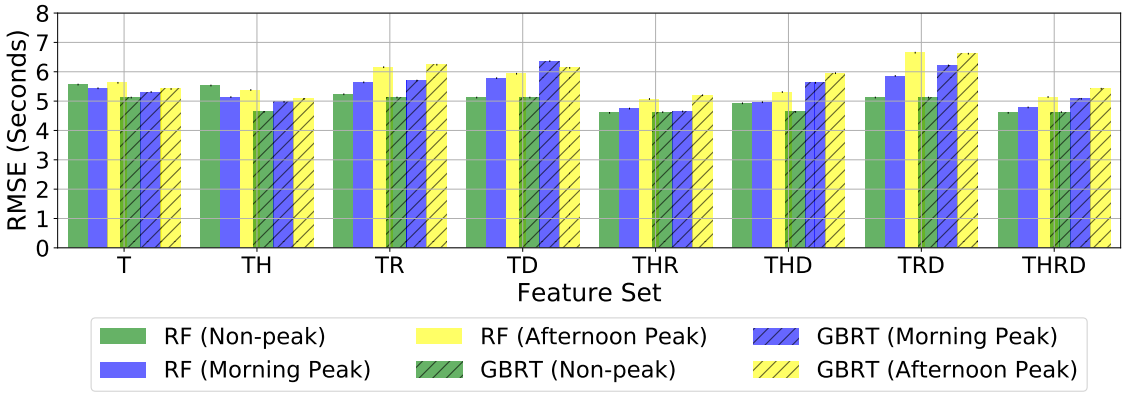
only the features of TH in our study. As is shown in Figure 6.6f and Figure 6.7f, TH yields unexpectedly large errors when REG is used as the model fitting strategy. This finding is consistent with other noise scales.



(a) REG



(b) MOD1

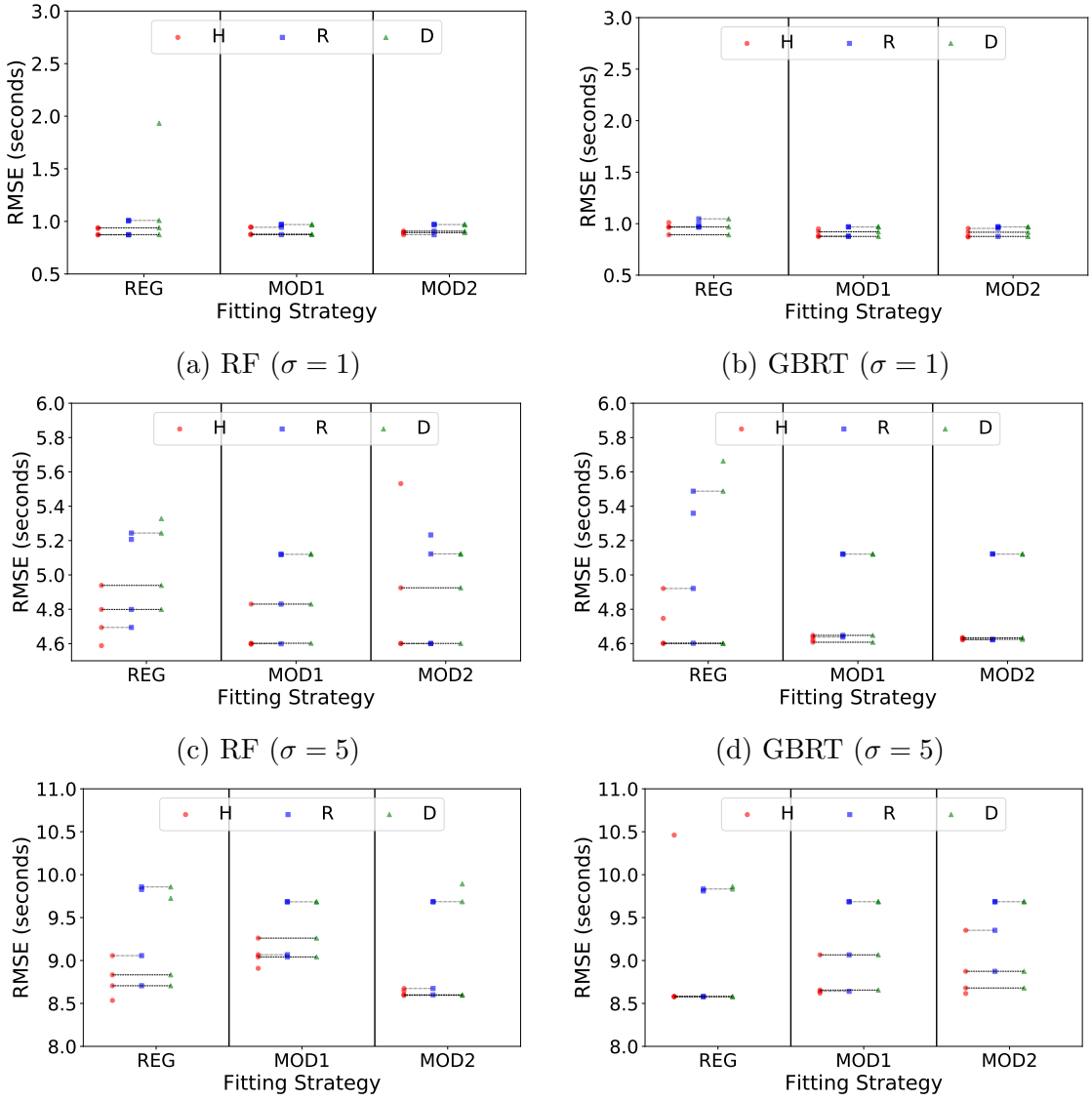


(c) MOD2

Figure 6.5: The RMSE result using the data with non-correlated noise. The noise scale $\sigma = 5$.

6.4.3 Results Using Data with Temporally Correlated Noises

The performance is also examined using the data with correlated noises of different scales. Four different correlation length scales (1, 5, 10, and 20) are selected to populate the correlation matrix $\boldsymbol{A}$ to generate temporally correlated noises. The



(a) RF ($\sigma = 1$)

(b) GBRT ($\sigma = 1$)

(c) RF ($\sigma = 5$)

(d) GBRT ($\sigma = 5$)

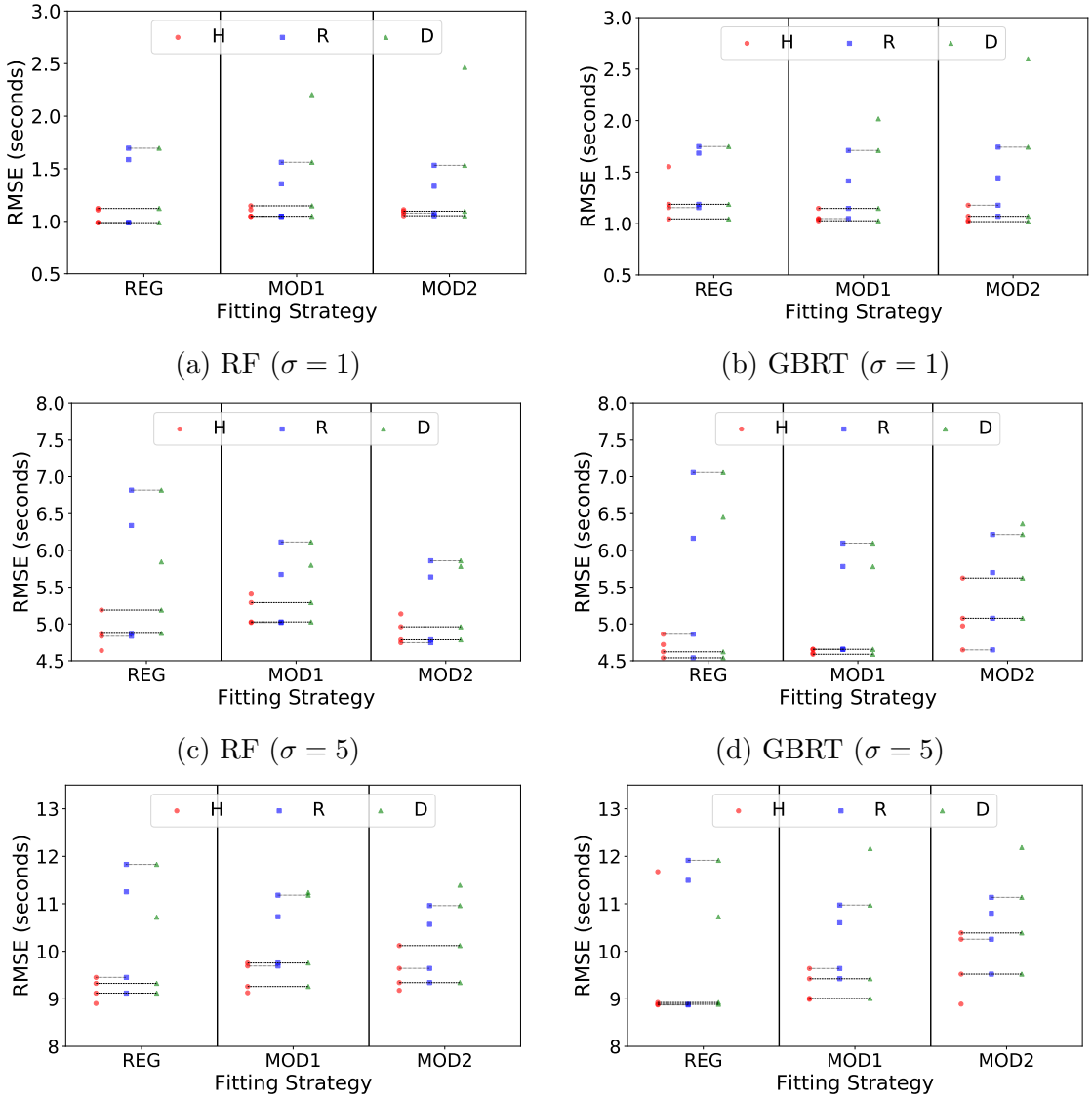
(e) RF ($\sigma = 10$)

(f) GBRT ($\sigma = 10$)

Figure 6.6: The relation between the error and the involvement of the features of type H, R and D for the travel time prediction of non-peaks. Noise scale σ: 1, 5, 10.

degree of correlation between two noise data points is computed using Equation (6.2). The smaller the correlation length scale is, the larger the degree of correlation is between the two points.

The results show that the performance of the RF and GBRT models are generally very similar for all noise scales. Similar to the results using non-correlated



(a) RF ($\sigma = 1$)

(b) GBRT ($\sigma = 1$)

(c) RF ($\sigma = 5$)

(d) GBRT ($\sigma = 5$)

(e) RF ($\sigma = 10$)

(f) GBRT ($\sigma = 10$)

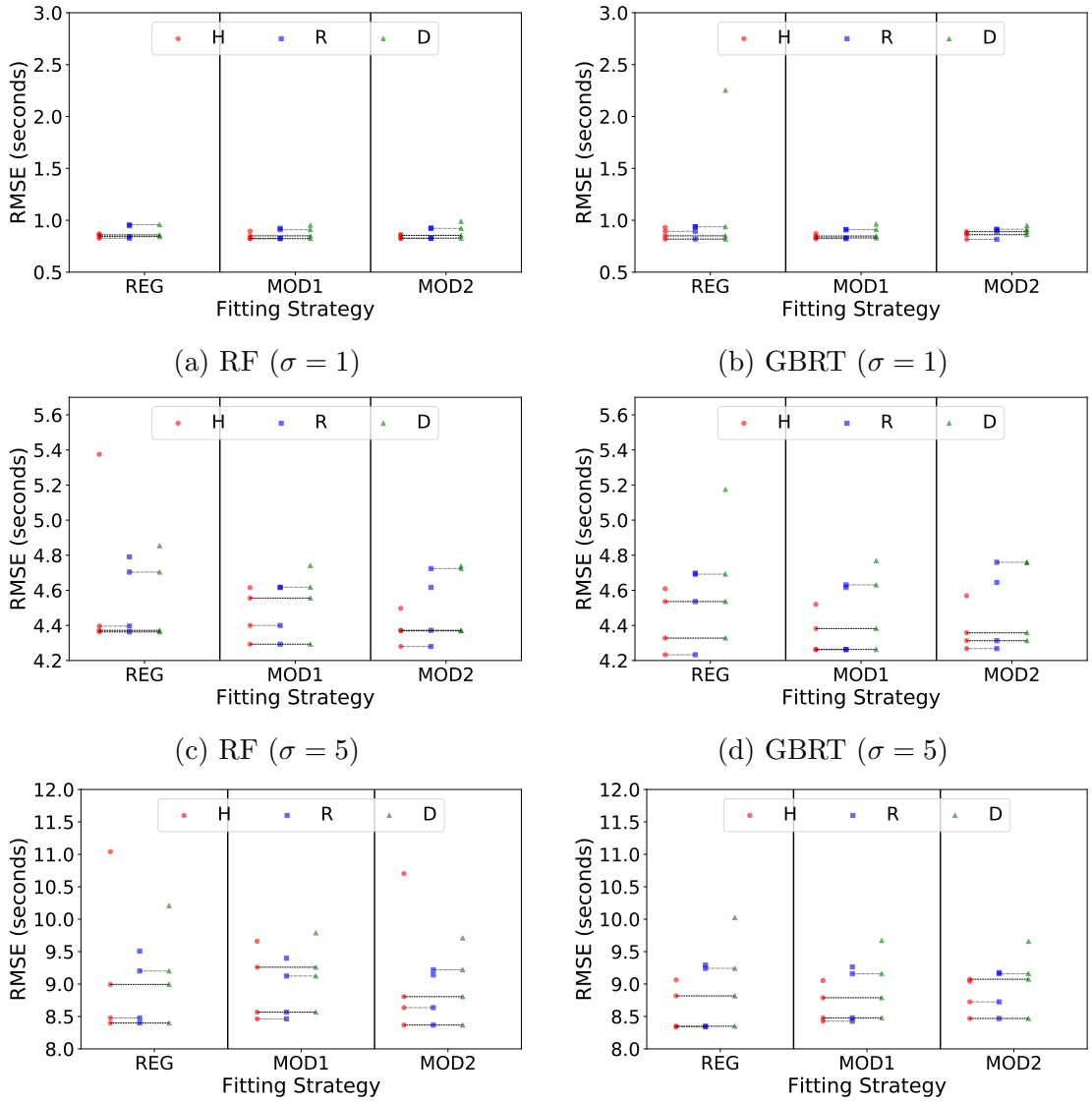
Figure 6.7: The relation between the error and the involvement of the features of type H, R and D for the travel time prediction of morning peaks. Noise scale σ: 1, 5, 10.

noises, when modular based model fitting strategies are used, the GBRT models perform slightly better than the RF models in general, particularly for the travel time prediction of peaks. Table. 6.8 summarises the best combinations for each scenario when $l = 1$. For both RF and GBRT, THR and THRD are considered the best combination as both results in the best performance the most often. The errors

Table 6.8: The best feature combinations using data with temporally correlated noises. Correlation length scale $l = 1$.

Noise Scale (σ)	Strategy	Ensemble	Best	RMSE		
				Non-peak	Morning Peak	Afternoon Peak
1	REG	RF	THD	0.859	1.065	0.986
		GBRT	THRD	0.819	1.085	0.973
	MOD1	RF	THRD	0.823	1.100	1.052
		GBRT	THD	0.848	1.085	0.997
	MOD2	RF	THRD	0.828	1.113	1.048
		GBRT	THRD	0.890	1.156	1.103
5	REG	RF	THRD	4.363	4.315	5.043
		GBRT	THR	4.233	4.044	4.658
	MOD1	RF	THRD	4.293	4.418	4.899
		GBRT	THRD	4.263	4.346	4.813
	MOD2	RF	THR	4.280	4.329	4.944
		GBRT	THRD	4.313	4.479	4.941
10	REG	RF	THR	8.477	8.570	8.911
		GBRT	THRD	8.349	8.402	8.938
	MOD1	RF	THR	8.463	9.030	9.650
		GBRT	THR	8.427	9.202	9.313
	MOD2	RF	THRD	8.367	9.139	9.802
		GBRT	THRD	8.469	8.580	9.346
20	REG	RF	THR	17.236	16.469	17.844
		GBRT	THR	16.969	16.105	17.063
	MOD1	RF	THR	17.111	16.762	18.635
		GBRT	THRD	17.254	16.396	17.558
	MOD2	RF	THR	17.102	17.021	17.901
		GBRT	THRD	17.040	16.382	17.592
50	REG	RF	THR	41.728	43.443	42.121
		GBRT	THRD	41.989	43.320	42.564
	MOD1	RF	THRD	42.411	45.222	45.226
		GBRT	THRD	41.794	45.777	44.312
	MOD2	RF	THR	43.310	45.204	43.128
		GBRT	THRD	42.336	44.587	43.020

are slightly smaller than those using non-correlated noises. Figure 6.8 and Figure 6.9 show the relation between the error and the involvement of H, R and D in the prediction of both non-peaks and morning peaks, respectively. It turns out that for both RF and GBRT, the errors are significantly smaller in the combinations with



(a) RF ($\sigma = 1$)

(b) GBRT ($\sigma = 1$)

(c) RF ($\sigma = 5$)

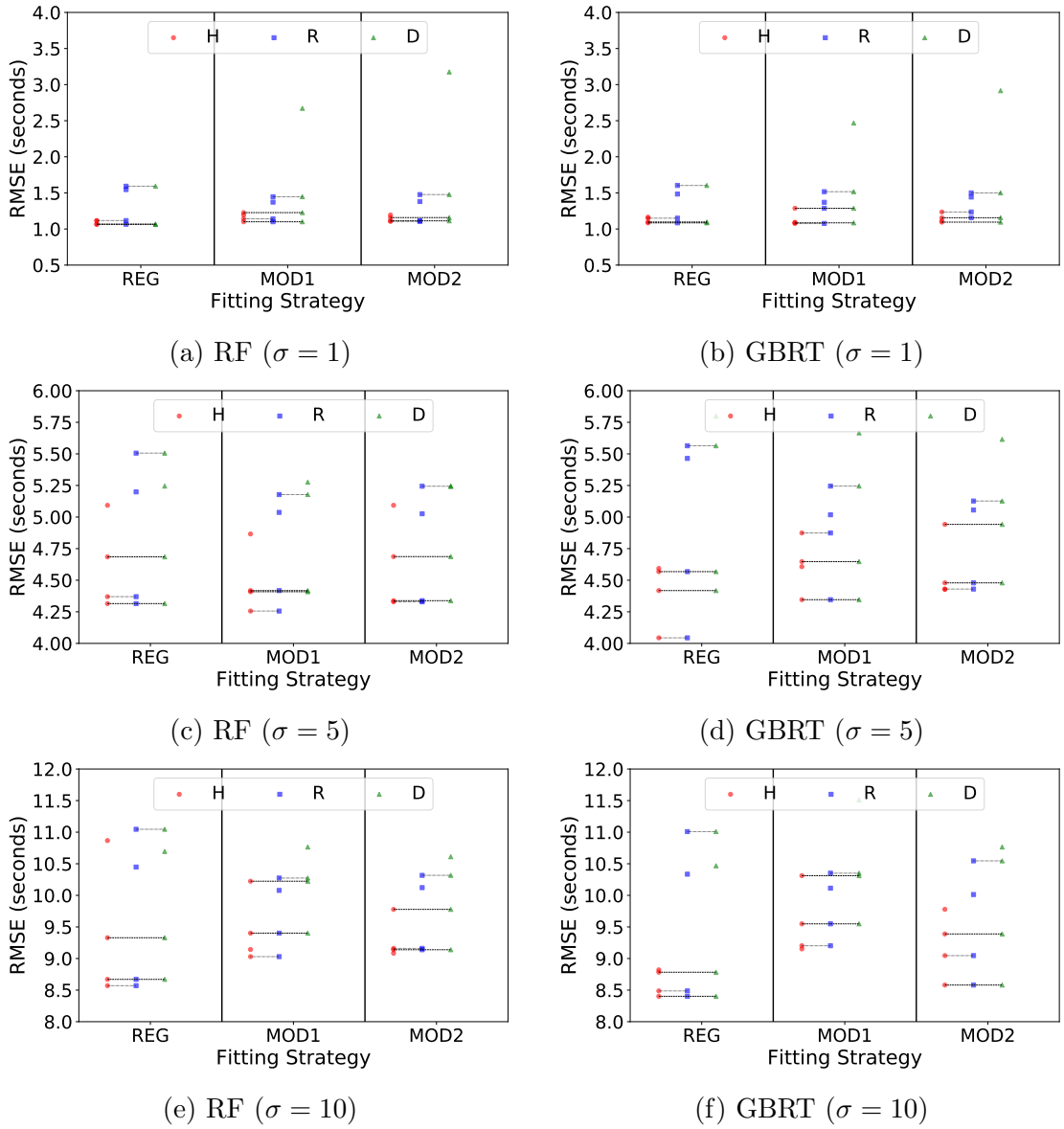
(d) GBRT ($\sigma = 5$)

(e) RF ($\sigma = 10$)

(f) GBRT ($\sigma = 10$)

Figure 6.8: The relation between the error and the involvement of the features of type H, R and D for the travel time prediction of non-peaks. Noise scale σ: 1, 5, 10; correlation length scale $l = 1$.

HR than those without. This result indicates that there is a positive combinatorial effect of the features of type H and R which is beneficial to the prediction. This positive combinatorial effect can be consistently observed for all noise scales. Applying the modular based model fitting strategies did not yield any practically significant performance boost.



(a) RF ($\sigma = 1$)

(b) GBRT ($\sigma = 1$)

(c) RF ($\sigma = 5$)

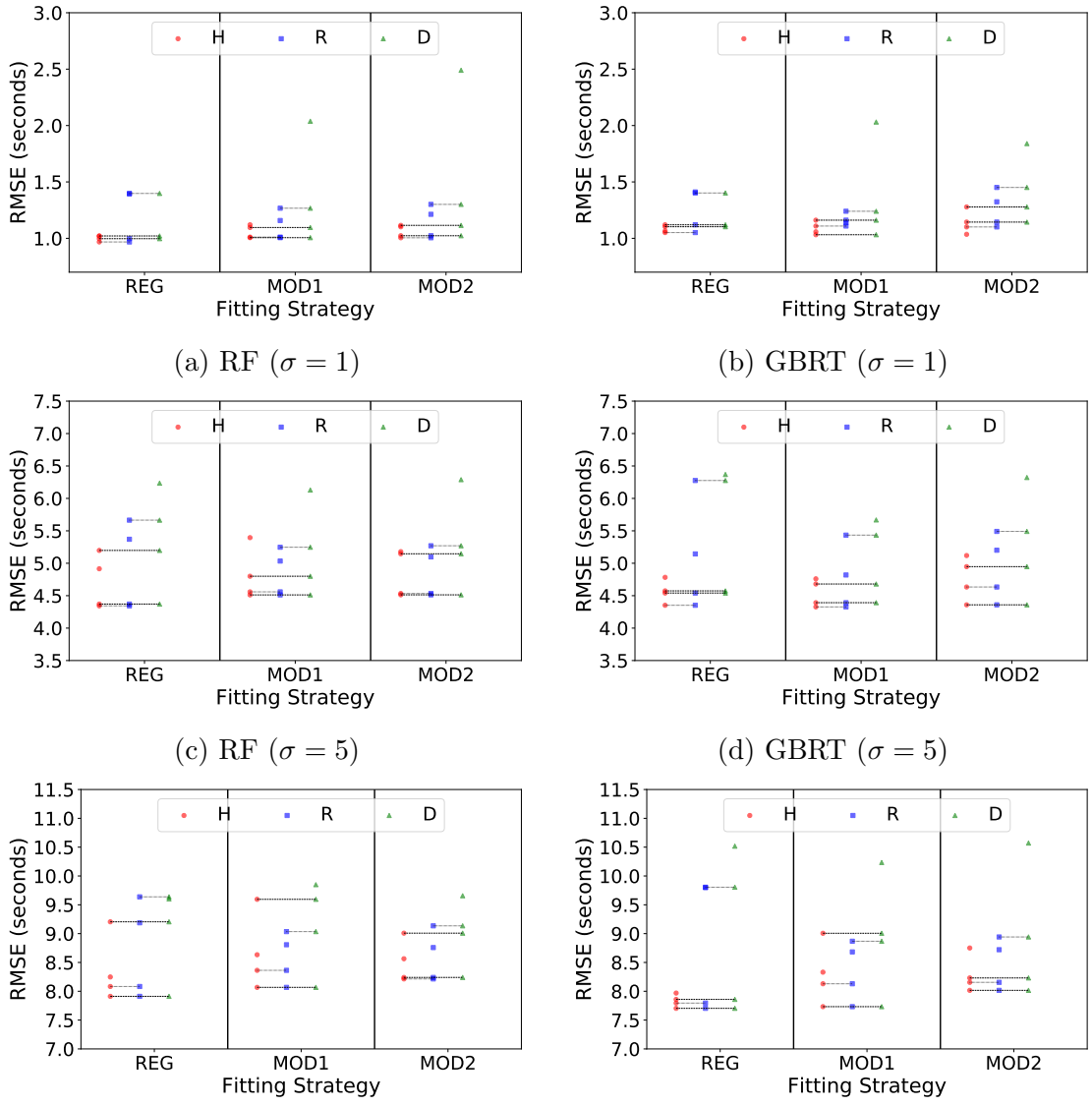
(d) GBRT ($\sigma = 5$)

(e) RF ($\sigma = 10$)

(f) GBRT ($\sigma = 10$)

Figure 6.9: The relation between the error and the involvement of the features of type H, R and D for the travel time prediction of morning peaks. Noise scale σ: 1, 5, 10; correlation length scale $l = 1$.

Further experiments using larger correlation length scales (5, 10, and 20) yield a similar result that THR and THRD are still the best combinations for both RF and GBRT. When l is relatively small, having the features of RD may yield a worse result than that using only the features of R (see Figure 6.10). But such negative combinatorial effect becomes negligible when l becomes larger. As is shown in Fig-



(a) RF ($\sigma = 1$)

(b) GBRT ($\sigma = 1$)

(c) RF ($\sigma = 5$)

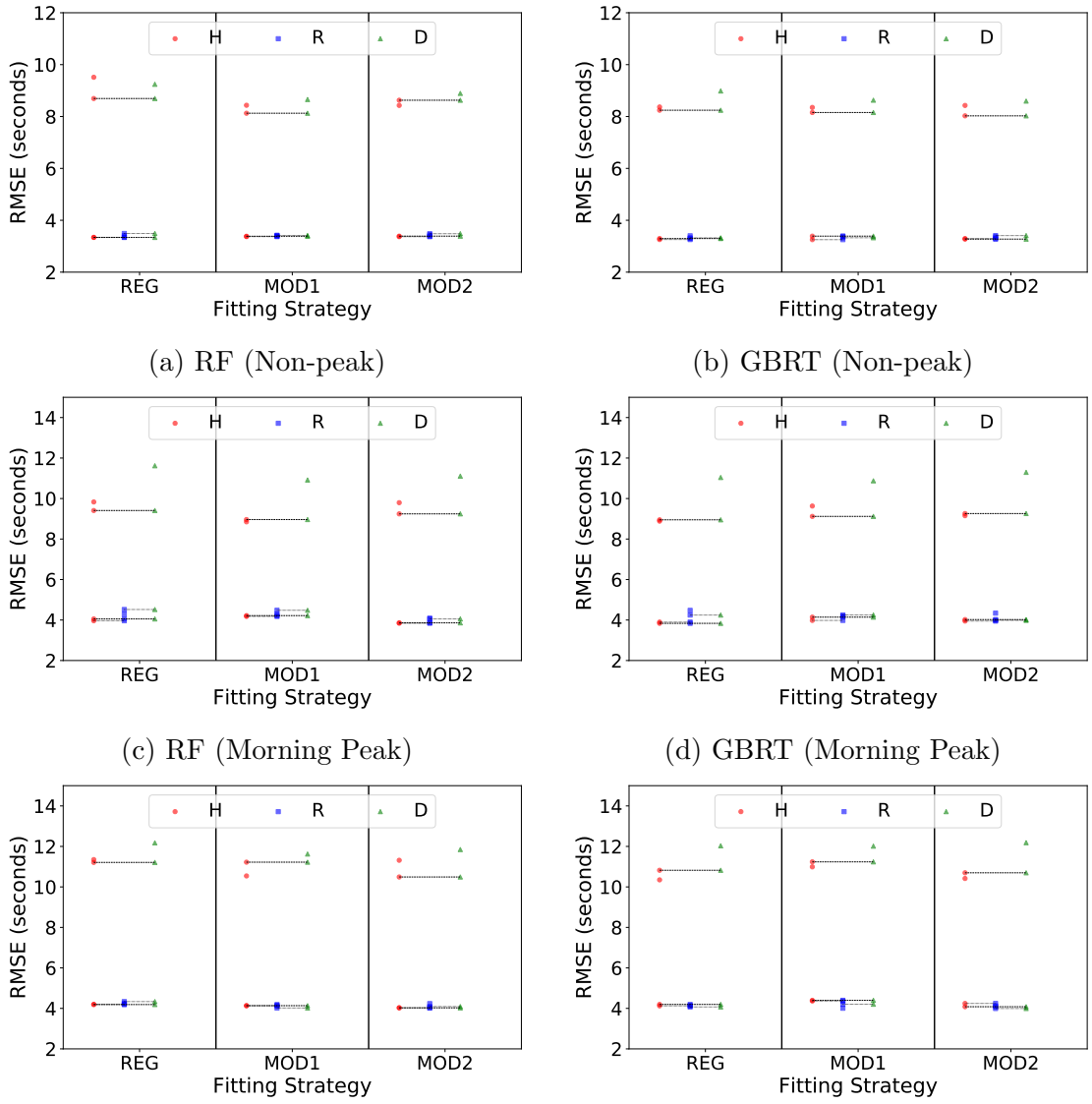
(d) GBRT ($\sigma = 5$)

(e) RF ($\sigma = 10$)

(f) GBRT ($\sigma = 10$)

Figure 6.10: The relation between the error and the involvement of the features of type H, R and D for the travel time prediction of morning peaks. Noise scale σ: 1, 5, 10; correlation length scale $l = 5$.

ure 6.11, the errors are significantly smaller in the combinations with the features of type R than those without. This result indicates that R is becoming more and more beneficial to the prediction as l increases. Similarly, applying the two modular based model fitting strategies does not receive any practically significant performance boost. Note that the errors with temporally correlated noises are smaller



(a) RF (Non-peak)

(b) GBRT (Non-peak)

(c) RF (Morning Peak)

(d) GBRT (Morning Peak)

(e) RF (Afternoon Peak)

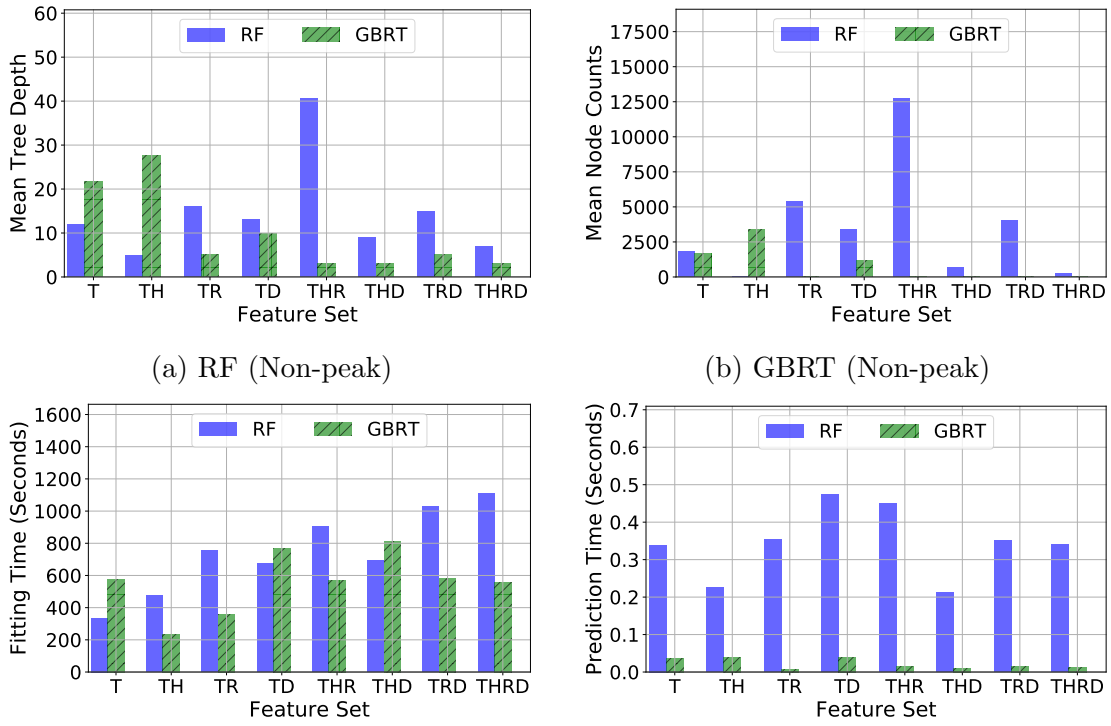
(f) GBRT (Afternoon Peak)

Figure 6.11: The relation between the error and the involvement of the features of type H, R and D for morning peak prediction. Noise scale σ: 10; correlation length scale l: 20.

than those with non-correlated noises, and the error decreases with the increase of l. Since more departure time intervals are correlated when l becomes larger, the features of type R becomes more effective in discovering the temporal correlation between travel time data points.

6.4.4 Comparison of Tree Size and Computation Time Between Random Forests and Gradient Boosting Regression Trees

The tree size, model fitting time, and prediction time were also analysed for both RF and GBRT. As is shown in Figure 6.12, the tree size in terms of the tree depth and nodes count for RF is generally larger than that for GBRT, particularly when more features are used. As a result, the corresponding model fitting time and prediction time for RF is significantly longer than that for GBRT. Note that the two modular based model fitting strategies yield significantly longer model fitting time than the regular approach (REG) and yield no significant advantages in our study.



(a) RF (Non-peak)

(b) GBRT (Non-peak)

(c) RF (Morning Peak)

(d) GBRT (Morning Peak)

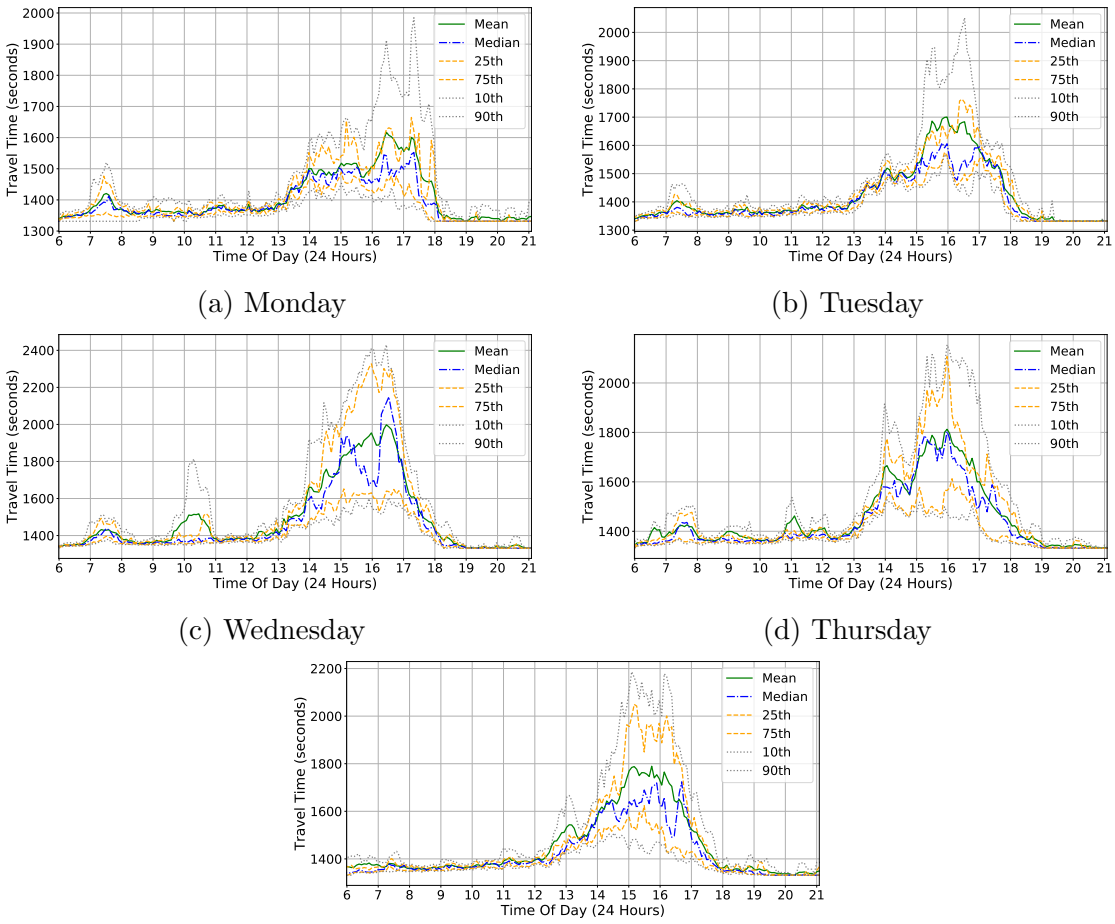
Figure 6.12: The tree size, model fitting time and prediction time of the tree ensemble models with different feature combinations. Noise settings: non-correlated noise, $\sigma = 10$.

Since the prediction models are fitted separately for different traffic conditions, they can be fitted in parallel. That can significantly reduce the model fitting time for the modular based approaches. However, the drawback is also apparent: applying modular based model fitting strategies requires human-effort on identifying the traffic conditions in both the training and test data, i.e., the learning process can not be fully automated. To address this drawback, clustering technique such as k-means might be helpful in automatically finding out the possible traffic conditions.

6.5 Case Study

To verify the findings from the last section which were based on artificially generated data, a case study was conducted using real-world data. The data is four weeks weekday (from Monday to Friday) aggregated departure travel times collected from a 22.19 miles long highway segment from I-5 and University St-NB to I-5 and 320th St, the State of Washington, US. The travel times were estimated every 5 minutes using loop detectors. For this case study, we only use the data between 06:00 to 21:00 on each day as the early data analysis results show that the traffic conditions in the rest time on each day are mostly in free-flow conditions, which is too simple and is not our primary interest. The first three weeks data is used for the training and the remaining one-week data is used for the test. Figure 6.13 shows the percentile and mean travel times for each day, based on the training data. Relatively larger peaks can be observed in the afternoon on each day, some small peaks also can be observed in the morning and at the noon hours. The robust skewness analysis [99] on the travel times shows that most of the travel times are positively skewed.

The same feature combinations are used for the case study. Two model fitting strategies, the regular approach (REG) and one modular based approach (MOD), are used for the case study. Because most of the afternoon peaks occur between 12:00 and 18:00, we can label out the departure time intervals between 12:00 and 18:00 first as '*the afternoon peak*' and the rest departure time intervals simply becomes



(a) Monday

(b) Tuesday

(c) Wednesday

(d) Thursday

(e) Friday

Figure 6.13: The percentile and mean travel times between 06:00 - 21:00 on each weekday.

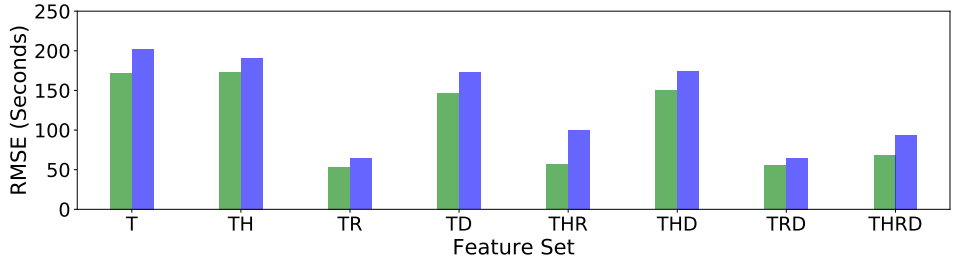
'*the others*'. Thus, the MOD approach first partitions the training data into two subsets: the afternoon peaks and the others; then, a tree ensemble model is fitted to each subset. To make the prediction on a test case, the traffic state of the test case is first identified (i.e., the afternoon peak or the others), then the corresponding prediction model is selected to make the prediction.

The same configuration is used for the BBO-LP based hyperparameters tuning for the tree ensembles. One-step (5 minutes) ahead prediction task is selected to evaluate the model performance. RMSE is selected as the performance evaluation metric. All the models are also implemented using scikit-learn.

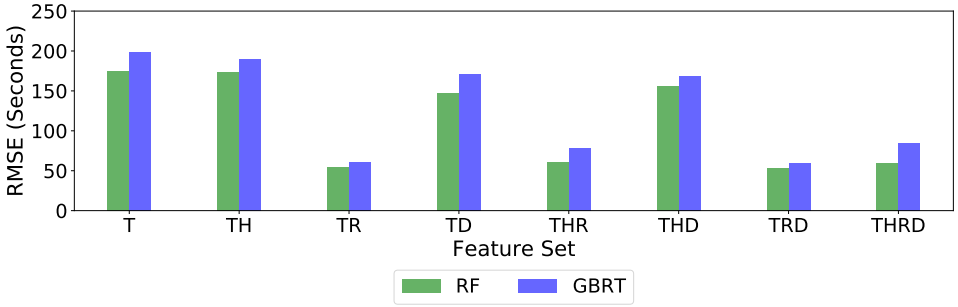
Figure 6.14 and Figure 6.15 shows the overall and piecewise RMSE results of the tree ensemble models using the two model fitting strategies, respectively. It can be seen that the RF models outperform the GBRT models. TR turns out to be the best combination overall. The errors are slightly smaller in the combinations with R than those without. This finding is consistent with that using the artificially generated data (with correlated noises). The historical mean travel time is much less useful than the travel time observations from the previous departure time intervals. That is mainly because the travel time distribution is more skewed and the variation of travel time is more complicated (i.e., given the same departure time interval and day-of-week, the variances are no longer identically distributed) than those in the earlier study. Applying the modular based model fitting approach does not receive any performance boost. This finding is consistent with that using the artificially generated data.

The travel time observations from the previous departure time intervals are essential to the prediction quality. However, these observations are not always available. If no other types of real-time data are available, the prediction entirely relies on the historical data. Since historical mean travel time is not good enough to describe a skewed travel time sample from the perspective of statistics, we can add other historical descriptive statistics such as percentiles, width measures and skewness measures to the feature columns. A further experiment was then conducted by extending the historical descriptive statistics feature (H) to include the mean, the median, four percentiles (10th, 25th, 75th, and 90th), three width measures (interdecile range, interquartile range, and width index [203]) and the standard skewness measure [99]. The results show that the extended H does not yield any performance boost.

Further analysis of the tree size in terms of the tree depth and nodes count shows that the GBRT models have smaller tree sizes than the RF models. As a result, the model fitting time for GBRT is much shorter than for RF. These findings are consistent with those using the artificially generated data.

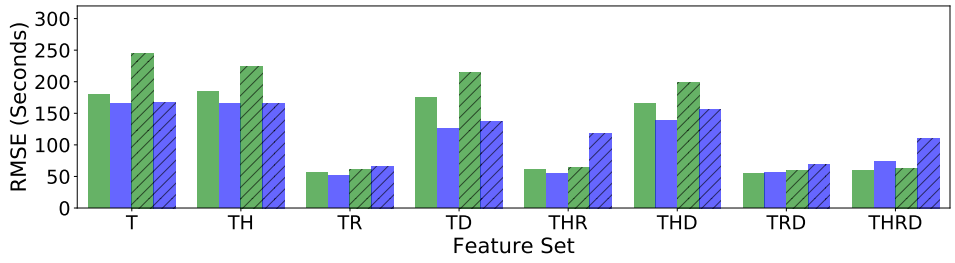


(a) REG

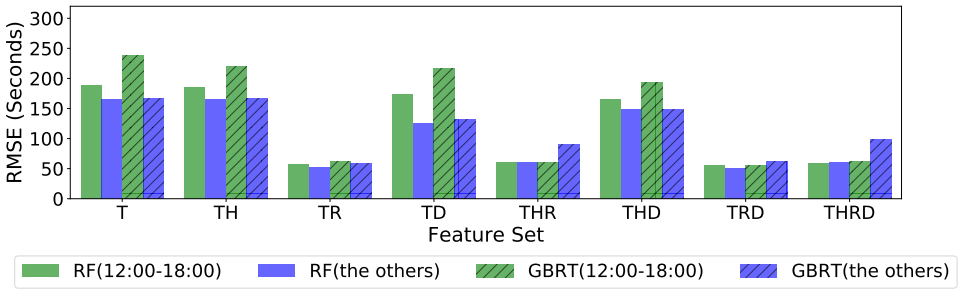


(b) MOD

Figure 6.14: The overall RMSE results using the two model fitting strategies.



(a) REG



(b) MOD

Figure 6.15: The piecewise RMSE results using the two model fitting strategies.

6.6 Discussions and Conclusions

In this chapter, we investigated the travel time prediction performance by two popular tree ensembles, namely RF and GBRT, using different feature combinations. First, the investigation was conducted using pseudo travel time data that was generated using a pseudo travel time sampling algorithm (PTTS) that we developed. PTTS allows generating travel time data using different noise processes so that we can study the prediction performance under the effect of different noises. Besides, two modular based model fitting strategies are examined to see if they are beneficial to the prediction. A case study was later conducted to verify the findings of the investigation using the artificially generated data. The findings are summarised below:

- The results using the artificially generated data show that historical mean travel time can be very beneficial to the prediction if the travel time distribution of a given departure time interval and day-of-week is not statistically skewed and the variances of travel time are identically distributed. However, this is hardly to be the case in the real-world cases. As is shown in the result of the case study, historical mean travel time is not very useful as the travel time distribution is more skewed and the variance of travel time is more complicated over time.

- Both the results using the artificially generated data and real-world data show that travel time observations from the previous departure time intervals are beneficial to the prediction. We strongly suggest to add them as the input features, mainly when no other types of real-time information (e.g., traffic flow, speed) are available. Both results also show that the changes between two consecutive travel time observations from previous departure time intervals are not very beneficial to the prediction.

- Although the tree ensembles can utilise important features to build the predic-

tion model, they can not directly handle the combinatorial effect of features which can have a negative impact on the prediction performance. Thus, it is not wise to dump a bunch of features into the model without much care. Proper feature analysis is still needed before formally building the prediction model.

- Both the results show that applying modular based model fitting strategy does not yield any practically significant performance boost.

- The results of the case study show that the RF models are marginally better than the GBRT models. The tree size analysis shows that trees in the GBRT models are much smaller than those in the RF models. As a result, the model fitting time for GBRT is much shorter than for RF. These findings are consistent with those using the artificially generated data. If the computational time cost is a big concern for the application, GBRT is recommended.

We have learned that the two tree ensembles can not directly handle the combinatorial effects of the features which can have a negative impact on the prediction performance. Our future work focuses on finding a solution for the tree ensembles so that the negative combinatorial effects can be minimised during the model fitting process.

Chapter 7

Conclusions and Future Works

This closing Chapter summarises the main conclusions of the work done in this thesis. The implication of this PhD work for other types of prediction problems is also discussed. The future research plans presented at the end.

7.1 Conclusions

In this thesis, we first performed travel time prediction for freight vehicles using ensemble learning algorithms. One Bayesian probabilistic ensemble algorithm and three decision tree (or tree) ensemble algorithm were selected for the study. These ensemble algorithms assemble multiple single models and usually perform better than single model algorithms. Then, a more detailed investigation was presented on two tree ensembles (random forests and gradient boosting regression trees). The study focused on investigating the travel time prediction performance of these two tree ensembles using several different feature combinations. The effect of the data noise was also studied. Recall that this thesis aims to: (1) develop some 'off-the-shelf' data-driven methods to predict travel time for freight transportation; (2) create a more comprehensive understanding of how to apply these methods more effectively for general travel time prediction problems. Correspondingly, the conclusion can be summarised into travel time prediction for freight transportation and

ensemble algorithms for travel time prediction.

7.1.1 Travel Time Prediction for Freight Transportation

Freight transportation (logistics) has got rapid growth due to the rise of consumption-driven lifestyle and the acceleration of economic globalisation in recent years. A few studies have been found that focus on travel time prediction for freight transportation, but the proposed methods are usually very simple and lack comparison with more advanced data-driven methods. Typically, a prediction model can be built using historical data based on the assumption that the future traffic is just a snapshot of the past. Unfortunately, such assumption is not valid in many real-world cases. Therefore, real-time data inputs are often needed to enhance the prediction quality. There are various types of real-time data such as travel time observations from previous departure time intervals, current traffic flow data from upstream road segments, etc. Travel time observations from previous departure time intervals can be relatively easy to get as long as each trip is logged once it is completed. However, if trip data is temporally very sparse (as shown in our case), such observations can be useless as the traffic behaviour can dramatically change in between. Many existing studies use traffic flow data (e.g., flow rate, flow speed) from upstream road segments. Unfortunately, such data collected from roadside sensors (e.g., dual loop detectors) are usually owned by traffic management authority. They may not be accessible to researchers and private freight transportation service providers. In addition, these types of data are more suitable for passenger vehicles which often have very different travel time profiles than freight vehicles. Many transportation service providers own a significant amount of trajectory data of their fleet, but these data are far away from being effectively utilised. Thus, there exists a mismatch between the demand for travel time analysis and the utilisation of trajectory data. This motivates us to conduct travel time prediction for freight transportation by utilising their trajectory data.

Three road segments (i.e., R1, R2 and R3) were selected for the prediction study in this thesis. Travel time predictions were first performed before departure. Two date-time indicators (i.e. departure time, day-of-week) and historical mean travel time were used as the input features. The results show that the performance of the models is unsatisfying. The features are insufficient to generalise travel times for the test trips. Although the traffic flow data is not available to us, we tend to extract some useful information from freight vehicles' trajectory data. In our study, we can estimate the mean speed every time a new trajectory report is received. The resulted series of mean speed estimates is called the mean speed sequence (MSS). MSS can be seen as the real-time speed information which reveals how fast the vehicle is in the finished part of the trip. By incorporating MSS into the model, we were able to perform travel time prediction for ongoing trips. Although MSS can help to boost prediction performance to some extent, the results are not always positive as the traffic is not always stationary over time on a different road segment. Moreover, MSS may not work well if the road segment is relatively short and has many intersections. The historical MSS (HMSS) was introduced to address the shortcoming of MSS. HMSS represents the historical profile of MSS. Instead of focusing on the first T minutes after a trip starts as that for MSS, HMSS focuses on the last T' minutes before a trip ends. To compose an HMSS of the last T' minutes, we first computed the MSS in the last T' minutes for each trip in the training data; then the HMSS can be obtained by simply averaging out all the MSSs of the last T' minutes. In this case, the speed behaviour of the vehicle in the unfinished part of the trip can be better understood. The results show that the error is slightly reduced for R1 and R2, but not for R3. The analysis on the trips shows that there are several trips in the test data whose departure times are not covered in the training data. Further experiment results show that the error can be slightly reduced by removing the trips whose departure times are not covered. This result suggests that there is a correlation between the effectiveness of HMSS and the departure time coverage ratio that larger departure time coverage is more likely to yield a better result, with

the help of HMSS features.

The quality of MSS and HMSS mainly rely on the quality of the trajectory data. The effectiveness of MSS and HMSS may decrease if the trajectory data has large noise. There are estimation errors for MSS and HMSS. If the errors are fairly distributed over a distance, the overall error does not fluctuate too much. However, sometimes we do not know if the error is fairly distributed over the distance. In this case, we recommend using the trajectory with a smaller sampling rate (e.g., 5 seconds). Although some useful speed information can be mined from vehicles' trajectory data, it is still necessary to include more relevant features such as traffic flow, task urgency, load status and weather when they are available.

7.1.2 Ensemble Algorithms for Travel Time Prediction

An ensemble algorithm assembles multiple single models. Many studies have shown that ensemble algorithms are superior to single model algorithms. In some studies, the so-called ensemble model is simply a collection of several single models built using different learning algorithms, and the prediction is just a weighted average or weighted sum. Such ensemble approaches have some drawbacks: (a) they lack theoretical proof of robustness from statistical perspective; (b) the selection of the learning algorithms may have significant impact on the performance; (c) it is difficult to determine the weights; (d) jointly tuning hyperparameters for ensemble model can be very difficult. Natural-born ensemble algorithms such as tree ensembles have been proven to be robust both theoretically and empirically. Since the model hyperparameters can be learned via data, the learning process can be fully automated. Therefore, we prefer these natural-born ensemble algorithms over postnatal ones. In this thesis, one Bayesian ensemble algorithm and three tree ensembles were selected for the travel time prediction study.

The Bayesian ensemble is Gaussian process regression (GPR). GPR has the computational complexity of $\mathcal{O}(N^3)$. Inference using GPR becomes slow if the

training data size N is more than a few thousands. To address this limitation, SGPR was adopted to build the prediction model based on the joint distribution of both training data and test data, in which the test data is conditionally independent of the training data via the inducing set. This decoupling of the dependency between the training and test data minimises the influence of those 'poor' data. As a result, the SGPR models performed very well for all cases. They outperformed the full GPR models, the SVR models, and the MLPs. The model fitting times for the SGPR models are much shorter than those for the others. The performance of the SGPR models can be affected both by the initialisation of the inducing set and the size of the inducing set. Theoretically, more inducing variables may help boost the performance, but our results show that this may not be the case if the data is sparse and noisy: larger inducing set may increase the chance of selecting more 'poor' data.

The three tree ensembles: random forests (RF), adaptive boosting regression trees (ABRT) and gradient boosting regression trees (GBRT) were investigated in this thesis. The batch Bayesian optimisation via local penalisation algorithm (BBO-LP) was applied to jointly tune the hyperparameters for the tree ensembles. With BBO-LP, the model fitting times were massively reduced when compared with the traditional cross validated grid search approach. The results show that in general, the two boosting tree ensembles, namely ABRT and GBRT, performed slightly better than RF. The comparative study shows that the three tree ensembles outperform SVR and MLPs for all cases.

To create a better understanding of how these tree ensembles can be effectively applied for travel time prediction, we further investigated the travel time prediction performance of RF and GBRT using various feature combinations. First, the investigation was conducted using pseudo travel time data that was generated using a pseudo travel time sampling algorithm (PTTS) that we developed. PTTS allows generating travel time data using different noise processes so that we can study the prediction performance under the effect of different noises. Besides, two modular based model fitting strategies were examined to see if they are beneficial to the pre-

diction. A case study was later conducted to verify the findings of the investigation using the artificially generated data. The findings are summarises below:

- The results using the artificially generated data show that historical mean travel time can be very beneficial to the prediction if the travel time distribution of a given departure time interval and day-of-week is not statistically skewed and the variances of travel times are identically distributed. However, this is hardly to be the case in the real-world cases. As is shown in the case study results, when the travel time distribution is more skewed and the variance of travel time is more complicated over time, historical mean travel time becomes less helpful.

- Travel time observations from the previous departure time intervals are beneficial to the prediction. We strongly suggest adding them as the input features, mainly when no other types of real-time information (e.g., traffic flow, speed) are available. The changes between two consecutive travel time observations from previous departure time intervals are not very beneficial to the prediction.

- Although the tree ensembles can utilise important features to build the prediction model, they can not directly handle the combinatorial effect of features which can have a negative impact on the prediction performance. Thus, it is not wise to dump a bunch of features into the model without enough care. Proper feature analysis is still needed before formally building the prediction model.

- Applying modular based model fitting strategy does not yield any practically significant performance boost.

- The tree size analysis shows that trees in the GBRT models are much smaller than those in the RF models. As a result, the model fitting time for GBRT is much shorter than for RF. These findings are consistent with those using the

artificially generated data. If the computational time cost is a big concern for the application, GBRT is recommended.

7.1.3 Implication For Other Types Prediction Problems

The thesis focused on travel time prediction problem. The selected ensemble algorithms have been proven to be capable of performing travel time prediction well even with temporally sparse and noisy data. SGPR utilises the kernel to capture the nature of complex correlation in the data. It works well in our cases mainly because our latent travel time function are approximately smooth (i.e., travel time changes smoothly from minute to minute). Thus, SGPR can be used for real-world problems (e.g., kinematic control, hydrodynamics) as long as the latent functions are assumed to be smooth. Note that SGPR may yield relatively higher model variance as it can be sensitive to the selection of inducing set. Thus, the selection of the inducing set is alway an important task in the use of SGPR. Prediction interval (PI) can be provided by SGPR because it is full Bayesian framework. PI is very useful for regression analysis. Note that a PI is estimated based on the variance of a GP which can be practically insensible. For example, in our travel time prediction problems, an estimated PI can have negative values which are not sensible in the real-world.

Tree ensemble algorithms are able to handle outliers, irrelevant features and missing values. They can work directly with mixed-type features without monotonic feature transformation. Tree ensembles do not yield very smooth decision surface. Thus, some argued that they may work better if the latent functions are less smooth. The tree ensembles have built-in feature importance analysis which is able to help us understand what are the important influencing factors on the target output. Despite this, our study results in Chapter 6 suggested that it is still necessary to perform careful feature analysis before conducting the formal model building because the tree ensembles can not directly handle combinatorial effect of features which may

have negative impacts on the model performance.

7.2 Future Works

In this last section, we present the future directions of this research work. The directions can be summarised as travel time prediction for freight transportation, hierarchical travel time prediction framework, and travel time interval prediction.

7.2.1 Travel Time Prediction for Freight Transportation

Load, task urgency (or planned finished time) may affect the speed of a freight vehicle. That is, the freight vehicle is more likely to move slower in fully loaded condition than in semi-loaded or unloaded condition; the freight vehicle is more likely to move faster if the current transportation task is urgent. Our future research work involves building a trip log system so that the travel times, the load and the task urgency information can be recorded for each trip. Then, we can use them as the input features to see if they are beneficial to the prediction.

7.2.2 Hierarchical Travel Time Prediction Framework

In Chapter 6, the noise was homogenised as a univariate Gaussian distribution and peak times were clearly demarcated from non-peak times. Thus, it is relatively easy for the tree ensembles to partition the data over different traffic conditions. In real-world cases, drawing clear boundaries between peaks and non-peaks can be more difficult. Thus, our future research work focuses on developing a hierarchical travel time prediction framework. In this framework: (1) a clustering technique (e.g. k-means) will be first applied to automatically partition training and test data; (2) a tree ensemble model is fitted to each partitioned data for the prediction; (3) the model fitting process will be performed in parallel so that the model fitting time can be significantly reduced. In addition, an algorithm will be developed and

incorporated into the framework to determine if a prediction model should be rebuilt when new data arrives.

7.2.3 Travel Time Interval Prediction

Single-value prediction is not considered a stable prediction scheme because the prediction can be higher for one day and lower for another day. This instability can be interpreted as the prediction uncertainty. In this sense, if a prediction result is presented using a prediction interval w.r.t a confidence level, the prediction becomes more stable and a user will have more confidence in utilising the upper and or lower bound for better task planning. SGPR is a Bayesian framework which can provide interval prediction. The only problem is that such interval is a Gaussian distribution where all the possible (predicted) values are symmetrically distributed around the mean (the expected one for single-value prediction). Such interval is not very meaningful from the perspective of transportation science because the distribution of travel times are often skewed. Our future research work focuses on optimising the prediction interval by taking the skewness of the travel time distribution into account. In addition, we tend to use quantile regression as the base learner for tree ensembles so that interval prediction can be provided. The quality of the interval prediction can be evaluated using metrics such as Prediction Interval Coverage Probability and Prediction Interval Normalized Averaged Width.

Bibliography

[1] "Special report 209," Washington, D.C, Tech. Rep., 1994.

[2] B. Abdulhai, H. Porwal, and W. Recker, "Short-term traffic flow prediction using neuro-genetic algorithms," *ITS Journal-Intelligent Transportation Systems Journal*, vol. 7, no. 1, pp. 3 – 41, 2002.

[3] M. S. Ahmed and A. R. Cook, "Analysis of freeway traffic time-series data by using box-jenkins techniques," *Transportation Research Record*, no. 722, 1979.

[4] R. Akcelik, "The highway capacity manual delay formula for signalized intersections," *Ite Journal-institute of Transportation Engineers*, vol. 58, no. 3, pp. 23 – 27, 1988.

[5] R. Akcelik, "Traffic signals: capacity and timing analysis," *Publication of: Australian Road Research Board*, 1981.

[6] C. Alecsandru and S. Ishak, "Hybrid model-based and memory-based traffic prediction system," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1879, pp. 59 – 70, 2004.

[7] I. Aleksander and H. Morton, *An Introduction to Neural Computing*. Chapman & Hall London, 1990.

[8] J. Anderson and M. Bell, "Travel time estimation in urban road networks," in *Proceedings of Conference on Intelligent Transportation Systems*, Nov 1997, pp. 924 – 929.

[9] M. Annunziato, I. Bertini, A. Pannicelli, and S. Pizzuti, "Evolutionary feed-forward neural networks for traffic prediction," in *Proceedings of the International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN 2003)*, 2003.

[10] C. Antoniou, M. Ben-Akiva, and H. N. Koutsopoulos, "On-line calibration of traffic prediction models," in *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, Oct 2004, pp. 82 – 87.

[11] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027 – 1035.

[12] J. Azimi, A. Fern, and X. Z. Fern, "Batch bayesian optimization via simulation matching," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 109 – 117.

[13] S. Bajwa, "An adaptive travel time prediction model based on pattern matching," in *Proceedings of the 11th World Congress on ITS*, 2004.

[14] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems 19*, 2007, pp. 153 – 160.

[15] G. H. Bham and R. F. Benekohal, "A high fidelity traffic simulation model based on cellular automata and car-following concepts," *Transportation Research Part C: Emerging Technologies*, vol. 12, no. 1, pp. 1 – 32, 2004.

[16] D. Billings and J. S. Yang, "Application of the ARIMA models to urban roadway travel time prediction - a case study," in *Proceedings of 2006 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, Oct 2006, pp. 2529 – 2534.

[17] G. E. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 4th ed. John Wiley & Sons, 2013.

[18] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees*. Taylor & Francis, 1984.

[19] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5 – 32, 2001.

[20] D. Buckley, "A semi-poisson model of traffic flow," *Transportation Science*, vol. 2, no. 2, pp. 107 – 133, 1968.

[21] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, "Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6164 – 6173, 2009.

[22] M. Cetin and G. Comert, "Short-term traffic flow prediction with regime switching models," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1965, pp. 23 – 31, 2006.

[23] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, "Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg-Marquardt algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 644 – 654, June 2012.

[24] R. E. Chandler, R. Herman, and E. W. Montroll, "Traffic dynamics: Studies in car following," *Operations Research*, vol. 6, no. 2, pp. 165 – 184, 1958.

[25] M. Chang and D. C. Gazis, "Traffic density estimation with consideration of lane-changing," *Transportation Science*, vol. 9, no. 4, pp. 308 – 320, 1975.

[26] C. Chen, A. Skabardonis, and P. Varaiya, "Travel-time reliability as a measure of service," *Transportation Research Record*, vol. 1855, no. 1855, pp. 74 – 79, 2003.

[27] H. Chen, S. Grant-Muller, L. Mussone, and F. Montgomery, "A study of hybrid neural network approaches and the effects of missing data on traffic forecasting," *Neural Computing and Applications*, vol. 10, no. 3, pp. 277 – 286, 2001.

[28] M. Chen and S. Chien, "Dynamic freeway travel-time prediction with probe vehicle data: Link based versus path based," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1768, pp. 157 – 161, 2001.

[29] M. Chen, X. Liu, J. Xia, and S. I. Chien, "A dynamic bus-arrival time prediction model based on APC data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, no. 5, pp. 364 – 376, 2004.

[30] F. Cheng, J. Yu, and H. Xiong, "Facial expression recognition in JAFFE dataset based on Gaussian process classification," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1685 – 1690, 2010.

[31] S. Chien, X. Liu, and K. Ozbay, "Predicting travel times for the south jersey real-time motorist information system," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1855, pp. 32 – 40, 2003.

[32] S. I.-J. Chien and C. M. Kuchipudi, "Dynamic travel time prediction with real-time and historic data," *Journal of Transportation Engineering*, vol. 129, no. 6, pp. 608 – 616, 2003.

[33] M. Cools, E. Moons, and G. Wets, "Investigating the variability in daily traffic counts through use of ARIMAX and SARIMAX models: assessing the effect of holidays on two site locations," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2136, pp. 57 – 66, 2009.

[34] C. F. Daganzo, "Requiem for second-order fluid approximations of traffic flow," *Transportation Research Part B: Methodological*, vol. 29, no. 4, pp. 277 – 286, 1995.

[35] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *Journal of the American Statistical Association*, vol. 74, no. 366, pp. 427 – 431, 1979.

[36] F. Dion, H. Rakha, and Y.-S. Kang, "Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections," *Transportation Research Part B: Methodological*, vol. 38, no. 2, pp. 99 – 122, 2004.

[37] M. Dougherty, "A review of neural networks applied to transport," *Transportation Research Part C: Emerging Technologies*, vol. 3, no. 4, pp. 247 – 260, 1995.

[38] M. S. Dougherty and M. R. Cobbett, "Short-term inter-urban traffic forecasts using neural networks," *International Journal of Forecasting*, vol. 13, no. 1, pp. 21 – 31, 1997.

[39] N. R. Draper and H. Smith, *Applied regression analysis*. John Wiley & Sons, 1981.

[40] H. Drucker, "Improving regressors using boosting techniques," in *Proceedings of the 14th International Conference on Machine Learning*, 1997, pp. 107 – 115.

[41] M. Elhenawy, H. Chen, and H. A. Rakha, "Dynamic travel time prediction using data clustering and genetic programming," *Transportation Research Part C: Emerging Technologies*, vol. 42, pp. 82 – 98, 2014.

[42] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179 – 211, 1990.

[43] J. Esser and M. Schreckenberg, "Microscopic simulation of urban traffic based on cellular automata," *International Journal of Modern Physics C*, vol. 8, no. 05, pp. 1025 – 1036, 1997.

[44] D. Fambro and N. Rouphail, "Generalized delay model for signalized intersections and arterial streets," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1572, pp. 112 – 121, 1997.

[45] R. Faragher, "Understanding the basis of the Kalman filter via a simple and intuitive derivation," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 128 – 132, 2012.

[46] X. Fei, C.-C. Lu, and K. Liu, "A Bayesian dynamic linear model approach for real-time short-term freeway travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1306 – 1318, 2011.

[47] P. Flach, *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[48] T. Forbes, H. Zagorski, E. Holshouser, and W. Deterline, "Measurement of driver reactions to tunnel conditions," in *Highway Research Board Proceedings*, vol. 37, 1958.

[49] M. Fosgerau, K. Hjorth, C. Brems, and D. Fukuda, "Travel time variability: Definition and valuation," *DTU Transport*, 2008.

[50] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 771 - 780, p. 1612, 1999.

[51] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning*, 1996, pp. 148 – 156.

[52] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *Annals of Statistics*, vol. 29, no. 5, pp. 1189 – 1232, 2001.

[53] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics And Data Analysis*, vol. 38, no. 4, pp. 367 – 378, 2002.

[54] A. Gal, A. Mandelbaum, F. Schnitzler, A. Senderovich, and M. Weidlich, "Traveling time prediction in scheduled transportation with journey segments," *Information Systems*, vol. 64, pp. 266 – 280, 2017.

[55] Y. Gal, M. van der Wilk, and C. E. Rasmussen, "Distributed variational inference in sparse Gaussian process regression and latent variable models," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 3257 – 3265.

[56] P. Gans, *Data Fitting in the Chemical Sciences: By the Method of Least Squares*. Wiley, 1992.

[57] D. Gazis and C. Liu, "Kalman filtering estimation of traffic counts for two network links in tandem," *Transportation Research Part B: Methodological*, vol. 37, no. 8, pp. 737 – 745, 2003.

[58] D. C. Gazis, R. Herman, and R. W. Rothery, "Nonlinear follow-the-leader models of traffic flow," *Operations Research*, vol. 9, no. 4, pp. 545 – 567, 1961.

[59] D. C. Gazis and C. H. Knapp, "On-line estimation of traffic densities from time-series of flow and speed data," *Transportation Science*, vol. 5, no. 3, pp. 283 – 301, 1971.

[60] B. Ghosh, B. Basu, and M. O'Mahony, "Time-series modelling for forecasting vehicular traffic flow in Dublin," in *Proceedings of Transportation Research Board 85th Annual Meeting*, 2005.

[61] D. Ginsbourger, J. Janusevskis, and R. Le Riche, "Dealing with asynchronicity in parallel gaussian process based global optimization," in *Proceedings of the 4th International Conference of the ERCIM WG on Computing & Statistics*, 2011.

[62] L. R. Glosten, R. Jagannathan, and D. E. Runkle, "On the relation between the expected value and the volatility of the nominal excess return on stocks," *Journal of Finance*, vol. 48, no. 5, pp. 1779 – 1801, 1993.

[63] J. Gonzalez, Z. Dai, P. Hennig, and N. Lawrence, "Batch Bayesian optimization via local penalization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, vol. 51, 09 - 11 May 2016, pp. 648 – 657.

[64] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[65] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popovi, "Style-based inverse kinematics," in *Proceedings of the 31st International Conference on Computer Graphics and Interactive Techniques*, vol. 23, no. 3, 2004, pp. 522 – 531.

[66] A. Guin, "Travel time prediction using a seasonal autoregressive integrated moving average time series model," in *Proceedings of 2006 IEEE Intelligent Transportation Systems Conference*, 2006, pp. 493 – 498.

[67] J. Guo, W. Huang, and B. M. Williams, "Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 50 – 64, 2014, special Issue on Short-term Traffic Flow Forecasting.

[68] Z. K. Gurmu and W. D. Fan, "Artificial neural network travel time prediction model for buses using only GPS data," *Journal of Public Transportation*, vol. 17, no. 2, p. 3, 2014.

[69] S. Hafstein, R. Chrobok, A. Pottmeier, M. Schreckenberg, F. C Mazur *et al.*, "A high-resolution cellular automata traffic simulation model with application in a freeway traffic information system," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, no. 5, pp. 338 – 350, 2004.

[70] B. Hamner, "Predicting travel times with context-dependent random forests by modeling local and aggregate traffic flow," in *Proceedings of 2010 IEEE International Conference on Data Mining Workshops*, Dec 2010, pp. 1357 – 1359.

[71] J. V. Hansen, J. B. McDonald, and R. D. Nelson, "Time series prediction with genetic-algorithm designed neural networks: An empirical comparison with modern statistical models," *Computational Intelligence*, vol. 15, no. 3, pp. 171 – 184, 1999.

[72] N. Hassanpour and L. Chen, "A hierarchical training and identification method using Gaussian process models for face recognition in videos," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 1, May 2015, pp. 1 – 8.

[73] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer, 2009.

[74] J. Haworth, J. Shawe-Taylor, T. Cheng, and J. Wang, "Local online kernel ridge regression for forecasting of urban travel times," *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 151 – 178, 2014.

[75] S. Haykin, *Kalman Filtering and Neural Networks*. John Wiley & Sons, 2001.

[76] B. Hellinga, P. Izadpanah, H. Takada, and L. Fu, "Decomposing travel times measured by probe-based traffic monitoring systems to individual road segments," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 6, pp. 768 – 782, 2008.

[77] J. Hensman, A. G. de G. Matthews, and Z. Ghahramani, "Scalable variational Gaussian process classification," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2015, pp. 351 – 360.

[78] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," *Uncertainty In Artificial Intelligence*, vol. 29, pp. 282 – 290, 2013.

[79] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735 – 1780, 1997.

[80] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* MIT Press, 1992.

[81] H. Hong, W. Huang, X. Xing, X. Zhou, H. Lu, K. Bian, and K. Xie, "Hybrid multi-metric k-nearest neighbor regression for traffic flow prediction," in *Proceedings of 2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sept 2015, pp. 2262 – 2267.

[82] R. G. Hoogendoorn, S. P. Hoogendoorn, K. A. Brookhuis, and W. Daamen, "Longitudinal driving behavior under adverse conditions: A close look at psycho-spacing models," *Procedia-Social and Behavioral Sciences*, vol. 20, pp. 536 – 546, 2011.

[83] S. Hoogendoorn and P. Bovy, "New estimation technique for vehicle-type-specific headway distributions," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1646, pp. 18 – 28, 1998.

[84] ——, "Gas-kinetic modeling and simulation of pedestrian flows," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1710, pp. 28 – 36, 2000.

[85] S. P. Hoogendoorn and P. H. Bovy, "State-of-the-art of vehicular traffic flow modelling," *Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 215, no. 4, pp. 283 – 303, 2001.

[86] S. Hoogendoorn and V. Knoop, *Traffic Flow Theory and Modelling.* Edward Elgar Publishing Limited, 2012.

[87] P. J. Huber, "Robust regression: asymptotics, conjectures and monte carlo," *The Annals of Statistics*, pp. 799 – 821, 1973.

[88] G. Huisken and E. C. van Berkum, "A comparative analysis of short-range travel time prediction methods," in *Proceedings of Transportation Research Board 82nd Annual Meeting*, Washington, DC, 2003.

[89] T. Idé and S. Kato, "Travel-time prediction using Gaussian process regression: A trajectory-based approach," in *Proceedings of the 2009 SIAM International Conference on Data Mining*. SIAM, 2009, pp. 1185 – 1196.

[90] S. Innamaa, "Short-term prediction of travel time using neural networks on an interurban highway," *Transportation*, vol. 32, no. 6, pp. 649 – 669, 2005.

[91] S. Ishak, P. Kotha, and C. Alecsandru, "Optimization of dynamic neural network performance for short-term traffic prediction," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1836, pp. 45 – 56, 2003.

[92] M. Jepsen, "On the speed-flow relationships in road traffic: a model of driver behaviour," in *Proceedings of the 3rd International Symposium on Highway Capacity*, no. Volume 1, 1998.

[93] Y. Jia, J. Wu, and M. Xu, "Traffic flow prediction with rainfall impact using a deep learning method," *Journal of Advanced Transportation*, vol. 2017, 2017.

[94] Z. Johnson, I. Psarros, M. Golias, and S. Mishra, "Developing freight performance measures using GPS truck data," in *Proceedings of Transportation Research Board 93rd Annual Meeting*, Washington, DC, 2014.

[95] Y. Kamarianakis, A. Kanas, and P. Prastacos, "Modeling traffic volatility dynamics in an urban network," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1923, pp. 18 – 27, 2005.

[96] Y. Kamarianakis and P. Prastacos, "Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1857, pp. 74 – 84, 2003.

[97] ——, "Space-time modeling of traffic flow," *Computers And Geosciences*, vol. 31, no. 2, pp. 119 – 133, 2005.

[98] A. Khosravi, E. Mazloumi, S. Nahavandi, D. Creighton, and J. Van Lint, "A genetic algorithm-based method for improving quality of travel time prediction intervals," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1364 – 1376, 2011.

[99] T.-H. Kim and H. White, "On more robust estimation of skewness and kurtosis," *Finance Research Letters*, vol. 1, no. 1, pp. 56 – 73, 2004.

[100] T. Kim, H. Kim, and D. J. Lovell, "Traffic flow forecasting: overcoming memoryless property in nearest neighbor non-parametric regression," in *Proceedings of 2005 IEEE Intelligent Transportation Systems*, Sept 2005, pp. 965 – 969.

[101] R. Kimber and E. M. Hollis, "Traffic queues and delays at road junctions," Tech. Rep., 1979.

[102] D. Kinga and J. B. Adam, "A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[103] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2015.

[104] L. Kisgyörgy and L. R. Rilett, "Travel time prediction by advanced neural network," *Periodica Polytechnica. Civil Engineering*, vol. 46, no. 1, p. 15, 2002.

[105] K. K. Kundé, "Calibration of mesoscopic traffic simulation models for dynamic traffic assignment," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.

[106] J. Kwon, B. Coifman, and P. Bickel, "Day-to-day travel-time trends and travel-time prediction from loop-detector data," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1717, no. 0, pp. 120 – 129, 2000.

[107] T. Lam and K. Small, "The value of time and reliability: Measurement from a value pricing experiment," *Transportation Research Part E: Logistics and Transportation Review*, vol. 37, no. 2-3, pp. 231 – 251, 2001.

[108] C. Ledoux, "An urban traffic flow model integrating neural networks," *Transportation Research Part C: Emerging Technologies*, vol. 5, no. 5, pp. 287 – 300, 1997.

[109] S. Lee and D. Fambro, "Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1678, pp. 179 – 188, 1999.

[110] G. Leshem and Y. Ritov, "Traffic flow prediction using adaboost algorithm with random forests as a weak learner," in *Proceedings of World Academy of Science, Engineering and Technology*, vol. 19, 2007, pp. 193 – 198.

[111] W. Leutzbach and R. Wiedemann, "Development and applications of traffic simulation models at the Karlsruhe Institut für Verkehrswesen," *Traffic Engineering And Control*, vol. 27, no. 5, pp. 270 – 278, 1986.

[112] M. Levin and Y.-D. Tsao, "On forecasting freeway occupancies and volumes (abridgment)," *Transportation Research Record*, no. 773, 1980.

[113] F. L. Lewis, L. Xie, and D. Popa, *Optimal and Robust Estimation: With an Introduction to Stochastic Control Theory*. CRC Press, 2007, vol. 29.

[114] X. Li and R. Bai, "Freight vehicle travel time prediction using gradient boosting regression tree," in *Proceedings of the 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec 2016, pp. 1010 – 1015.

[115] ——, "Freight vehicle travel time prediction using sparse gaussian processes regression with trajectory data," in *Proceedings of the 17th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, 2016, pp. 143 – 153.

[116] X. Li, R. Bai, P.-O. Siebers, and C. Wagner, "Modeling urban road risky driving behaviors in china with multi-agent microscopic traffic simulation," in *Proceedings of the 17th IEEE International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 1740 – 1745.

[117] M. J. Lighthill and G. B. Whitham, "On kinematic waves. II. a theory of traffic flow on long crowded roads," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 229, no. 1178, 1955, pp. 317 – 345.

[118] P. Lingras and P. Mountford, "Time delay neural networks designed using genetic algorithms for short term inter-city traffic forecasting," *Engineering of Intelligent Systems*, pp. 290 – 299, 2001.

[119] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 871 – 882, 2013.

[120] H. Liu, "Travel time prediction for urban networks," Ph.D. dissertation, Delft University of Technology, 2008.

[121] H. Liu, H. van Zuylen, H. van Lint, and M. Salomons, "Predicting urban arterial travel time with state-space neural networks and Kalman filters," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1968, pp. 99 – 108, 2006.

[122] J. Liu, W. Wang, X. Gong, X. Que, and H. Yang, "A hybrid model based on Kalman filter and neutral network for traffic prediction," in *Proceedings of 2nd IEEE International Conference on Cloud Computing and Intelligence Systems*, vol. 02, Oct 2012, pp. 533 – 536.

[123] M. Liu, R. Wang, J. Wu, and R. Kemp, "A genetic-algorithm-based neural network approach for short-term traffic flow forecasting," in *Proceedings of International Symposium on Neural Networks*. Springer, 2005, pp. 965 – 970.

[124] T. Lomax, D. Schrank, S. Turner, and R. Margiotta, "Selecting travel reliability measures," *Texas Transportation Institute Monograph (May 2003)*, 2003.

[125] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865 – 873, April 2015.

[126] J. Ma, J. Theiler, and S. Perkins, "Accurate on-line support vector regression," *Neural Computation*, vol. 15, no. 11, pp. 2683 – 2703, 2003.

[127] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[128] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187 – 197, 2015.

[129] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PLOS One*, vol. 10, no. 3, 2015.

[130] D. J. MacKay, "Probable networks and plausible predictionsa review of practical bayesian methods for supervised neural networks," *Network: Computation in Neural Systems*, vol. 6, no. 3, pp. 469 – 505, 1995.

[131] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14, 1967, pp. 281 – 297.

[132] J. Mahler, S. Krishnan, M. Laskey, S. Sen, A. Murali, B. Kehoe, S. Patil, J. Wang, M. Franklin, P. Abbeel *et al.*, "Learning accurate kinematic control of cable-driven surgical robots using data cleaning and Gaussian process regression," in *Proceedings of 2014 IEEE International Conference on Automation Science and Engineering*, 2014, pp. 532 – 539.

[133] C. D. Mark, A. W. Sadek, and D. Rizzo, "Predicting experienced travel time with neural networks: a paramics simulation study," in *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems*, 2004, pp. 906 – 911.

[134] A. D. May, *Traffic Flow Fundamentals*. Prentice-Hall, 1990.

[135] W. R. McShane and R. P. Roess, *Traffic Engineering*. Prentice-Hall, 1990.

[136] J. Mendes-Moreira, A. M. Jorge, J. F. de Sousa, and C. Soares, "Comparing state-of-the-art regression methods for long term travel time prediction," *Intelligent Data Analysis*, vol. 16, no. 3, pp. 427 – 449, 2012.

[137] ——, "Improving the accuracy of long-term travel time prediction using heterogeneous ensembles," *Neurocomputing*, vol. 150, pp. 428 – 439, 2015.

[138] J. Miller, S. i. Kim, M. Ali, and T. Menard, "Determining time to traverse road sections based on mapping discrete GPS vehicle data to continuous flows," in *Proceedings of 2010 IEEE Intelligent Vehicles Symposium*, June 2010, pp. 615 – 620.

[139] F. Moretti, S. Pizzuti, S. Panzieri, and M. Annunziato, "Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling," *Neurocomputing*, vol. 167, pp. 3 – 7, 2015.

[140] S. Moridpour, T. Anwar, M. T. Sadat, and E. Mazloumi, "A genetic algorithm-based support vector machine for bus travel time prediction," in *Proceedings of 2015 International Conference on Transportation Information and Safety (ICTIS)*, June 2015, pp. 264 – 270.

[141] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[142] J. Myung, D.-K. Kim, S.-Y. Kho, and C.-H. Park, "Travel time prediction using k nearest neighbor method with combined data from vehicle detector system and automatic toll collection system," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2256, pp. 51 – 59, 2011.

[143] J. n. Wang, X. m. Chen, and S. x. Guo, "Bus travel time prediction model with ν - support vector regression," in *Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems*, Oct 2009, pp. 1 – 6.

[144] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *Journal De Physique I*, vol. 2, no. 12, pp. 2221 – 2229, 1992.

[145] C. Nanthawichit, T. Nakatsuji, and H. Suzuki, "Application of probe-vehicle data for real-time traffic-state estimation and short-term travel-time prediction on a freeway," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1855, pp. 49 – 59, 2003.

[146] R. P. D. Nath, H.-J. Lee, N. K. Chowdhury, and J.-W. Chang, "Modified k-means clustering for travel time prediction based on historical traffic data," in *Proceedings of the International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 2010, pp. 511 – 521.

[147] G. Newell, "A theory of traffic flow in tunnels," *Theory of Traffic Flow*, pp. 193 – 206, 1961.

[148] N. L. Nihan and K. O. Holmesland, "Use of the box and jenkins time series technique in traffic forecasting," *Transportation*, vol. 9, no. 2, pp. 125 – 143, 1980.

[149] R. B. Noland and J. W. Polak, "Travel time variability: a review of theoretical and empirical issues," *Transport Reviews*, vol. 22, no. 1, pp. 39 – 54, 2002.

[150] S. Oh, Y.-J. Byon, K. Jang, and H. Yeo, "Short-term travel-time prediction on highway: A review of the data-driven approach," *Transport Reviews*, vol. 35, no. 1, pp. 4 – 32, 2015.

[151] I. Okutani and Y. J. Stephanedes, "Dynamic prediction of traffic volume through Kalman filtering theory," *Transportation Research Part B: Methodological*, vol. 18, no. 1, pp. 1 – 11, 1984.

[152] M. Papageorgiou, J.-M. Blosseville, and H. Hadj-Salem, "Modelling and real-time control of traffic flow on the southern part of Boulevard Peripherique in Paris: Part I: Modelling," *Transportation Research Part A: General*, vol. 24, no. 5, pp. 345 – 359, 1990.

[153] H. J. Payne, "Models of freeway traffic and control," *Mathematical Models of Public Systems*, 1971.

[154] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn:

Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825 – 2830, 2011.

[155] L. A. Pipes, "An operational analysis of traffic dynamics," *Journal of Applied Physics*, vol. 24, no. 3, pp. 274 – 281, 1953.

[156] I. Prigogine and F. C. Andrews, "A boltzmann-like approach for traffic flow," *Operations Research*, vol. 8, no. 6, pp. 789 – 797, 1960.

[157] I. Prigogine and R. Herman, "Kinetic theory of vehicular traffic," Tech. Rep., 1971.

[158] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81 – 106, 1986.

[159] C. A. Quiroga and D. Bullock, "Travel time studies with global positioning and geographic information systems: an integrated methodology," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 1, pp. 101 – 127, 1998.

[160] J. Quionero-candela, C. E. Rasmussen, and R. Herbrich, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, 2005.

[161] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[162] R. Reza, S. S. Pulugurtha, and V. R. Duddu, "Arima model for forecasting short-term travel time due to incidents in spatio-temporal context," in *Proceedings of Transportation Research Board 94th Annual Meeting*, no. 15-5553, 2015.

[163] J. Rice and E. Van Zwet, "A simple and effective method for predicting travel times on freeways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 3, pp. 200 – 207, 2004.

[164] M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour, "Two lane traffic simulations using cellular automata," *Physica A: Statistical Mechanics and Its Applications*, vol. 231, no. 4, pp. 534 – 550, 1996.

[165] D. Robertson, "Traffic models and optimum strategies of control: a review," in *Proceedings of the International Symposium on Traffic Control Systems*, vol. 1, 1979, pp. 262 – 288.

[166] S. Robinson and J. Polak, "Modeling urban link travel time with inductive loop detector data by using the k-nn method," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1935, pp. 47 – 56, 2005.

[167] S. Rogers and M. Girolami, *A First Course in Machine Learning*. CRC Press, 2012.

[168] L. Rui-Min, J. Jian-Gang, and T. Jin, "Performance evaluation of short-term travel time prediction model on urban arterials," in *Proceedings of the 5th International Conference on Measuring Technology and Mechatronics Automation*, Jan 2013, pp. 792 – 795.

[169] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive Modeling*, vol. 5, no. 3, p. 1, 1988.

[170] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015.

[171] M. Seeger, C. K. I. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse Gaussian process regression," in *Proceedings of the 9th*

International Workshop on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics, 2003.

[172] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148 – 175, 2016.

[173] A. Shalaby and A. Farhan, "Prediction model of bus arrival and departure times using AVL and APC data," *Journal of Public Transportation*, vol. 7, no. 1, p. 3, 2004.

[174] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[175] S. Shekhar and B. Williams, "Adaptive seasonal time series models for forecasting short-term traffic flow," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2024, pp. 116 – 125, 2008.

[176] L. Shen, "Freeway travel time estimation and prediction using dynamic neural networks," Ph.D. dissertation, Florida International University, 2008.

[177] B. L. Smith and M. J. Demetsky, "Traffic flow forecasting: comparison of modeling approaches," *Journal of Transportation Engineering*, vol. 123, no. 4, pp. 261 – 266, 1997.

[178] B. L. Smith, B. M. Williams, and R. K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 4, pp. 303 – 321, 2002.

[179] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. MIT Press, 2006, pp. 1257 – 1264.

[180] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 2951 – 2959.

[181] A. Sohr and P. Wagner, "Short term travel time prediction using floating car data based on cluster analysis," in *Proceedings of the15th World Congress on ITS 2008*, no. EPFL-CONF-155460, 2008.

[182] H. Sun, H. X. Liu, H. Xiao, R. R. He, and B. Ran, "Short term traffic forecasting using the local linear regression model," in *Proceedings of Transportation Research Board 82nd Annual Meeting*, Washington, DC, 2003.

[183] W. Suwardo, M. Napiah, and I. Kamaruddin, "Arima models for bus travel time prediction," *Journal of the Institute of Engineers Malaysia*, pp. 49 – 58, 2010.

[184] H. Symon, *Neural Networks: A Comprehensive Foundation.* Prentice-Hall, 1999.

[185] S. Takaba, T. Morita, T. Hada, T. Usami, and M. Yamaguchi, "Estimation and measurement of travel time by vehicle detectors and license plate readers," in *Proceedings of Vehicle Navigation and Information Systems Conference, 1991*, vol. 2, Oct 1991, pp. 257 – 267.

[186] C. Taylor and D. Meldrum, "Freeway traffic data prediction using neural networks," in *Proceeding of Pacific Rim TransTech Conference. 1995 Vehicle Navigation and Information Systems Conference Proceedings. 6th International VNIS. A Ride into the Future*, Jul 1995, pp. 225 – 230.

[187] S. Teply, D. I. Allingham, D. B. Richardson, and B. W. Stephenson, "Canadian capacity guide for signalized intersections," Canada, Tech. Rep., 1995.

[188] The GPy authors, "GPy: A Gaussian process framework in Python," http://github.com/SheffieldML/GPy, since 2012.

[189] The GPyOpt authors, "GPyOpt: A Bayesian Optimization framework in Python," http://github.com/SheffieldML/GPyOpt, 2016.

[190] M. K. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," in *Artificial Intelligence and Statistics*, 2009, pp. 567 – 574.

[191] T. Tsekeris and A. Stathopoulos, "Real-time traffic volatility forecasting in urban arterial networks," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1964, pp. 146 – 156, 2006.

[192] ——, "Short-term prediction of urban traffic variability: Stochastic volatility modeling approach," *Journal of Transportation Engineering*, vol. 136, no. 7, pp. 606 – 613, 2009.

[193] L. Tsirigotis, E. I. Vlahogianni, and M. G. Karlaftis, "Does information on weather affect the performance of short-term traffic forecasting models?" *International Journal of Intelligent Transportation Systems Research*, vol. 10, no. 1, pp. 1 – 10, 2012.

[194] T. Usami, K. Ikenoue, and T. Miyasako, "Travel time prediction algorithm and signal operations at critical intersections for controlling travel time," in *Proceedings of the International Conference on Road Traffic Control*, 1986.

[195] C. P. I. van Hinsbergen, T. Schreiter, F. S. Zuurbier, J. W. C. van Lint, and H. J. van Zuylen, "Fast traffic state estimation with the localized extended Kalman filter," in *Proceedings of the13th International IEEE Conference on Intelligent Transportation Systems*, Sept 2010, pp. 917 – 922.

[196] C. Van Hinsbergen and J. van Lint, "Bayesian combination of travel time prediction models," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2064, pp. 73 – 80, 2008.

[197] C. Van Hinsbergen, J. Van Lint, and H. Van Zuylen, "Bayesian training and committees of state-space neural networks for online travel time pre-

diction," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2105, pp. 118 – 126, 2009.

[198] J. Van Lint, S. Hoogendoorn, and H. Van Zuylen, "Freeway travel time prediction with state-space neural networks: modeling state-space dynamics with recurrent neural networks," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1811, pp. 30 – 39, 2002.

[199] J. Van Lint, "Reliable real-time framework for short-term freeway travel time prediction," *Journal of Transportation Engineering*, vol. 132, no. 12, pp. 921 – 932, 2006.

[200] J. van Lint, "Reliable travel time prediction for freeways," Ph.D. dissertation, Delft University of Technology, 2008.

[201] J. van Lint, S. Hoogendoorn, and H. van Zuylen, "Accurate freeway travel time prediction with state-space neural networks under missing data," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 5, pp. 347 – 369, 2005.

[202] J. van Lint and C. van Hinsbergen, "Short-term traffic and travel time prediction models," *Artificial Intelligence Applications to Critical Transportation Issues*, vol. 22, pp. 22 – 41, 2012.

[203] J. van Lint, H. J. van Zuylen, and H. Tu, "Travel time unreliability on freeways: Why measures based on variance tell only half the story," *Transportation Research Part A: Policy and Practice*, vol. 42, no. 1, pp. 258 – 277, 2008.

[204] L. Vanajakshi and L. R. Rilett, "Support vector machine technique for the short term prediction of travel time," in *Proceedings of 2007 IEEE Intelligent Vehicles Symposium*, June 2007, pp. 600 – 605.

[205] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network

with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371 – 3408, 2010.

[206] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Optimized and meta-optimized neural networks for short-term traffic flow prediction: a genetic approach," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 3, pp. 211 – 234, 2005.

[207] J. Wahle, L. Neubert, J. Esser, and M. Schreckenberg, "A cellular automaton traffic flow model for online simulation of traffic," *Parallel Computing*, vol. 27, no. 5, pp. 719 – 735, 2001.

[208] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic speed prediction and congestion source exploration: A deep learning method," in *Proceedings of 2016 IEEE 16th International Conference on Data Mining (ICDM)*, Dec 2016, pp. 499 – 508.

[209] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 283 – 298, 2008.

[210] J. Wang, I. Tsapakis, and C. Zhong, "A space-time delay neural network model for travel time prediction," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 145 – 160, 2016.

[211] J. Wang and Q. Shi, "Short-term traffic speed forecasting hybrid model based on Chaos-Wavelet analysis-support vector machine theory," *Transportation Research Part C: Emerging Technologies*, vol. 27, pp. 219 – 232, 2013.

[212] Y. Wang and M. Papageorgiou, "Real-time freeway traffic state estimation based on extended Kalman filter: a general approach," *Transportation Research Part B: Methodological*, vol. 39, no. 2, pp. 141 – 167, 2005.

[213] Z. Wang, A. V. Goodchild, and E. McCormack, "Freeway truck travel time prediction for freight planning using truck probe GPS data." *European Journal of Transport And Infrastructure Research*, vol. 16, no. 1, 2016.

[214] F. Webster, "Traffic signal settings, road research technical paper no. 39," Tech. Rep., 1958.

[215] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65 – 85, 1994.

[216] B. Williams, "Multivariate vehicular traffic flow prediction: evaluation of arimax modeling," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1776, pp. 194 – 200, 2001.

[217] B. Williams, P. Durvasula, and D. Brown, "Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1644, pp. 132 – 141, 1998.

[218] D. B. Work, O.-P. Tossavainen, S. Blandin, A. M. Bayen, T. Iwuchukwu, and K. Tracton, "An ensemble Kalman filtering approach to highway traffic estimation using GPS enabled mobile devices," in *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008, pp. 5062 – 5068.

[219] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 276 – 281, 2004.

[220] Y. Wu and H. Tan, "Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework," *arXiv preprint arXiv:1612.01022*, 2016.

[221] J. Xia, Q. Nie, W. Huang, and Z. Qian, "Reliable short-term traffic flow forecasting for urban roads: Multivariate generalized autoregressive conditional

heteroscedasticity approach," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2343, pp. 77 – 85, 2013.

[222] Y. Xie, K. Zhao, Y. Sun, and D. Chen, "Gaussian processes for short-term traffic volume forecasting," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2165, pp. 69 – 78, 2010.

[223] C. Xumei, G. Huibo, and J. Wang, "BRT vehicle travel time prediction based on SVM and Kalman filter," *Journal of Transportation Systems Engineering and Information Technology*, vol. 12, no. 4, pp. 29 – 34, 2012.

[224] J.-S. Yang, "Travel time prediction using the GPS test vehicle and Kalman filtering techniques," in *Proceedings of the 2005 American Control Conference*, vol. 3, June 2005, pp. 2128–2133.

[225] M. Yang, Y. Liu, and Z. You, "The reliability of travel time forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 162 – 171, March 2010.

[226] M. Yildirimoglu and N. Geroliminis, "Experienced travel time prediction for congested freeways," *Transportation Research Part B: Methodological*, vol. 53, pp. 45 – 63, 2013.

[227] J. You and T. J. Kim, "Development and evaluation of a hybrid travel time forecasting model," *Transportation Research Part C: Emerging Technologies*, vol. 8, no. 1, pp. 231 – 256, 2000.

[228] B. Yu, Z.-Z. Yang, K. Chen, and B. Yu, "Hybrid model for prediction of bus arrival times at next station," *Journal of Advanced Transportation*, vol. 44, no. 3, pp. 193 – 204, 2010.

[229] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," *Sensors*, vol. 17, no. 7, p. 1501, 2017.

[230] J. Yu, G. L. Chang, H. W. Ho, and Y. Liu, "Variation based online travel time prediction using clustered neural networks," in *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems*, Oct 2008, pp. 85 – 90.

[231] X. Zeng and Y. Zhang, "Development of recurrent neural network considering temporal-spatial input dynamics for freeway travel time modeling," *Computer-Aided Civil and Infrastructure Engineering*, vol. 28, no. 5, pp. 359 – 371, 2013.

[232] H. Zhang and E. Kwon, "Travel time estimation on urban arterials using loop detector data," 1997.

[233] T. Zhang, L. Hu, Z. Liu, and Y. Zhang, "Nonparametric regression for the short-term traffic flow forecasting," in *Proceedings of 2010 International Conference on Mechanic Automation and Control Engineering*, June 2010, pp. 2850 – 2853.

[234] X. Zhang and J. A. Rice, "Short-term travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 11, no. 3, pp. 187 – 210, 2003.

[235] Y. Zhang and A. Haghani, "A gradient boosting method to improve travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 58, Part B, pp. 308 – 324, 2015.

[236] Y. Zhang, A. Haghani, and R. Sun, "Stochastic volatility modeling approach that accounts for uncertainties in travel time reliability forecasting," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2442, pp. 62 – 70, 2014.

[237] Y. Zhang, A. Haghani, and X. Zeng, "Component garch models to account for seasonal patterns and uncertainties in travel-time prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 719 – 729, 2015.

[238] Y. Zhang, Y. Zhang, and A. Haghani, "A hybrid short-term traffic flow forecasting method based on spectral analysis and statistical volatility model," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 65 – 78, 2014.

[239] W. Zhao and A. V. Goodchild, "Truck travel time reliability and prediction in a port drayage network," *Maritime Economics And Logistics*, vol. 13, no. 4, pp. 387 – 418, 2011.

[240] W. Zheng, D.-H. Lee, and Q. Shi, "Short-term freeway traffic flow prediction: Bayesian combined neural network approach," *Journal of Transportation Engineering*, vol. 132, no. 2, pp. 114 – 121, 2006.

[241] M. Zhong, S. Sharma, and P. Lingras, "Refining genetically designed models for improved traffic prediction on rural roads," *Transportation Planning and Technology*, vol. 28, no. 3, pp. 213 – 236, 2005.

[242] N. Zou, J. Wang, and G. L. Chang, "A reliable hybrid prediction model for real-time travel time prediction with widely spaced detectors," in *Proceedings of the11th International IEEE Conference on Intelligent Transportation Systems*, Oct 2008, pp. 91 – 96.