AUTOMATED TRAFFIC TIME SERIES PREDICTION

Bin Sun

Blekinge Institute of Technology
Doctoral Dissertation Series No. 2018:10

Department of Creative Technologies

Automated Traffic Time Series Prediction

Bin Sun

Blekinge Institute of Technology Doctoral Dissertation Series
No 2018:10

Automated Traffic Time Series Prediction

Bin Sun

Doctoral Dissertation in
Computer Science

Department of Creative Technologies
Blekinge Institute of Technology
SWEDEN

Department of Creative Technologies
Publisher: Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
Printed by Exakta Group, Sweden, 2018
ISBN: 978-91-7295-360-4
ISSN:1653-2090
urn:nbn:se:bth-17210

A journey of a thousand miles begins with a single step.

– Laozi

To my dear parents and my dear wife.

Abstract

Intelligent transportation systems (ITSs) are becoming more and more effective. Robust and accurate short-term traffic prediction plays a key role in modern ITS and demands continuous improvement. Benefiting from better data collection and storage strategies, a huge amount of traffic data is archived which can be used for this purpose especially by using machine learning.

For the data preprocessing stage, despite the amount of data available, missing data records and their messy labels are two problems that prevent many prediction algorithms in ITS from working effectively and smoothly. For the prediction stage, though there are many prediction algorithms, higher accuracy and more automated procedures are needed.

Considering both preprocessing and prediction studies, one widely used algorithm is *k*-nearest neighbours (*k*NN) which has shown high accuracy and efficiency. However, the general *k*NN is designed for matrix instead of time series which lacks the use of time series characteristics. Choosing the right parameter values for *k*NN is problematic due to dynamic traffic characteristics. This thesis analyses *k*NN based algorithms and improves the prediction accuracy with better parameter handling using time series characteristics.

Specifically, for the data preprocessing stage, this work introduces gap-sensitive windowed *k*NN (GSW-*k*NN) imputation. Besides, a Mahalanobis distance-based algorithm is improved to support correcting and complementing label information. Later, several automated and dynamic procedures are proposed and different strategies for making use of data and parameters are also compared.

Two real-world datasets are used to conduct experiments in different papers. The results show that GSW-*k*NN imputation is 34% on average more accurate than benchmarking methods, and it is still robust even if the missing ratio increases to 90%. The Mahalanobis distance-based models efficiently correct and comple-

ment label information which is then used to fairly compare performance of algorithms. The proposed dynamic procedure (DP) performs better than manually adjusted *k*NN and other benchmarking methods in terms of accuracy on average. What is better, weighted parameter tuples (WPT) gives more accurate results than any human tuned parameters which cannot be achieved manually in practice. The experiments indicate that the relations among parameters are compound and the flow-aware strategy performs better than the time-aware one. Thus, it is suggested to consider all parameter strategies simultaneously as ensemble strategies especially by including window in flow-aware strategies.

In summary, this thesis improves the accuracy and automation level of short-term traffic prediction with proposed high-speed algorithms.

Acknowledgments

First of all, I would like to deeply and sincerely thank my supervisors and examiner, Professor Wei Cheng, Professor Guohua Bai and Dr. Prashant Goswami for their endless support. It would be not possible to finish this work without their guidance and help. I would also like to express many thanks to Professor Siamak Khatibi and Professor Kai Petersen who have done much more work beyond being external reviewers to support my PhD journey.

It is my pleasure to have many colleagues from the Department of Creative Technologies and other departments for their kind support and help, particularly, Veronica Sundstedt and Maria Lillqvist for work procedure and work environment support, Stefan Petersson and Francisco Lopez Luro for practical experiments advice.

I would also like to thank Professor Bengt Aspvall, Professor Niklas Lavesson, Professor Claes Wohlin among others for their great courses and face-to-face talks that helped me to develop related competence which is needed to conduct the experiments and academic writing.

It is hard to say bye to my dear friends who have supported me both emotionally and academically, especially Wei Wen, Cong Peng, Yuchi Kang, Yang Guo, Florian Westphal, Siva Dasari, Shahrooz Abghari, Sai-Datta Vishnubhotla, Sai Josyula, Anton Borg, Markus Wejletorp.

Finally, I would like to thank the collaborators in traffic management centres in Yunnan and Stockholm for their support and assistance with data collection and domain expertise.

Preface

This compilation thesis consists of six papers that have been peer reviewed and published in IEEE/ACM proceedings or SCI journals. The following publications are included and ordered according to publication relationships:

A. Bin Sun, Liyao Ma, Wei Cheng, Wei Wen, Prashant Goswami, Guohua Bai. "An Improved *k*-Nearest Neighbours Method for Traffic Time Series Imputation." Chinese Automation Congress (CAC). IEEE: October 2017.

B. Bin Sun, Wei Cheng, Guohua Bai, Prashant Goswami. "Correcting and Complementing Freeway Traffic Accident Data Using Mahalanobis Distance Based Outlier Detection." Technical Gazette (ISSN: 1330-3651). 24(5), pp.1597–1607, October 2017.

C. Bin Sun, Wei Cheng, Liyao Ma, Prashant Goswami. "Anomaly-Aware Traffic Prediction Based on Automated Conditional Information Fusion." International Conference on Information Fusion (FUSION). IEEE: July 2018.

D. Bin Sun, Wei Cheng, Prashant Goswami, Guohua Bai. "Short-Term Traffic Forecasting Using Self-Adjusting *k*-Nearest Neighbours." IET Intelligent Transport Systems (ISSN: 1751-956X). 12(1), pp.41–48, February 2018.

E. Bin Sun, Wei Cheng, Prashant Goswami, Guohua Bai. "Flow-Aware WPT *k*-Nearest Neighbours Regression for Short-Term Traffic Prediction." Symposium on Computers and Communication (ISCC). IEEE: July 2017.

F. Bin Sun, Wei Cheng, Prashant Goswami, Guohua Bai. "An Overview of Parameter and Data Strategies for *k*-Nearest Neighbours Based Short-Term Traffic Prediction." ACM International Conference Proceeding Series ICITT/ICSET. ACM: October 2017.

Authorship

For all publications, the thesis author was the main driver in designing the algorithm, setting up the experiments, writing the paper and analyzing the data.

Publication Relationships

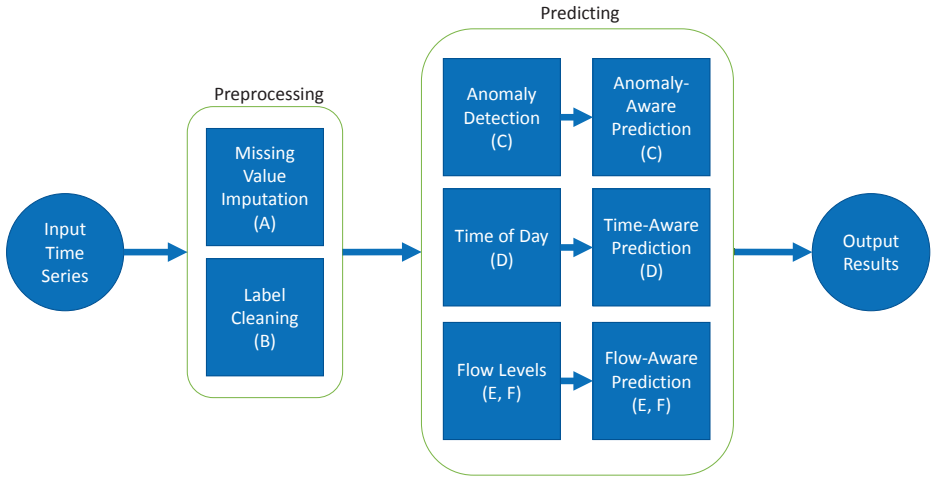

Figure 1: Publication relationships. The related papers are noted as A to F.

The thesis is divided into two parts. The first two papers improve data cleaning for traffic flow and event information, then the following papers investigate how to improve the prediction.

Firstly, missing data need to be imputated before using. Most algorithms cannot conduct analysis or prediction on time series with missing values. Imputing data makes it possible to use different algorithms and improve the final analysis or prediction.

Secondly, incidents are influencing the traffic as anomalies. To investigate the influence of those data, we need to have labelled data to conduct analysis. A multi-metric time series analysis and decision support system is built to ease the labelling process.

Third, the ground truth labelled dataset is compared with different conditional prediction methods based on anomaly detection. Thus, we can know how much useful the ground truth labels or the detection methods are.

Later, different ways to improve prediction accuracy are considered. Ensemble method can improve automation level at the same time. Different ensemble strategies are used to improve prediction and then compared.

Each paper's method can be used individually with existing systems to improve predictions. All methods can also be used together as a complete methodology for traffic time series analysis and prediction.

Related Papers

The following studies are related, but not included in the thesis:

a. Bing Li, Wei Cheng, Yiming Bie, Bin Sun. "Capacity of Advance Right-Turn Motorized Vehicles at Signalized Intersections for Mixed

Traffic Conditions." *Mathematical Problems in Engineering (ISSN: 1024-123X).* [Submitted].

b. Liyao Ma, Bin Sun, Chunyan Han. "Learning Decision Forest from Evidential Data: the Random Training Set Sampling Approach." *International Conference on Systems and Informatics (ICSAI).* IEEE: November 2017.

c. Liyao Ma, Bin Sun, Ziyi Li. "Bagging Likelihood-Based Belief Decision Trees." *International Conference on Information Fusion (FUSION).* IEEE: July 2017.

d. Liyao Ma, Bin Sun, Chunyan Han. "Training Instance Random Sampling Based Evidential Classification Forest Algorithms." *International Conference on Information Fusion (FUSION).* IEEE: July 2018.

e. Bin Sun. "How to Learn Research Ethics Regarding External Validity." *Blekinge Institute of Technology.* Report, June 2017.

Contents

List of Figures

1

Introduction

The efficient control of traffic flow on motorways or freeways can bring about shorter travel time, fewer pollutant emissions, and increased road safety [1]. Reliable and accurate short-term traffic prediction is fundamental for modern intelligent transportation systems (ITS) to achieve this target [2, 3]. ITS are becoming more and more effective, benefiting from big data generated by modern sensors and devices [2]. Efficient traffic management and accident detection rely on reliable and accurate short-term traffic forecasting [2]. As a complex task, it has been studied in the past few decades using different methods [4]. Different domains proposed different solutions, including traffic engineering methods, classical mathematical methods, modern machine learning methods. Machine learning based prediction methods rely on such big data, for example, frequently used prediction methods, including seasonal auto-regressive integrated moving average (SARIMA) [5], neural network [6,7,8], support vector regression (SVR) [9], k-nearest neighbour (kNN) regression [10] among others. Therefore, a vast number of traffic monitoring devices have been installed to collect traffic data. As a consequence, a huge amount of traffic data has been archived, sometimes together with related information such as accident records [2].

While the amount of data captured by various devices is getting larger, missing data and messy labels are causing problems. Many methods cannot make full use of dataset in case of missing values [11]. The missing ratio of traffic data is usually 5% to 25%, sometimes more than 90% [11,12,13]. For supervised learning or evaluation, we need to know related traffic data given an accident record, i.e. labelled data is required. Nevertheless, data labels are neither accurate nor complete. Further, the calibration or tuning of parameters of machine learning methods may discourage traffic management centres from using the algorithm [14]. Thus, it is necessary to impute, correct and complement data and related labels before prediction. Besides, the predictions should be further automated to be suitable for dynamic traffic.

This thesis first imputes and corrects data and labels, then improves the prediction accuracy and automation levels using ensemble methods. The predictions are considering and comparing different schemas such as time awareness, flow-rate awareness, anomaly-condition awareness.

1.1 Aim and Scope

The aim of this thesis is to improve the automation level and accuracy of short-term traffic prediction by applying statistical and machine learning methods to real-world data. To address this aim, we focus on the following objectives:

1. Generate ground truth labels for later analysis and comparison of imputation and prediction methods.

2. Develop a more accurate and automated missing data imputation method.

3. Develop more accurate and automated prediction methods considering different aspects of traffic time series.

1.2 Outline

The chapters in kappa are organized as follows. Chapter 2 presents the background, as well as terminology and related work. Chapter 3 describes the approach of the study, as well as research questions, methodology and validity threats. Chapter 4 presents the findings and conclusions of the thesis, as well as contributions, discussion and ideas for future work.

For the logic thread, the publications are presented in Chapter A-F and can be divided into two parts. Firstly, publication in Chapter A (Paper A) to Chapter B (Paper B) investigates methodology that acts as a pre-processing which cleans the data for the work in the second part of this thesis. Secondly, publication in Chapter C (Paper C) to Chapter F (Paper F) use the data cleaned in Part 1 to develop and evaluate new short-term traffic prediction methods according to the aims of this thesis.

2

Background and Related Work

Short-term traffic prediction has been a research topic for four decades since 1979 [15, 16]. It was first suggested as the core part of computer-aided traffic surveillance and control systems, i.e. intelligent transportation systems. The prediction time ranges from several seconds to several hours for future traffic [16]. The research regarding short-term traffic prediction can be categorized hierarchically into three layers [17]. On the top layer, one prediction research can target different road users and different road types. One aspect is regarding management officers vs. travellers. Another aspect is to distinguish among freeways, highways and urban roads. The middle layer concerns about data characteristics in general, for example, which parameters to predict and how long further to predict. The bottom layer covers research questions about how to model design parameters on the above two layers in detail, such as methods, data types and data quality.

This thesis focuses on freeways and the research can be generalized to different countries more easily. Three main parameters in traffic include flow rate (volume), speed and density. On one hand, flow rate and density are more related to safety. On another hand, flow rate and speed are easier to measure. When traffic states change, flow speed is changing

less than flow rate while both have high noise, i.e. the speed has a lower signal-to-noise ratio, hence low-quality data. Thus, this work considers more flow rate than flow speed.

Short-term traffic prediction methods can be divided into two categories: parametric (classical statistical) and non-parametric (data-driven approaches) [4, 16, 18]. Parametric methods are based on either traffic models like Papageorgiou's model [19] or mathematical models such as linear regression and SARIMA [5]. Non-parametric methods are modern machine learning methods, such as support vector machines, neural networksand k-Nearest neighbours. This categorization method is used through the thesis. Alternatively, some other research proposes different ways to categorize methods [4]. For example, "parametric methods" are sometimes reserved for pure traffic engineering model based methods while the mathematical ones are moved to non-parametric category [20, 21]. Sometimes, neural network methods are considered separately within their own category [22].

Machine learning based methods perform similar or better under stable traffic [4] and better than parametric ones under unstable traffic [16]. The reason is that machine learning methods are more robust and adaptive benefiting from wider general assumptions [23]. Also, they can "learn" more from bigger and heterogeneous data [4, 16].

However, the weaknesses of machine learning methods also come from big data. One weakness is that the data are not cleaned and contain missing values and messy labels. Another weakness is that many machine learning methods need parameter tuning according to dataset characteristics. They are either preventing the methods to work, or preventing them to be evaluated.

2.1 Concepts and Terminology

This section provides some basic definitions and terminologies which are used through the thesis.

Traffic Engineering. Traffic engineering studies streets and highways, and their usage by vehicles [24]. They are one key part of the transportation system which helps the movement of people and goods.

Traffic Prediction. Traffic prediction, or traffic forecasting, is the attempt to estimate the number of vehicles or people, or their speed, density, travel time related to a specific transportation facility or physical point in the future. Short-term prediction considers the future in several seconds to several hours [16,25]. Medium-term refers hours to days while long-term prediction implies the time further than one day, even beyond several weeks [25,26]. There are overlaps among different definitions of prediction time.

Machine Learning. The field of machine learning is about developing algorithms that can automatically discover and describe patterns in history data. The patterns can be used to predict future data or provide other useful information [27]. The design of algorithms is trying to find more accurate patterns when the algorithm experience more competent data [28]. Two main types of machine learning are supervised learning and unsupervised learning and a less common one is reinforcement learning [27]. *Supervised Learning* algorithms take inputs and their corresponding outputs (targets) at the same time and learn the mapping from inputs to outputs [27]. It includes both classification and regression algorithms [29]. As the purpose is to predict unknown outputs, supervised learning is also known as predictive learning [27]. *Unsupervised Learning* algorithms use only inputs to find patterns, such as groups of similar input records, the distribution of input space and representation with fewer dimensions [29]. As the purpose is to describe the known inputs, unsupervised learning is also known as descriptive learning or knowledge discovery [27].

2.2 Related Work

There are two categories of methods for short-term traffic forecasting, i.e. parametric and non-parametric methods [4, 16, 18]. During early

stages, traffic engineering based parametric models contain parameters that should be calibrated according to manually selected fixed "representative data" [30]. This problem is partially solved by introducing mathematical time series analysis. A classical mathematical parametric method is SARIMA [31] which consists of the following four parts. SARIMA originates from moving average (MA), usually weighted average of several recent white noise values given the expectation of 0. An autoregressive (AR) model is based on interdependent observations of stationary time series. The non-stationary problem of a time series can be solved by adding an integrated (I) part in the model so the non-stationary component is removed by differencing. Finally, daily and weekly periods form the important seasonality (S) part in SARIMA. Without seasonality, ARIMA itself usually cannot meet performance expectations [11, 32, 33]. One advantage of using SARIMA is the easiness of understanding it as the parameters have clear physical meanings. SARIMA sometimes performs very well [18], especially after being enhanced [34] or when available datasets are small [5, 35], though not always [36]. It is used as a benchmark method in this thesis.

Compared to parametric methods, non-parametric ones require less prior knowledge of the data and are now widely used. Tree-based algorithms are typical methods in this category. Classification and regression tree (CART) is a classical tree-based algorithm proposed by Breiman and others in 1984 [37]. It tries all possible splits to get the best split point. The best split gives the minimum node impurity (or maximum information gain) where least square criteria can be used for regression. This procedure will not stop until the tree is complete or it meets stopping criteria. Cross-validation or other methods can be used to back prune the tree to be optimal [38]. A big advantage of tree-based methods is their high speed.

Many ensemble algorithms are using trees as their base estimators to make use of the efficiency. One important ensemble method is bagging (bootstrap aggregating) which is also proposed by Breiman [39, 40]. The basic idea is to conduct sampling from the original dataset with replace-

ment to produce several new different datasets and get different trained models. The final output is the averaged result of all those models. If the input features are also randomly selected while sampling instances, then the algorithm becomes a special bagging algorithm, i.e. random forest [41,42].

Another commonly used ensemble method is boosting (similar to arcing). Kearns proposed the idea of boosting weak learners to form a strong one [43]. This idea is confirmed by Schapire [44]. Instead of having parallel models like in bagging, boosted models are connected sequentially. During training, previous misestimated input data will have a higher chance of being selected for training later models [41]. Both bagging and boosting reduce mainly the variance of unstable learning algorithms, such as trees and neural networks [45], and sometimes also reduce bias [46].

Though the usage is rare in traffic prediction, the CART algorithm has been used to predict bus dwell time [38] while the normal bagging has been used for travel time prediction [47]. They have been used more for classification and prediction of driver behaviour, lane changing and accident possibilities or risks. Compared to those two methods, random forest is an outstanding bagging algorithm which has been used in many studies to predict traffic information [48,49,50]. Recently, as a state-of-the-art boosting method, extreme gradient boosting (XGB) has shown outstanding efficiency [51]. The performance is shown to be comparable with or better than random forest [52,53]. Thus, XGB is also compared in this thesis.

Some other non-parametric methods have also been used beside tree-based algorithms. Support vector regression (SVR) is proposed by Vapnik and Drucker among others in 1996 [54] and become popular in traffic engineering about ten years later. It is good at solving non-linear problems given appropriate kernels [55]. By using different kernels, a good trade-off between accuracy and efficiency can be selected [34]. Much other work focuses on using neural networks, though mainly by doing

classification before regression to accomplish the later regression methods [7,8,56,57].

Within non-parametric methods, pattern-searching algorithms form a big subcategory and *k*NN is one of them [18,58,59,60]. The first researchers mentioning *k*NN are Cover and Hart [61]. It is then used as an alternative method for traffic prediction since 1990s [62,63]. The idea is to find history data with similar patterns of current data. *k*NN has been improved from different aspects, such as time aspect [64], spatial aspect [60], data source aspect [65,66], complicated mathematical model aspect [67], among others.

This work uses *k*NN because of the substantial increase in data availability [4], the flexibility of *k*NN for solving non-linear problems and easiness of understanding and implementation [55]. Three parameters of *k*NN are the number of nearest neighbours, search step length (also known as lag) and window size (also known as constraint) [58]. Many studies tried to tune the number of nearest neighbours [68,69,70] and some work also tried to tune search step length [71,72] while few researchers considered shifting neighbours on timescale, i.e., using windows. Previous work only studies some of the parameters and the value assignment for those three parameters at the same time is still a problem. Besides, for different traffic data, it is necessary to choose suitable parameter values differing from case-to-case and time-to-time. A *k*NN algorithm with fully automatic parameter self-adjustment and higher accuracy is needed. Also, the thesis compares data strategies and other settings.

We found a problem while developing traffic prediction algorithms. To be able to fairly compare the performance of algorithms under different traffic situation, data labels are needed, for example, if the data are from accidents. However, the traffic labels are often messy or missing and should be corrected or complemented. There are two ways to detect accidents or other anomalies [73]. The first way is to "recognize" accidents if the new query traffic is similar to previous accident traffic.

This can be done by conventional methods such as McMaster [74] as well as novel machine learning based classification methods [75]. The conventional methods usually require physical location characteristics like the shape of roads or multi-device data as part of the input, and machine learning based classification requires labelled data. The second way discovers observations that are significantly different from typical values. This procedure is called outlier detection [76]. There are two types of outliers, global outlier and local outlier [77]. Global outliers are considered as outliers regardless of the concept, whereas, local outliers are concept-related. For example, 40°temperature is normal in India, but outlier value in Sweden. When applying outlier detection for labelling, we found much research assumes outliers as global [78], but local outlier detection can be more suitable to solve traffic data related problems. Further, existing research preferred to use single metric and its threshold to detect outliers such as flow rate [73,79], speed [80] or density [81,82,83]. As there are fundamental differences among characteristics of different roads, multi-metric detection may be more suitable depending on the roads [84, 85]. Additionally, on the timescale, though some work has been done with regard to transit fundamental diagram [85], none considered differential time-varied fundamental diagram.

Actually, labelled data can also be used for conditional prediction which improves accuracy, especially during extreme events [86]. For example, prediction-after-classification approaches [87] have been used, which considers one whole day as one data point without dynamic time series characteristics. Another work found four different pattern changes, awareness of related patterns improves prediction accuracy [88]. A work [89] considers typical vs. atypical conditions with manual algorithm assignment and fixed parameters. However, we have not seen conditional ensemble methods with automated parameter tuning for short-term traffic prediction considering incidents or anomalies, and conditional prediction should rely on more automated anomaly detection, not manual work. One type of methods for more automated anomaly detection is based on distribution, distance or density. For example, dynamic Poisson distribution based detection [90], Chi-square test [91], and

modern machine learning methods such as local outlier factor [92, 93], one class support vector machine [94] and iForest [95]. However, transforming time series into multidimensional space often loses time characteristics. Another type is to use regression methods [96] to get residuals and detect outliers according to a fixed or dynamic threshold. One issue is that the dynamic/local traffic fluctuation is not considered in existing work. Usually, higher traffic comes with higher fluctuation and deviation. Multi-seasonality characteristic is also problematic.

One more problem which occurs frequently in our data processing actions and causes many small issues, i.e., missing data. Most algorithms rely on continuous data to work so the missing data should be imputed. Data imputation methods can be divided into several categories: prediction based, interpolation based and statistical/machine learning based [11, 97]. Prediction based methods, such as ARIMA and neural network, only make use of continuous chunks in the time series [11]. They are subject to missing data problem themselves. A typical interpolation based method is regression spline, which is a piecewise polynomial function that tries to approximate the unknown function [98, 99]. In the machine learning category, *k*-Nearest neighbours (*k*NN) [12, 100] and principal component analysis (PCA) based methods [11, 101] have shown great performance [11, 102] and efficiency [103]. Besides, there are some ad-hoc methods, such as mean, median and last observation carried forward (LOF) [104]. Among those methods, *k*NN is accurate and efficient [11, 103]. One issue is that the general *k*NN is not considering gap sizes of continuous missing data though the gap sizes matter, which has been shown in prediction-related work [105, 106].

The above-mentioned problems and issues are considered important to the improvement of the short-term traffic prediction accuracy and automation level so that they are investigated in this thesis.

3

Approach

This thesis is contributing to the subjects of computer science to the domain of transportation engineering. More specifically, the thesis overlaps the areas of data science and traffic analysis by improving, developing and applying traditional statistical methods and machine learning methods to short-term traffic time series prediction.

The main question of this thesis is: *How can the traffic data be processed in a more automated and accurate manner?* This question is addressed via several smaller research questions (RQs) in Section 3.1.

Later, this chapter presents the research methods used in the thesis to answer those specific sub-questions. Datasets and validity threats are also detailed.

3.1 Research Questions

Part I

RQ I. *How to impute time series missing values considering data missing*

gaps and other time series characteristics?
It is necessary to impute missing values so that general detection and prediction algorithms that require continuous time series without missing values can perform more accurately and automatically. However, time series characteristics can be further investigated. Thus, we want to investigate how to impute the time series in a better way. This is addressed in Paper A (Chapter A).

RQ II. *How to generate ground truth labels accurately and actively using existing but messy event data?*
Ground truth labels can be used to improve accuracy and to evaluate prediction results conditionally. However, existing event records are usually messy. Thus, we want to investigate how to use the dirty and messy event records together with traffic flow history to get accurate labels actively so that to ease the analysis later. This is addressed in Paper B (Chapter B).

Part II

RQ III. *How kNN can be automated even for real-time requirements considering different traffic conditions, flow rates and parameter configurations?*
One important aspect of algorithms is automation. Many algorithms are only suitable for dedicated local situations. We want to improve automation level of traffic prediction in general using ensemble methods to make them more dynamic. This is addressed in the last four papers.

RQ IV. *How can kNN be improved w.r.t. accuracy by considering time series characteristics, parameter configurations using ensemble methods?*
Ensemble methods have shown promising results for improving algorithm accuracy. However, some time series characteristics have not been fully investigated, especially traffic flow

characteristics. We want to improve ensemble methods considering more time series characteristics. This is addressed in the last four papers.

3.2 Research Methodology

We investigate the data-flow from raw data to final traffic prediction. One fundamental characteristic of traffic time series is that the flow values are changing continuously and smoothly, when being compared with categorical data. Thus, a part or period of a time series can be *shifted* to a nearby part or period. This introduces *window* parameter which is the maximum limit of shifts. It dramatically improves the performance of analysis and prediction algorithms, especially ensemble methods. The ability to shift also makes it possible to have a gap-sensitive imputation method. We also enhance the dynamic capabilities using ensemble methods. The proposed methods are automated according to current traffic flow characteristics. One exception of the automation is the data labelling part which is only used for benchmarking and analysis purpose. The algorithms we propose consider traffic flow from several different aspects, including anomaly-aware, time-aware, and flow-aware. The specific methodologies and their evaluations are described in detail in the included papers. The methods are designed in the way that they can be used together or individually to improve traffic time series analysis and prediction. Quantitative methods, i.e. experiments, are applied in the research approach.

3.3 Experiments

As the work is targeting prediction, the design of experiments starts from prediction. However, it is hard to do so as the data labels are messy when being provided by the traffic centre. Thus, a process is designed and a system is implemented to label the data, as described in Paper B. Many prediction algorithms cannot handle missing values, so simple kNN is used to fill the missing values. Later, the data are

used in prediction papers (Paper D, E, F) The influence of this simple imputation is concerned as well as traffic incidents. Only the time-series ranges without (or with little, less than 10%) missing values or incidents are used during evaluation.

We can see that the above design can be improved. First, the imputation is too simple to be considered as accurate. Second, the influence of incidents is excluded which might be not possible of no detection and no lables are provided, while sometimes it is even needed specially for incident flow prediction. Thus, the imputation is improved in Paper A and the influence of incidents is investigated in Paper C. Additionally, the 2nd paper for prediction (Paper E) starts to one important time-series parameter, window size (the limit of shits), which is not considered in the 1st prediction paper (Paper D). Some details and comparision of those two prediction papers leads to the 3rd prediction paper (Paper F).

The algorithms are implemented in CUDA, R and Python according to situations. Some parallel computing frameworks are used to accelerate experiments, such as CUDA-GPU and Spark. The GSW imputation algorithm (Paper A) is implemented using Spark in Python. The accident information cleaning model (Paper B) is implemented using Shiny in R and then used as a decision support system to label the existing traffic flow data. The proposed DP (Paper D) and WPT (Paper E) algorithms are implemented using CUDA in C/C++ and run on NVidia GPUs. The analysis of anomaly-aware conditional predictions (Paper C) is done in Python and the comparison of different strategies (Paper F) is using R. All the core codes used for this thesis are available on GitHub which can be found by searching the related paper titles. Part of the experiemnt data are available, while the non-available part is described statistically and the data structure is provided.

The main evaluation metric is mean absolute error (MAE) for predictions and root mean square error (RMSE) is for imputations. The reason is that they are frequently used for predictions and imputations respectively and as it is easier for readers to compare with literature results.

Though mean absolute percentage error (MAPE) is also frequently used for predictions, it is not suitable for our data due to the fact that traffic flow values can be zero especially during night.

3.4 Datasets

Two real-world datasets are used. One dataset is from Bluedon surveillance camera devices on Kunshi Road and the data are extracted using virtual-loop based vehicle detections. One device sends a monitored flow record at five-minute intervals. Each record contains some traffic statistics such as flow rate and average speed. This road carries undersaturated flow except in holidays noons. Another dataset, PeMS, is from the Bay Area, USA and the details can be found in [107].

4

Results

4.1 Answers to Research Questions

Paper A (Chapter A) is intended to answer RQ I. This paper investigates two time series characteristics that can be used to improve the imputation accuracy. One characteristic is the gaps between missing values and existing values. Another characteristic is the windows for shifting time series. The results of Paper A (Chapter A) showed that the method GSW-*k*NN can provide more accurate imputation for time series by considering time series characteristics.

Paper B (Chapter B) is intended to answer RQ II. This paper investigates multi-variate based Mahalanobis distance that can improve events detection in traffic flow. Together with dirty and messy event records from heterogeneous sources, we developed a web user interface as a decision support system to correct and complement event information. The results of Paper B (Chapter B) showed that the proposed model and application provides more event information to support labelling traffic time series data.

The other papers are intended to answer RQ III and RQ IV. The

*k*NN method is improved from both automation and accuracy aspects while considering time-domain relations, flow conditions (incident vs. non-incident), flow rate levels, and parameter relations. The results of Paper C to Paper F (Chapter C to Chapter F) show that our proposed methodology enhance *k*NN using ensemble method w.r.t. accuracy and automation level.

4.2 Contributions

This thesis contributes to two major parts of traffic data analysis, data cleaning and traffic prediction. The first part contributes to cleaning data (including imputation and labelling) for traffic time series data. The second part contributes to improving prediction accuracy (in terms of reducing the error of prediction) and automation level of traffic flow prediction. The following paragraphs address the specific contributions of each paper.

Paper A (Chapter A) proposes GSW-*k*NN for traffic time series imputation. Much work has been done to impute missing data. Among different imputation methods, *k*-nearest neighbours (*k*NN) has shown excellent accuracy and efficiency. However, the general *k*NN imputation is designed for matrix instead of time series so it lacks the usage of time series characteristics such as windows, gap-sensitive searching steps and weights. Thus, the method GSW takes gap sizes into consideration during distance measurement and uses windows to broaden the amount of neighbours. The experiment results from the public PeMS dataset shows that GSW-*k*NN performs 14% to 59% better than benchmarking methods and 34% better on average. It is also more robust even for datasets with high missing ratios. This helps to answer RQ I.

Paper B (Chapter B) describes how to correct and complement accident information. Previous research has investigated different methods for detecting traffic accidents, few of them used multi-metric data. However, no work has clearly defined how to use Mahalanobis distance [108] for detecting traffic accidents. This research proposes to use multi-

metric data instead of commonly used single metric data for traffic analysis and accident information correction and complementation. We also introduce a general method to pre-process traffic data to be suitable for multivariate M-distance based algorithm. In this process, we introduce the importance of differential distance. Then we modify the algorithm to be updatable and describe the methodology to detect accident and correct and complement accident data. Finally, based on proposed algorithm, we develop a system with illustrative and interactive user interface to visualize different outliers in time domain and help to fix accident information efficiently. This helps to answer RQ II.

Paper C (Chapter C) investigates conditional traffic prediction. Previous research lacks automated anomaly detection and conditional information fusion with ensemble methods. This works aims to improve prediction accuracy by fusing information considering different traffic conditions with ensemble methods. Apart from conditional information fusion, a day-week decomposition (DWD) method is introduced for preprocessing before anomaly detection. A *k*-nearest neighbours (*k*NN) based ensemble method is then using the detected information to conduct prediction. Real-world dataset is used to test the proposed method with stratified ten-fold cross-validation. The results show that the proposed method with incident labels improves predictions by up to 15.3% and the DWD enhanced anomaly-detection improves predictions up to 8.96%. Conditional information fusion improves ensemble prediction methods, especially for incident traffic. DWD increased detection ratio by 31% - 44%, and reduced false alarm ratio by 69% - 74%. The proposed method works well with enhanced detections and the procedure is fully automated. This helps to answer mainly RQ III and partly RQ IV.

Paper D (Chapter D) proposes an online dynamic procedure that adjusts *k*NN parameters automatically according to recent traffic information. Previous research has investigated different ways to adjust *k*NN parameters for short-term traffic forecasting. However, they usually require labelled data for training, and cannot run directly for real-time online tasks. This work successfully modifies *k*NN to DP-*k*NN with

fully automated self-adjustment for the parameters without calibration or training. The results show that DP-kNN gives 9% to 40% improvement than benchmarking methods on average when considering both holidays and workdays. In addition, the dynamic procedure is a general methodology that can be applied to similar algorithms and systems which are in need of self-adjustable parameters. This helps to answer mainly RQ III and partly RQ IV.

Paper E (Chapter E) proposes to use WPT to make kNN dynamically tuned considering dynamic flow rate levels. Previous research has investigated how to choose the right parameter values for kNN. However, they did not consider all parameters of kNN at the same time, especially under dynamic flow rate. This work proposes weighted parameter tuples (WPT) to calculate weighted average dynamically according to flow rate. WPT gives the performance that cannot be achieved using manual tuning. Besides, WPT is 3.05% better than XGB and 11.7% better than SARIMA. WPT is not only accurate but also space efficient. Only weights are saved which is one hundred kilobytes in the experiment, and one-year history traffic data uses less than one-megabyte space. Additionally, WPT is more robust when conduct multi-step (multi-interval) prediction. This helps to answer mainly RQ IV and partly RQ III.

Paper F (Chapter F) analyses parameter strategies and data strategies that are used to improve kNN prediction. Previous research has proposed many strategies to improve kNN prediction. However, we did not see studies that compares strategies from different aspects/categories. The results show that kNN prediction should consider all parameter strategies simultaneously as ensemble strategies especially by including v together with k and d in flow-aware strategies. This helps to answer RQ IV.

4.3 Conclusion

This thesis investigates how traffic time series can be analysed especially by using machine learning methods. The work uses different ap-

proaches, including traditional statistical methods and machine learning algorithms for time series analysis. The target is to improve both the accuracy and automation level of traffic flow time series analysis, including both preprocessing and prediction stages.

The data cleaning preprocessing targets two issues which are missing value imputation and data labelling. So that the later predictions and related evaluations are more automated and more accurate. The predictions are improved using ensemble kNN from three different aspects, including time awareness, flow awareness and anomaly awareness. The prediction is not only more accurate but also more automated. Higher automation levels of approaches means easier adaption and more objective prediction process, while also reducing human effort and mistakes. Consequently, the traffic prediction and management procedure is more robust and uniform across traffic management centres and authorities.

4.4 Future Work

One limitation of our kNN based algorithms is that they require much computing resource. The proposed algorithms are implemented in CUDA to take advantage of GPU acceleration to achieve real-time prediction. XGB should be firstly considered if its accuracy is acceptable. The similar issue occurs for imputation algorithms. Currently, when higher accuracy or automation level is needed, our proposed algorithms can be used. One reason that leads to heavy resource usage is that this thesis applies a bagging-like philosophy to parameter configurations. The underneath problem can be treated as a hyper-parameter optimization (hyper-opt) question. Current widely used hyper-opt techniques include Bayesian search, grid search, gradient descent, random search, evolutionary search among others. Despite this, we have found that using one single configuration of parameters cannot achieve the best performance, even it is dynamic. Nevertheless, it is worth trying them and conduct comparison especially when their automation levels have been improved recently in practice, though it is also time-consuming [109].

Considering existing research and the aforementioned layered classification at the beginning of Chapter 2, several more aspects can be further investigated. Firstly, other prediction methods such as recurrent neural network (RNN) deep learning and integrated SARIMA+Kalman filter can be considered. RNN is not used as it was heavy, from both time complexity and space complexity aspects, as well as in both training and running stages. However, most recent studies are able to reduce the memory space usage by 10 to 15 times [110] and increase the speed to be 20 times faster than real-time for some tasks that are similar to time series analysis [111]. Also, the manual tuning work of the deep architecture can be reduced [112]. Besides, the interpretability of RNN has been increased with physical and intuitive meanings, for example, by using finite state automaton [113]. SARIMA+Kalman filter can be further automated and integrated to improve ensemble method. For the *k*NN method itself, different similarity or distance measurements of neighbours can be compared. Secondly, for the type of input data, GPS data from floating cars like taxies and buses can be used. Third, for the output data, also for the travellers, travel time prediction is an important and interesting topic to consider. Fourth, different quality of data should be considered together, for example, low and high missing ratio of values, random record intervals. Some information fusion methods for inaccurate data can be used. From traffic engineering aspect, spatial and temporal data within the same or even different road networks can be fused. This has been proved to be useful and we want to combine with our methods in an automated way. Besides, we can improve the quality of labels and compare algorithms under more different traffic conditions.

Links and DOIs for included papers:

A. "An Improved k-Nearest Neighbours Method for Traffic Time Series Imputation." @IEEE, https://ieeexplore.ieee.org/document/8244105, DOI: 10.1109/CAC.2017.8244105

B. "Correcting and Complementing Freeway Traffic Accident Data Using Mahalanobis Distance Based Outlier Detection." @Technical Gazette https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=277502&lang=en, DOI: 10.17559/TV-20150616163905

C. "Anomaly-Aware Traffic Prediction Based on Automated Conditional Information Fusion." @IEEE, https://ieeexplore.ieee.org/abstract/document/8455244, DOI: 10.23919/ICIF.2018.8455244

D. "Short-Term Traffic Forecasting Using Self-Adjusting k-Nearest Neighbours." @IET, http://digital-library.theiet.org/content/journals/10.1049/iet-its.2016.0263, DOI: 10.1049/iet-its.2016.0263

E. "Flow-Aware WPT k-Nearest Neighbours Regression for Short-Term Traffic Prediction." @IEEE, https://ieeexplore.ieee.org/document/8024503, DOI: 10.1109/ISCC.2017.8024503

F. "An Overview of Parameter and Data Strategies for k-Nearest Neighbours Based Short-Term Traffic Prediction." @ACM https://dl.acm.org/citation.cfm?id=3157749, DOI: 10.1145/3157737.3157749

A

An Improved *k*-Nearest Neighbours Method for Traffic Time Series Imputation

Bin Sun, Liyao Ma, Wei Cheng, Wei Wen, Prashant Goswami, Guohua Bai. Chinese Automation Congress (CAC). IEEE: October 2017.

Abstract

Intelligent transportation systems (ITS) are becoming more and more effective, benefiting from big data. Despite this, missing data is a problem that prevents many prediction algorithms in ITS from working effectively. Much work has been done to impute those missing data. Among different imputation methods, *k*-nearest neighbours (*k*NN) has shown excellent accuracy and efficiency. However, the general *k*NN is designed for matrix instead of time series so it lacks the usage of time series characteristics such as windows and weights that are gap-sensitive. This work introduces gap-sensitive windowed *k*NN (GSW-*k*NN) imputation for time series. The results show that GSW-*k*NN is 34% more accurate than benchmarking methods, and it is still robust even if the missing ratio increases to 90%.

Index terms— Traffic Time Series, Gap-Sensitive Windowed *k*-Nearest Neighbours (GSW-*k*NN), Missing Data Imputation

A.1 Introduction

Nowadays, intelligent transportation systems (ITS) are becoming more and more effective, benefiting from big data generated by modern sensors and devices [2]. Many prediction algorithms rely on such big data, for example, frequently used prediction methods, including auto regressive integrated moving average (ARIMA) [114], neural network [6,7,8], support vector regression (SVR) [9], *k*NN regression [10] among others. These methods cannot make full use of dataset in case of missing values [11]. While the amount of data captured by various devices is getting larger, the amount of missing data increases as well. The missing ratio of traffic data is usually 5% to 25%, sometimes more than 90% [11,12,13].

There has been much work on imputation for time series data and *k*NN has produced imputations accurately and efficiently [11,103]. However, two key topics are not found in the literature. Firstly, missing values lead to gaps within time series which should be considered during neighbour distance measurement. Secondly, time series is continuous on time scale which is not considered in general *k*NN imputation. To solve the problems mentioned above, this work proposes a gap-sensitive windowed *k*NN (GSW-*k*NN) to impute missing values in traffic time series.

A.2 Background and Related Work

Data missing situation can be categorized into three types [102,104,115]. In the first type, the values are missing completely at random (MCR). The missing positions are completely unrelated with each other and are randomly located on the time scale. In the second type, the values are missing at random (MR) which means the missing positions are dependent on neighbouring samples. They are kind of clustered or continuous,

but the clusters are randomly located on the time scale. The situation is usually due to device damage or maintenance. The third type is missing values not at random (MNR). The missing clusters are not randomly located on the time scale. There can be more types when considering spatial information such as multi-sensor station situation, which is out of the scope of this work [12].

To impute missing values, data imputation methods are developed which can be divided into three categories: prediction based, interpolation based and statistical/machine learning based [11, 97]. Prediction based methods, such as ARIMA and neural network, have the problem that they only make use of continuous chunks in the time series [11]. A typical interpolation based method is regression spline, which is a piecewise polynomial function that tries to approximate the unknown function [98, 99]. In the statistical/machine learning category, *k*-Nearest neighbours (*k*NN) [12, 100] and principal component analysis (PCA) based methods [11, 101] have shown great performance [11, 102] and efficiency [103]. It has been shown that different PCA based methods perform similar with regards to accuracy [102]. Another work introduces an ad hoc category which includes mean, median and last observation carried forward (LOF) [104]. Among those methods, *k*NN is accurate and efficient [11, 103].

The general *k*NN imputation of traffic data is shown in Figure A.1. One value at time t on day D is missing. Each searchable day in the history data is considered as a neighbour. The distance between the day D and other days are calculated and the nearest k days are selected as nearest neighbours. The values (white-dashed dark green squares) at time t of the k days are averaged to impute the missing value at time t on day D.

One problem is the general *k*NN imputation is that it is developed for matrix such as microarrays in human genes, which is not considering traffic data on the time scale. An important characteristic of time series is correlation and continuous change on the time scale. A pos-

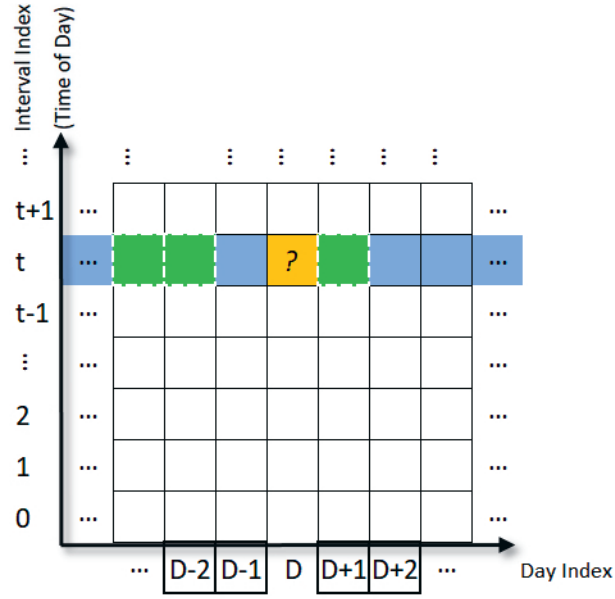

Figure A.1: General *k*NN imputation matrix. Imputation for time t on day D using other days' data. The entire data of each day as a vector is taken to calculate similarity and the most similar days are selected. Three dashed dark green squares are the selected nearest neighbours' data at t of corresponding days ($k = 3$). 4 searchable history days (no shifts) lead to 4 potential neighbours.

sible approach to make use of such a characteristic is to use window based shifting, which have been used for *k*NN based time series prediction [36,116]. Though this approach works well, it has not been used for imputation.

Another problem is that we can see that the general *k*NN is not considering the gap of missing data. Gaps occur if any values are missing. When the general *k*NN algorithm tries to impute a value, it measures its distance to neighbours with the positions of missing values ignored. However, missing values at different positions have diverse influences on the distance measurement. This gap matters, which has been shown in prediction-related work [105,106], but the general *k*NN considers all elements in the vectors equally.

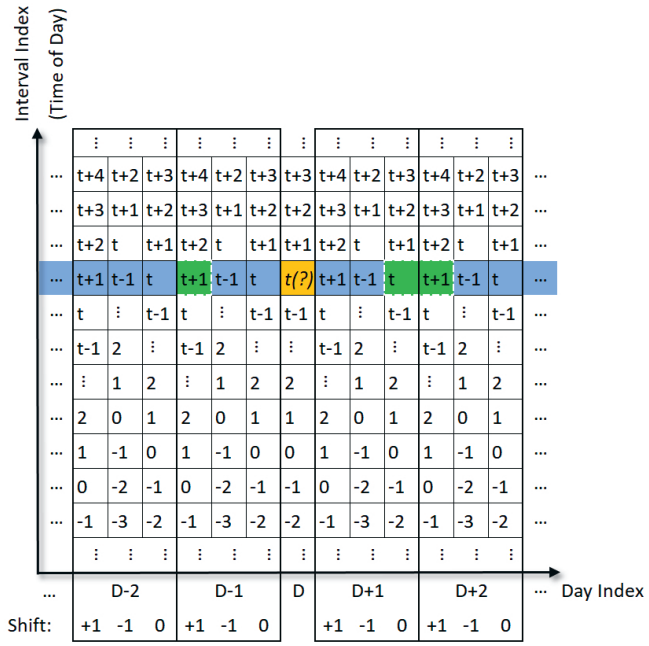

Figure A.2: Traffic prediction for time t using GSW-kNN. The position of missing data is (D, t). The shown example window is $v = 1$, thus three shifts: (-1,0,+1). Three dashed dark green squares are the selected nearest neighbours' data at t or $t \pm 1$ of corresponding days ($k = 3$). 4 searchable history days with three shifts lead to 12 potential neighbours. Besides, the instances indexes are extended into the neighbouring previous or later day when the lag d is big.

A.3 Methodology

This section proposes a gap-sensitive windowed kNN for imputation to solve the issues mentioned above. Later, the data and experimental setup are described.

A.3.1 GSW-kNN

As shown in Figure A.2, suppose the position of missing data is (D, t) which is at the time t on the day D. The search lag length is d on both

sides of (D, t), so the state vector of the missing data is:

$$\boldsymbol{r}_{D,t} = (r_{D,t-d}, \ldots, r_{D,t-1}, r_{D,t+1}, \ldots, r_{D,t+d}) \tag{A.1}$$

where $\boldsymbol{r}$ means traffic flow volume rate. As an example, one searchable neighbour vector on the previous day $(D-1)$ without window is:

$$\begin{aligned} &\boldsymbol{r}_{D-1,t} \\ &= (r_{D-1,t-d}, \ldots, r_{D-1,t-1}, r_{D-1,t+1}, \ldots, r_{D-1,t+d}) \end{aligned} \tag{A.2}$$

where $\boldsymbol{r}_{D-1,t}$ is the *matching point data*. This step is similar to the general kNN neighbours except that the state vector of missing data and searchable history neighbours $\boldsymbol{r}$ can extend to neighbouring days when d is big.

Here we add a window to the algorithm. A *window* (v) is a limitation of *shift* that can be applied to all searchable history day and its matching point. For example, if $v = 1$, the shift can be $-1, 0, 1$. If a shift 1 is applied to the neighbour vector $\boldsymbol{r}_{D-1,t}$ in (A.2), the neighbour vector becomes:

$$\begin{aligned} &\boldsymbol{r}_{D-1,t+1} \\ &= (r_{D-1,t-d+1}, \ldots, r_{D-1,t}, r_{D-1,t+2}, \ldots, r_{D-1,t+d+1}) \end{aligned} \tag{A.3}$$

Introducing window into kNN is a good way to make full use of data. Suppose the window size is v, then the shifts contained in the window are $-v, -v+1, \ldots, -1, 0, 1, \ldots, v-1, v$. Consequently, the non-zero shifts make the searchable dataset v times bigger than the original dataset. This is important due to the fact that machine learning algorithms heavily rely on the big amount of available data. Especially when there is not too much data, such as only one single station/sensor's data are available.

The distance between neighbours is measured by Euclidean distance as it is frequently used in literature [117]. Other distance metrics such as Mahalanobis distance [118] or dynamic time warping [100, 119] are

sometimes used but out of the scope of this work. Due to the fact that the search length d is dynamic, using Euclidean distance is not suitable for d-dimension vectors. Also, to avoid the curse of dimension problem [120], we treat the time domain as one dimension instead of d-dimension by averaging the distance values from all search steps. The distance between the missing data vector (A.1) and one neighbour (A.2) is defined as follows:

$$\dot{\boldsymbol{r}} \times \boldsymbol{w} = (\boldsymbol{r}_{D,t} - \boldsymbol{r}_{D,t-1}) \times \boldsymbol{w} \tag{A.4}$$

where $\dot{\boldsymbol{r}}$ is the vector of absolute values of element-wise subtraction of $\boldsymbol{r}_{D,t}$ and $\boldsymbol{r}_{D,t-1}$. The $\boldsymbol{w}$ is the vector of weights for each search lag and it is gap-sensitive and is defined as follows.

Firstly, the initial scores are calculated:

$$\begin{aligned} \boldsymbol{w}^* &= (w_1^*, w_2^*, \ldots, w_{2d}^*) \\ &= (1, 2, \ldots, d-1, d, d-1, \ldots, 2, 1) \end{aligned} \tag{A.5}$$

Those scores are given according to how far each element is differing from the missing data position in (A.1) or matching point position in (A.2). Later, the remaining values in $\boldsymbol{w}^*$ are normalized so that the sum of elements is 1.

$$\boldsymbol{w} = \frac{\boldsymbol{w}^*}{\sum_{i=1}^{2d} w_i^*} \tag{A.6}$$

It is worth mentioning that any values of missing positions in either vectors (A.1) or (A.2) are deleted from both vectors. If any element is deleted, both vectors should extend to include more values from further time points to keep the number of elements on both sides of the missing/matching point as d. This is because the missing data cannot be used to calculate distance.

A.3.2 Data Specification

The real world data is collected by the PeMS system traffic management centre in the Bay Area [107]. Each device sends one statistical record

at five-minute intervals. Each record contains a timestamp and some statistical values such as the number of vehicles passed during the last five minutes, i.e. the flow rate. Part of the data from one monitoring device on the road is used, which is the public Dodgers dataset [90]. The time range of data is from April 2015 to October 2015.

The original dataset contains MCR and MR missing values which are 5%. To test missing ratios higher than the original 5%, MCR missing values are added.

A.3.3 Experimental Setup

The experiments are conducted using Python programming language version 2.7 [121] for GSW and with the main package impyute version 0.0.4 [122] for the general *k*NN method in Ubuntu 16.04 LTS. Other methods are conducted in R programming language version 3.3.3 [123] and the package imputets version 2.5 [124] in Window 10. For each missing ratio, 5% non-missing instances are selected randomly as test data and each of them is marked as missing during one experiment so that the influence the missing ratio is negligible. Best parameter configurations are used if needed.

The imputation accuracy is measured by root mean square error (RMSE), which is also known as root mean square deviation. RMSE is selected as it is frequently used in previous literature to measure imputation results so it is easier for readers to compare with previous work. Later, the imputation results are analyzed using cross comparison and eigenvalues.

A.4 Results and Analysis

This section describes benchmarking results among five methods, followed by more detailed comparison between two *k*NN based methods.

The RMSE comparison results of different methods are shown in

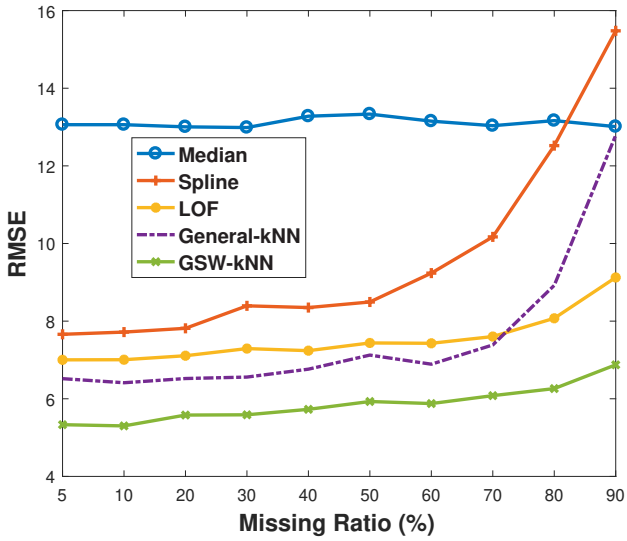

Figure A.3: RMSE of proposed GSW-*k*NN and four benchmarking methods. GSW-*k*NN is not only more accurate but also more stable when missing ratio is high.

Figure A.3. The missing ratio ranges from 5% to 90%. It is shown that RMSE values of GSW-*k*NN are the smallest compared to the other four methods. GSW-*k*NN performs 14% to 59% better than others and 34% better on average. Also, the RMSE values of GSW-*k*NN are very stable and the algorithm is robust. Except our proposed method, the general *k*NN method works better than the other three methods.

Now we focus on the comparison of two *k*NN based methods. Figure A.4 shows the improvement of accuracy using our proposed method than the general *k*NN w.r.t. RMSE. GSW-*k*NN is 18% to 46% better when being compared to general *k*NN. The squares represent the mean values of RMSE for each missing ratio and the bars indicate standard deviation. As it is shown in Figure A.4, GSW-*k*NN is consistently better than general *k*NN. GSW-*k*NN has smaller RMSE on average and less variation (standard deviation), which means it works more stable from 5% to 90% missing ratio than the general *k*NN method.

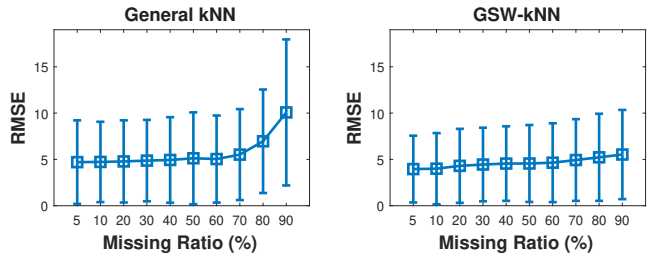

Figure A.4: RMSE and variation (standard deviation) of general *k*NN and GSW-*k*NN methods. GSW-*k*NN is more robust to high missing ratio.

The traffic data used in the experiments are distributed between 0 and 60. To understand more about how GSW-*k*NN performs in different ranges of the traffic, several flow ranges are selected, say, 0 to 10, 10 to 20, 20 to 30 and 30 to 40. 40 to 60 is not used due to very few data. In each range, 200 values are randomly selected. The improvement of RMSE from general *k*NN to the proposed GSW-*k*NN is shown in Figure A.5, indicating that in each of the four ranges, our proposed method performs better than general *k*NN with few exceptions. Especially, when the missing ratio is higher than 70%, the accuracy improvement given by GSW-*k*NN increased dramatically in all the four ranges. Among the four ranges, the improvement in the range 0 to 10 is the highest. The reason is possibly that night traffic is more consistent across different days compared to day-time traffic which exhibits large variations during holidays. Thus, the windowed algorithm can make more use of existing night-time data [116].

Figure A.6 and Figure A.7 show the cross comparison plots of the ground truth and the imputed data from the general *k*NN and our proposed method. The x-axis represents the ground truth values, and the y-axis represents imputed values. When the missing ratio is lower than 70%, the shapes of point distributions from two methods are similar with each other, as well as among different missing ratios within the same method. This indicates that the two methods are both working in a stable way when the missing ratio is not too high. To describe the distribution of the cross comparison points in detail, the eigenvalues

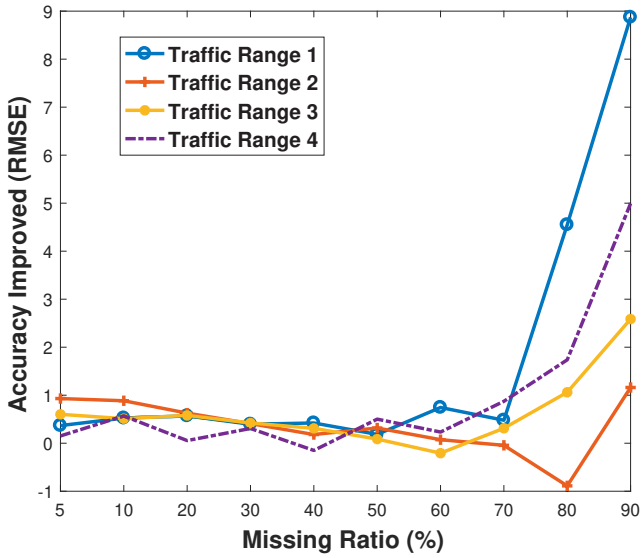

Figure A.5: The improvement of accuracy w.r.t. RMSE, when comparing GSW-*k*NN with general *k*NN. GSW-*k*NN imputation is performing better especially in traffic flow range 1 (night time traffic) with high missing ratio.

and eigenvectors of each covariance matrix is computed using singular value decomposition (SVD) method. According to the definition of the eigenvalues computed by SVD, the first eigenvalue depicts the data value range of the distribution, and the second eigenvalue represents the accuracy of the data imputation. It means that the smaller the second eigenvalue is, the more accurate the imputation is. If the distribution of points is similar to a diagonal straight line, the imputed values are similar to the ground truth values.

Figure A.8 shows the second eigen values of the covariance matrix between ground truth value and the imputed value by the method of GSW-*k*NN and general *k*NN. Its results are consistent with our previous results discussion. Our proposed method, GSW-*k*NN works very stable and better than general *k*NN.

Additionally, we also compared to PCA imputation from [125]. How-

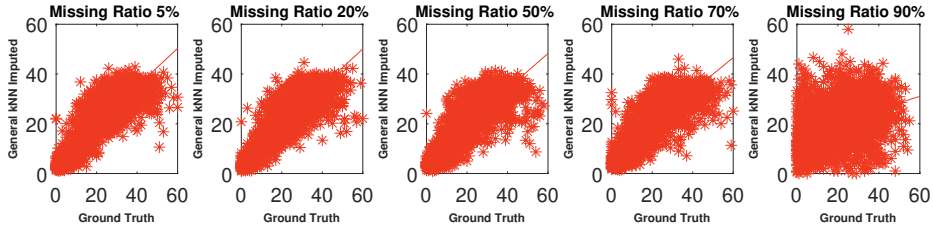

Figure A.6: The cross comparison between ground truth values and the imputed values by general *k*NN. Its performance drops suddenly when the missing ratio is higher than 70%.

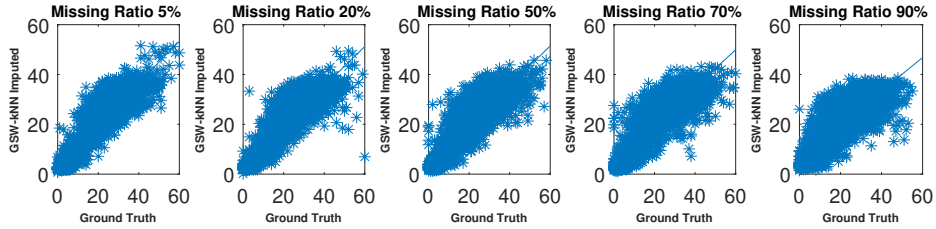

Figure A.7: The cross comparison between ground truth values and the imputed values by GSW-*k*NN. The distribution of points is more similar to a diagonal straight line than general *k*NN, especially when the missing ratio is higher than 70%. Thus, GSW-*k*NN is more accurate.

ever, it cannot perform imputation for datasets with very high missing ratio, such as 80% and 90%, so it is not included in the plots. The available results indicate that PCA is about 8% to 10% less accurate on average compared to GSW-*k*NN.

A.5 Conclusion

This paper proposes GSW-*k*NN for traffic time series imputation. The method takes gaps caused by missing values into consideration via using weights during distance measurement and uses windows to broaden the amount of neighbours. The experiment results from the public PeMS dataset are showing that GSW-*k*NN performs 14% to 59% better than benchmarking methods and 34% better on average. It is also more robust

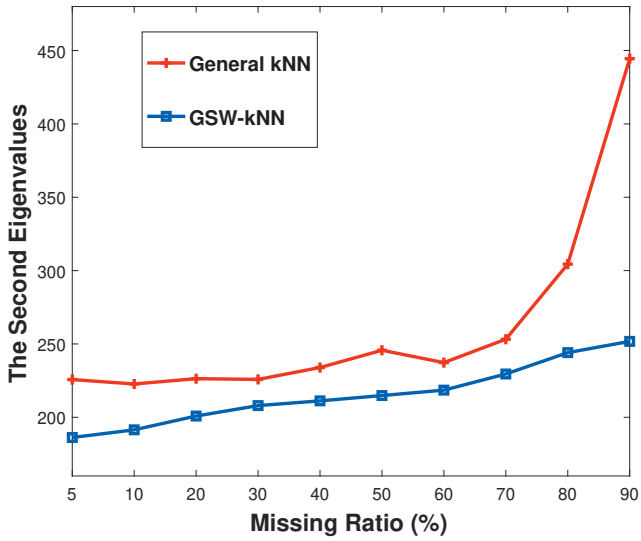

Figure A.8: The second eigenvalues of the cross comparison between ground truth values and the imputed values by the method of general *k*NN and GSW-*k*NN. GSW-*k*NN is giving smaller the second eigenvalues, hence the better imputation.

even for datasets with high missing ratios.

There are some issues that should be considered when using GSW-*k*NN. Firstly, more traditional time series based algorithms can be compared or used as part of ensemble methods as ensemble strategies are showing promising performance improvements compared to a single model for a variety of tasks [126, 127]. Secondly, our work does not distinguish data under normal situation with data under events. We plan to investigate those tasks in future work.

Acknowledgment

This work was supported by National Natural Science Foundation of China (NSFC), agreement 61364019.

B

Correcting and Complementing Freeway Traffic Accident Data Using Mahalanobis Distance Based Outlier Detection

Bin Sun, Wei Cheng, Guohua Bai, Prashant Goswami. Technical Gazette (ISSN: 1330-3651). 24(5), pp.1597–1607, October 2017.

Abstract

A huge amount of traffic data is archived which can be used in data mining especially supervised learning. However, it is not being fully used due to lack of accurate accident information (labels). In this study, we improve a Mahalanobis distance based algorithm to be able to handle differential data to estimate flow fluctuations and detect accidents and use it to support correcting and complementing accident information. The outlier detection algorithm provides accurate suggestions for accident occurring time, duration and direction. We also develop a system with interactive user interface to realize this procedure. There are three contributions for data handling. Firstly, we propose to use multi-metric traffic data instead of single metric for traffic outlier detection. Secondly, we present a practical method to organise traffic data and to evaluate the organisation for Mahalanobis distance. Thirdly, we describe a

general method to modify Mahalanobis distance algorithms to be updatable.

Index terms— Accident Data, Data Labelling, Differential Distance, Mahalanobis Distance, Outlier Detection, Traffic Data, Updatable Algorithm

B.1 Introduction

Nowadays, increasing road traffic is causing more accidents and it gains more attention from authorities. Therefore, a vast number of traffic monitoring devices have been installed to collect traffic data. As a consequence, a huge amount of traffic data has been archived, sometimes together with related information such as accident records [2]. However, patterns and relationships between the traffic data and accident records are invisible. To discover hidden information, the stored huge amount of data can be investigated using data mining techniques [128,129], especially supervised learning [130,131] for analysis and prediction. To do this, we need to know related traffic data given an accident record, i.e. labelled data is required. Nevertheless, accident records are neither accurate nor complete. In many authorities' databases, accident occurring time is estimated by the witnesses or drivers and duration of accident time is often missing. What is worse is that the direction of road where accident happened is also missing. Those problems make it impossible to know which archived traffic data is related exactly. Messy accident information is hard to be used directly to label traffic data in supervised learning [77]; even semi-supervised learning needs some initial labels [132]. Thus, it is necessary to correct and complement accident records.

The procedure of manual correction and complementation of accident records requires repeated actions and consumes a lot of time [133]. Besides, it is hard to make decisions regarding accident occurring time, duration and direction from millions of raw traffic values without any

external help. To regain the accident related information, in this work we developed a system to help correct and complement accident records. The system will calculate accident occurring time, duration and direction using the accident detection technique that we propose.

Accident detection techniques can be separated into two categories [73]. The first category provides "recognition" of accidents if the monitored traffic situation is similar to previous accidents situation. The second category discovers observations that are significantly different from typical values and this procedure is called outlier detection [76].

The first category includes conventional methods such as McMaster [74] as well as novel machine learning based classification methods [75]. The conventional methods usually require physical location characteristics like shape of the roads or multi-device data as part of the input, and machine learning based classification requires labelled data.

The second category outlier detection compares one observation with normal situation or prediction. There are two types of outliers, global outlier and local outlier [77]. Global outliers are considered as outliers regardless of the concept, whereas, local outliers are concept-related. For example, 40°temperature is normal in India, but outlier value in Sweden. Much research provided methods to detect outliers while assuming outliers as global for transportation [78]. However, efforts made to adapt local outlier detection for transportation are insufficient.

Further, existing research preferred to use single metric and its threshold to detect outliers. For example, [73, 79] compare monitored flow rate with its threshold. In [80] researchers use speed to do comparison and detection, and [81, 82, 83] use density (occupancy). As there are fundamental differences among characteristics of different roads, procedures considering only single metric are unsuitable. For example, in [84] flow speed is influenced more than flow rate during breakdown events. In [85] events change flow rate suddenly but maintain flow speed. Some data also shows that during certain events, both flow speed and rate can change. However, we cannot find existing research which considers

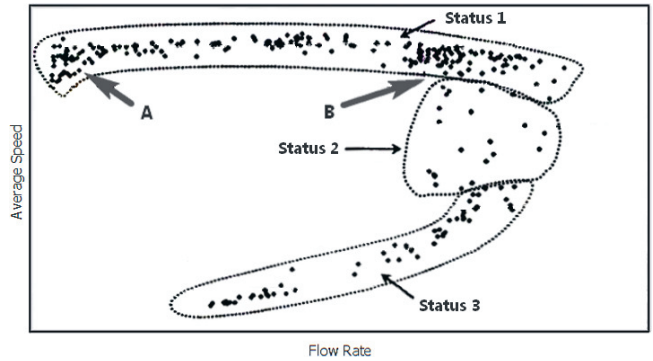

Figure B.1: Speed-flow fundamental diagram. Status (dotted circle) 1: under-saturatated flow. 2: queue discarge flow. 3: over-saturated flow. High density area A: typical night time flow. B: day time flow.

this kind of change in data. Though some work has been done with regard to transit fundamental diagram [85], none considered differential time-varied fundamental diagram. Gonzalez [80] uses the term "level of change" to describe one kind of change, but this change detection is based only on speed. Mentioning differential calculation, the most popular algorithms are ARIMA-like algorithms. Nevertheless, they can handle only single variable and the variable needs to be stationary [134,135]. Moreover, their related vectorized versions require more assumptions which are not practical [136].

In this paper, we propose to use Mahalanobis distance [108] (M-distance) based analysis to detect accidents. M-distance is a general distance used in multivariate analysis and has been widely used for detecting outliers [137, 138, 139]. However, few work used it for detecting traffic accidents. One possible reason might be that few researchers are considering more than one metric together, so M-distance is not necessary. Even if multivariate data is considered, another problem is M-distance is only for multivariate data that is clustering like filled ellipses, i.e., data is normally distributed [108, 140, 141]. However, traffic data in traditional speed-flow fundamental diagram [24] cannot hold this assumption as shown in Figure B.1.

B.2 Data Pre-Processing

In this section we propose a general method that can pre-process traffic data to be suitable for the main methodology in the third section.

B.2.1 Metrics Selection

Flow rate, speed and density are three fundamental metrics in traffic engineering [24]. It is possible to calculate one metric given both the other two, so two metrics should be considered at the same time. Previous research prefers to consider only one of them within time domain, usually flow rate or speed. We use both flow rate and speed within the time domain.

B.2.2 Time-Separated Data Organisation

As shown in the density plot of speed-flow fundamental diagram (Figure B.1), data instances (points) are clustering mainly as day time and night time parts with transits between them. It is unsuitable to use M-distance directly in this conventional diagram. One solution is to find a time period to organise data according to time of day. The results are time-separated datasets. The time period should separate different data points into several datasets. Data in each dataset are clustered as filled ellipse. Additionally, the separation should keep the differences between neighbouring datasets ignorable (insignificant) to avoid breaking continuous time series data, otherwise the time period should be changed to a smaller value.

Below is a mathematical description of how hypothesis testing [142] can be used to evaluate time-separated traffic data.

H_0 (null hypothesis): there is no difference among all datasets. Consequently, its competing hypothesis H_1 (alternative hypothesis) is: there are differences among all or some datasets.

In this hypothesis testing, we are considering two-dimensional data

with flow rate r and speed s. Both r and s are normally distributed in datasets. We now pick two datasets (two groups of data according to different time of day) and name them Dataset 1 (D_1) and Dataset 2 (D_2). Thus, we get two distributions for datasets D_1 and D_2 as:

$$(r_1, s_1) \sim \mathcal{N}_q(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \tag{B.1}$$

$$(r_2, s_2) \sim \mathcal{N}_q(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \tag{B.2}$$

where q = 2 (2 dimensions); $\boldsymbol{\mu}$ is dataset's centroid and $\boldsymbol{\Sigma}$ is covariance matrix. According to the properties of operations on independent multivariate normal distributions, a linear combination of multivariate normal distributed variables is still distributed normally:

$$\sum_{i=1}^{n} k_i \boldsymbol{x_i} \sim \mathcal{N}_q(\sum_{i=1}^{n} k_i \boldsymbol{\mu}_i, \sum_{i=1}^{n} k_i^2 \boldsymbol{\Sigma}_i) \tag{B.3}$$

Therefore, the difference between two neighbouring datasets:

$$((r_1, s_1) - (r_2, s_2)) \sim \mathcal{N}_2(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2) \tag{B.4}$$

is a multivariate normal distribution. Now, the null hypothesis can be written as:

$$H_0 : \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 = \mathbf{0} \tag{B.5}$$

i.e.

$$H_0 : \boldsymbol{\mu}_{D1-D2} = \mathbf{0} \tag{B.6}$$

which means no difference between two centroids. Meanwhile, the alternative hypothesis is equivalent to:

$$H_1 : \boldsymbol{\mu}_{D1-D2} \neq \mathbf{0} \tag{B.7}$$

For that pair of hypotheses, we can calculate p-values by using hypothesis testing and compare the testing results with significance level to know if there are significant differences among datasets. When the separation is done, we can proceed to analyse separated data using the methodology proposed in the next section.

B.3 New Accident Information from Outlier Detection

In this section we firstly introduce existing works in the first two subsections and then propose our modifications and improvements so that we can correct and complement traffic accident information.

B.3.1 Mahalanobis Distance

Euclidean distance [117] is a widely used traditional and ordinary distance for outlier detection [143, 144]. It is easy to understand, implement and fast to calculate [145]. However, it cannot represent a concept-related distance, as it does not consider the shape of distribution (scatter) [146]. To avoid this weakness, M-distance [108] based analysis is used in this work to detect multivariate outliers.

Here is a brief description of M-distance. Suppose $i(1 \leq i \leq n)$is instance index and $j(1 \leq j \leq k)$ is variable index in dataset $\boldsymbol{X} = [[a_{ij}]]$ that contains n observations of k variables. The covariance between variable l and variable m is:

$$q_{lm} = \frac{1}{n-1} \sum_{i=1}^{n} (a_{il} - \mu_l)(a_{im} - \mu_m) \tag{B.8}$$

where μ_l and μ_m are variables' expected values.

Thus the covariance matrix of dataset $\boldsymbol{X}$ can be expressed as a $k \times k$ matrix $\boldsymbol{\Sigma} = [[q_{lm}]]$.

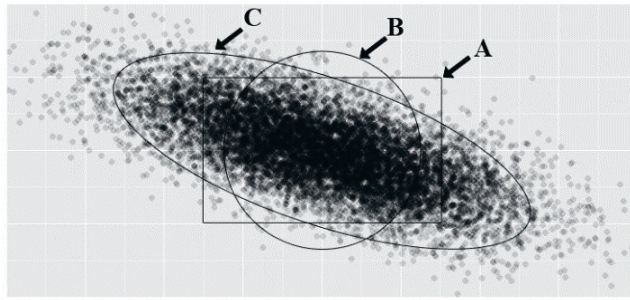

Figure B.2: Example of different thresholds. A: the rectangle shows threshold considering each metric separately. B: the circle shows threshold considering two metrics together. C: the ellipse shows M-distance threshold considers two metrics together as well as the shape of distribution.

Finally, M-distance to centroid is the distance between instance $\boldsymbol{x_i}$ and centroid $\boldsymbol{\mu}$:

$$MD_i = \sqrt{(\boldsymbol{x_i} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\boldsymbol{x_i} - \boldsymbol{\mu})^T} \tag{B.9}$$

Although M-distance can also be used to measure distance between any two points, it stands for "M-distance to centroid" in our work especially. When being compared with conventional thresholds, Figure B.2 shows that M-distance is suitable for multivariate outlier detection and can provide better thresholds by considering the shape of distribution [147].

B.3.2 Adaptive Threshold

Though M-distance considers the shape of distribution, it comes with a shortcoming. When calculating the covariance, M-distance is sensitive to outliers and extremes [141, 148]. To reduce the sensitivity, adaptive reweighted location and scatter estimation [147] (ARW) is used to determine an adaptive threshold c_n. If the distribution function of χ_p^2 is noted as $G(u)$, and the empirical distribution function as $G_n(u)$, where n is the number of observations, ARW can be described as pseudo code

Algorithm ARW

function ARW($\mathbf{x}$)
  $\delta \leftarrow \chi^2_{2;1-0.02}$
  $p_n \leftarrow \sup(G_n - G)^+$
  $p_c \leftarrow \frac{0.234}{\sqrt{n}}$
  if $p_n > p_c$ **then**
    $\alpha_n \leftarrow p_c$
  else
    $\alpha_n \leftarrow 0$
  end if
  $F \leftarrow 1 - \alpha_n$
  $c_n \leftarrow \mathrm{CDF}(F)$
  return c_n
end function

below. p_c is used to describe difference between theoretical distribution and empirical distribution.

A data instance can be detected as an outlier if its M-distance is bigger than the adaptive threshold. As estimating the centroid and scatter consumes a vast amount of computing resources, minimum covariance determinant [149] (MCD) is used to calculate centroid and scatter. Outliers can now be detected by comparing time-separated data's M-distance with adaptive threshold, and those outliers are time-separated outliers.

B.3.3 Differential Outlier

Though flow rate and speed are considered together within time domain, the detection is not using a nature of the time domain that is differential characteristic. Thus, we also propose to use differential data to detect outliers. For differential calculation, we can get the differential data from two consecutive instances. That is, the differential data is a result of subtracting one data instance with its neighbour in a consecutive

time series.

If we note each instance as $\boldsymbol{x_i} = [r_i, s_i]$, we can get the differential data as:

$$\boldsymbol{x_{diff}} = \boldsymbol{x_{i+1}} - \boldsymbol{x_i} = [r_{i+1} - r_i, s_{i+1} - s_i] \tag{B.10}$$

For example, if there are originally 100 instances, we can have 99 differential instances. Then the differential data is divided by time of day according to the timestamp of instance $\boldsymbol{x_i}$. Each differential dataset has almost five thousand instances. The remaining analysis procedure is the same as normal outlier detection. Each differential dataset was analysed using M-distance to detect outliers. Now the detected outliers are time-separated differential outliers.

B.3.4 Monthly Updatable Algorithm

The aforementioned outlier and differential outliers are detected by comparing data instances with thresholds calculated from all archived data. Therefore, the model that is being compared with is not timely dynamic. This might cause two time-related problems. Firstly, if some weeks or months are special, the algorithm may behave unexpectedly. Secondly, the algorithm cannot reflect the fact that more and more cars are coming on the road nowadays gradually. To solve those problems, one solution is to calculate weighted instances in ARW and give more weighting to recent instances. This solution requires the whole archived data and increases calculation cost. Another solution is to use updatable method.

Updatable algorithms [150] can produce a new result from the latest old result and new data without old data, which can dramatically reduce calculation time. Thus, we improved the original algorithm to be updatable which will consider recent data instances more than other history instances when deciding if the new instance is an outlier. In addition, we weighted the influence of old data by limiting the number of old instances, so the new result gets adjusted according to the new data dynamically.

Below is a description of the improved algorithm. Consider $\boldsymbol{X_1} = [\boldsymbol{r_1}, \boldsymbol{s_1}]$ containing n_1 old instances and $\boldsymbol{X_2} = [\boldsymbol{r_2}, \boldsymbol{s_2}]$ containing n_2 newly arrived instances, we can use the existing covariance matrix of $\boldsymbol{X_1}$ and instances in $\boldsymbol{X_2}$ to detect updatable outliers. If the old covariance matrix is noted as:

$$\boldsymbol{C_1} = \begin{bmatrix} \mathrm{COV}(\boldsymbol{r_1}, \boldsymbol{r_1}) & \mathrm{COV}(\boldsymbol{r_1}, \boldsymbol{s_1}) \\ \mathrm{COV}(\boldsymbol{s_1}, \boldsymbol{r_1}) & \mathrm{COV}(\boldsymbol{s_1}, \boldsymbol{s_1}) \end{bmatrix} \tag{B.11}$$

and the covariance matrix for new data is:

$$\boldsymbol{C_2} = \begin{bmatrix} \mathrm{COV}(\boldsymbol{r_2}, \boldsymbol{r_2}) & \mathrm{COV}(\boldsymbol{r_2}, \boldsymbol{s_2}) \\ \mathrm{COV}(\boldsymbol{s_2}, \boldsymbol{r_2}) & \mathrm{COV}(\boldsymbol{s_2}, \boldsymbol{s_2}) \end{bmatrix} \tag{B.12}$$

then overall updatable covariance matrix is:

$$\boldsymbol{C_0} = \begin{bmatrix} \mathrm{COV}(\boldsymbol{r_0}, \boldsymbol{r_0}) & \mathrm{COV}(\boldsymbol{r_0}, \boldsymbol{s_0}) \\ \mathrm{COV}(\boldsymbol{s_0}, \boldsymbol{r_0}) & \mathrm{COV}(\boldsymbol{s_0}, \boldsymbol{s_0}) \end{bmatrix} \tag{B.13}$$

where $\boldsymbol{r_0}^{\mathrm{T}} = [\boldsymbol{r_1}^{\mathrm{T}}\ \boldsymbol{r_2}^{\mathrm{T}}]$, $\boldsymbol{s_0}^{\mathrm{T}} = [\boldsymbol{s_1}^{\mathrm{T}}\ \boldsymbol{s_2}^{\mathrm{T}}]$.

In accordance with Eq. (B.8), we derived:

$$\begin{aligned} &\mathrm{COV}\left(\boldsymbol{r_0}, \boldsymbol{s_0}\right) \\ &= \mathrm{COV}\left(\begin{bmatrix} \boldsymbol{r_1} \\ \boldsymbol{r_2} \end{bmatrix}, \begin{bmatrix} \boldsymbol{s_1} \\ \boldsymbol{s_2} \end{bmatrix}\right) \\ &= \frac{n_1 - 1}{n_0 - 1} \times \mathrm{COV}\left(\boldsymbol{r_1}, \boldsymbol{s_1}\right) + \frac{n_1 \times (n_0 - n_1) \times \mu_{r_1} \times \mu_{s_1}}{n_0 \times (n_0 - 1)} \\ &+ \frac{n_2 - 1}{n_0 - 1} \times \mathrm{COV}\left(\boldsymbol{r_2}, \boldsymbol{s_2}\right) + \frac{n_2 \times (n_0 - n_2) \times \mu_{r_2} \times \mu_{s_2}}{n_0 \times (n_0 - 1)} \\ &- \frac{n_1 \times n_2 \times (\mu_{r_1} \times \mu_{s_2} + \mu_{r_2} \times \mu_{s_1})}{n_0 \times (n_0 - 1)} \end{aligned} \tag{B.14}$$

where $n_0 = n_1 + n_2$. The other three items in $\boldsymbol{C_0}$ can also be calculated using similar equations. (μ_{r_1}, μ_{s_1}) and (μ_{r_2}, μ_{s_2}) are centroids of old and

new data instances respectively. Therefore, new updatable centroid is calculated as:

$$(\mu_{r_0}, \mu_{s_0}) = \left(\frac{n_1 \times \mu_{r_1} + n_2 \times \mu_{r_2}}{n_0}, \frac{n_1 \times \mu_{s_1} + n_2 \times \mu_{s_2}}{n_0} \right) \tag{B.15}$$

To take advantage of this new algorithm, time series data can be grouped by a time gap, say a month, and then each group's centroid, scatter and threshold can be calculated separately. Hence, we get updatable M-distance based algorithm that leads to Updatable Time-Separated Outliers and Updatable Time-Separated Differential Outliers.

B.3.5 Accident Occurring Time, Duration and Direction

Through previous calculation, we already have four different outliers, and we need to select suitable ones to use according to real world data. For one data instance, the system will calculate its M-distance and then divide by adaptive threshold. If the quotient is bigger than threshold, and its timestamp is near selected accident record, it is the accident occurring time. Otherwise, the biggest quotient should be used.

The outlier following occurring time is accident ending time which means the traffic starts to recover from the accident. Sometimes traffic situation will resume from one accident gradually and the second outlier is unobvious, then the largest quotient related timestamp in three hours after the accident occurring time will be considered as road cleaning time. This is due to the fact that most accident duration is less than three hours [151].

Accident direction is given by accident indicator which measures the deviation of accident traffic from non-accident traffic (in Figure B.3).

The indicator is defined as:

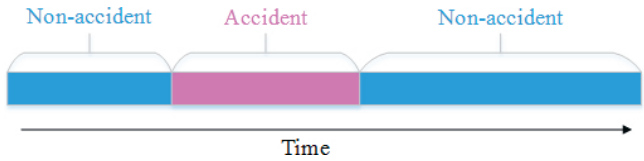

Figure B.3: The indicator of traffic direction is calculated from the difference between accident traffic and non-accident traffic.

$$indicator = \frac{\mathrm{MD}(\boldsymbol{\mu}_{\text{accident}} - \boldsymbol{\mu}_{\text{non.accident}})}{c_n} \tag{B.16}$$

Centroids of traffic during accident time $\boldsymbol{\mu}_{\text{accident}}$ and non-accident time $\boldsymbol{\mu}_{\text{non.accident}}$ are calculated respectively. The difference between those two centroids, i.e., differential centroid, is analysed using differential outlier detection. The M-distance is compared with adaptive threshold to get accident indicator. The biggest indicator gives the detection device as well as the direction of accident.

B.3.6 Proposed Steps

The aforementioned procedure is robust to outliers, also adaptive to both degree of freedom and number of instances.

Our work uses the procedure to analyse traffic data as shown in Figure B.4.

Firstly, the system queries the data from database. The original time series is processed according to the left side steps while the differentialized time series is processed according to the right side steps. Secondly, the time series will be separated into several datasets according to data's timestamps. The results can be visualized as two types of diagrams. Thirdly, the time-separated data will be analysed. Thereafter, both updatable and non-updatable algorithms are used to detect outliers. Thus, non-differential and differential data lead to four types of outliers. Finally, the system considers those outliers together with the inaccurate or

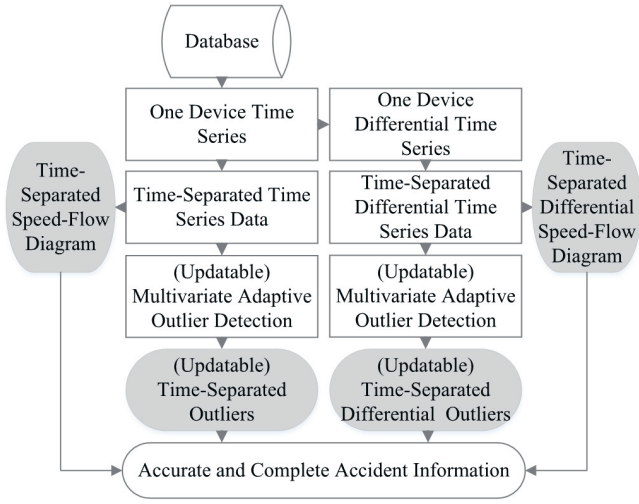

Figure B.4: Main steps of the proposed method. The data is analysed device by device. In-process outcomes (grey coloured) are two types of diagrams and four types of outliers. Final outcome is accurate and complete accident information.

incomplete accident records to produce accurate and complete accident information as suggestions to fix the record.

B.4 Experiments and Results

In this section, the proposed steps are used to process real world data. Our source of data comes from devices monitoring a freeway named Kunshi in Southwest China (in Figure B.5). The freeway is 70 kilometres long and thirty devices are collecting traffic data. The devices report traffic statistics every 5 minutes. Each reported record contains total number of cars in 5 minutes and time averaged speed during the same period. The data is from April 2013 to May 2014, and more than six million records are stored in the database.

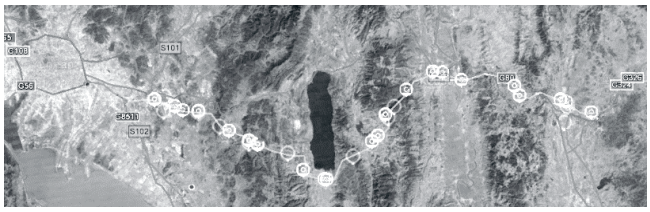

Figure B.5: There are totally 30 monitoring devices as circled on map. Multi circle icons may be displayed as one when close to each other. Total length of monitored freeway is 70 km.

B.4.1 Cleaning Data

The raw data needs to be cleaned before data pre-processing. For each monitoring device, there can be one or more cameras and each camera will report records of one lane statistic data independently. This brings two problems. Firstly, one monitoring device generates multiple records according to lanes and those records should be aggregated. Secondly, the arriving timestamps of reported records from different cameras might have several seconds' delay which makes it harder to find the right records to aggregate. After analysing the raw data, we found that the records from one device have the same timestamp until minute level but different at the second level. We aggregate the records according to timestamps until minute level to get data instances. The aggregated results show that this method is working correctly when being compared with raw data.

B.4.2 Hourly Data as Time-Separated Data

Among collected traffic metrics, flow rate and speed are used in our analysis as proposed in the second section. Cleaned data is plotted in Figure B.6. The road usually carries under-saturated flow that is part of the conventional speed-flow fundamental diagram in Figure B.1.

The data then is ready for organisation using the method in section 2.2. We found that dividing data into datasets by "hour of day" is suitable for M-distance based algorithm. Each dataset of one device has about

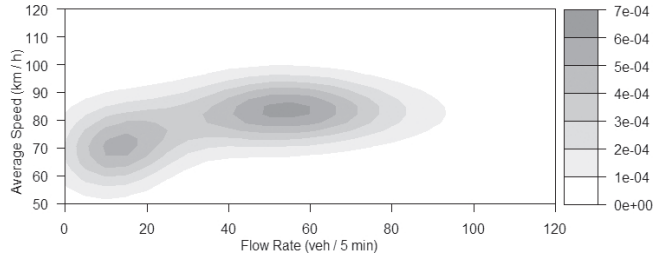

Figure B.6: Density of speed-flow fundamental diagram for one device data during the whole day. The distribution cannot hold M-distance assumptions.

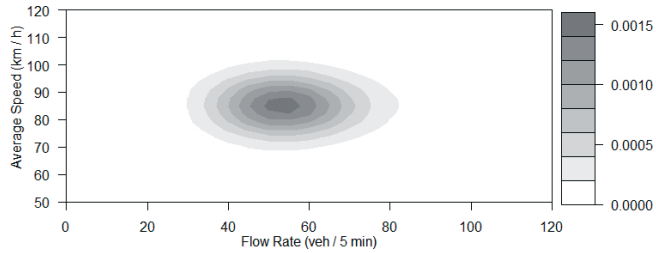

Figure B.7: Density of hourly speed-flow diagram for one device data (10 AM to 11 AM). The distribution can hold M-distance assumptions.

five thousand instances. Figure B.7 shows an example that the density plot which shows that the data points from a device during one hour are clustering as an ellipse and suitable for M-distance to use.

By using inferential statistics and hypothesis testing, we get p-values of differences among all hourly datasets. As shown in Figure B.8, there are significant differences among 30% of those hourly dataset pairs, so it is necessary to analyse data according to its hour of day. Thus, the null hypothesis H_0 is rejected and H_1 is accepted. Therefore, the data needs to be divided according to its hour of day before analysis.

On the other hand, no pairs of neighbouring hourly datasets are significantly different (all dark cells are connected). Which means the transits between neighboured hourly datasets are smooth. Therefore, choosing one hour as the time gap not only separates different data but

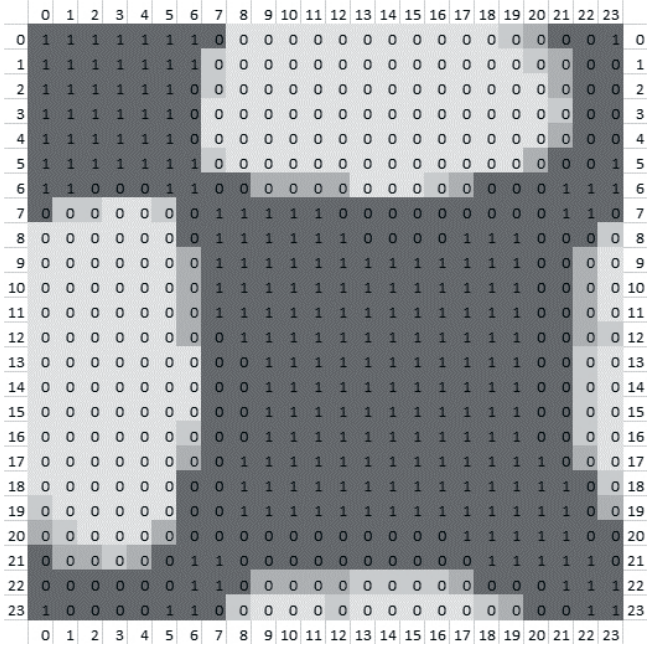

Figure B.8: 30% of pairs (lightest grey cells) are under or equal to 0.02 (values are rounded to 1 or 0), so data should be seperated by hour of day before analysis. 35% are under or equal to 0.05. 41% are under or equal to 0.1 (darkest cells), which means transits of hourly data are smooth and suitable.

also keeps the transits smooth.

B.4.3 Different Outliers in Speed-Flow Diagram

After data organisation, M-distance analysis introduced in the third section is used to detect outliers.

Below is an example of analyses result from a dataset that contains data from 10 AM to 11 AM. Figure B.9 is the dot plot of Figure B.7.

After applying adaptive M-distance outlier detection, we calculated outliers in hourly speed-flow fundamental diagrams. Non-hourly outliers and hourly outliers are plotted respectively with their thresholds.

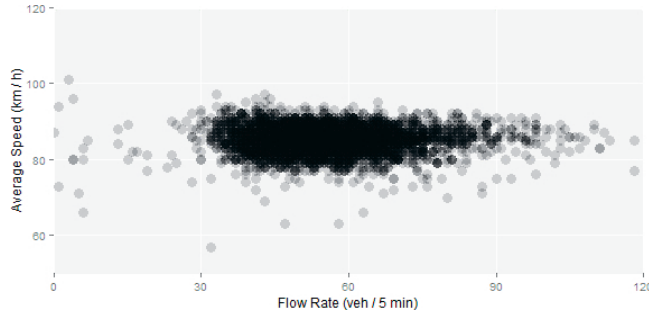

Figure B.9: Before adaptive M-distance outlier detection, one hourly dataset in speed-flow diagram (10 AM to 11 AM).

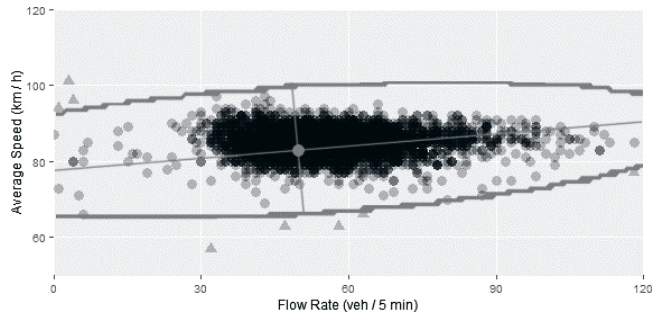

Figure B.10: After using all data (non-hourly data) in adaptive M-distance outlier detection. Non-hourly outliers are marked as triangles in speed-flow diagram, unacceptable poor quality (10 AM to 11 AM).

When being compared to non-hourly outliers and threshold (in Figure B.10), hourly outliers (in Figure B.11) are more reasonable.

B.4.4 Hourly Differential Outlier in Speed-Flow Diagram

Using differential calculation, we can get differential datasets. The differential data points are clustering as a cross shape instead of ellipse, so it is no possible to use M-distance directly. However, if we plot hourly differential datasets, we can see the expected ellipse (in Figure B.12).

Applying hourly adaptive M-distance analysis, we finally calculated

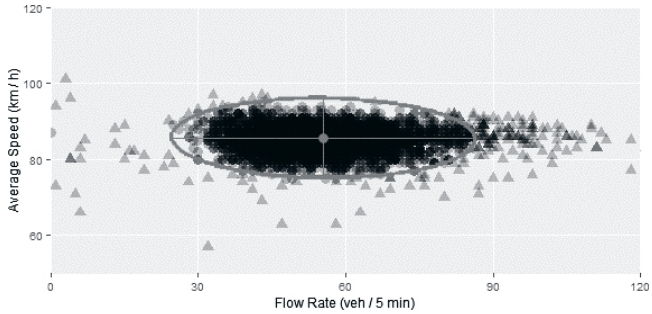

Figure B.11: After using hourly data in adaptive M-distance outlier detection. Hourly outliers are marked as triangles in speed-flow diagram. Quality is better than non-hourly outliers (10 AM to 11 AM).

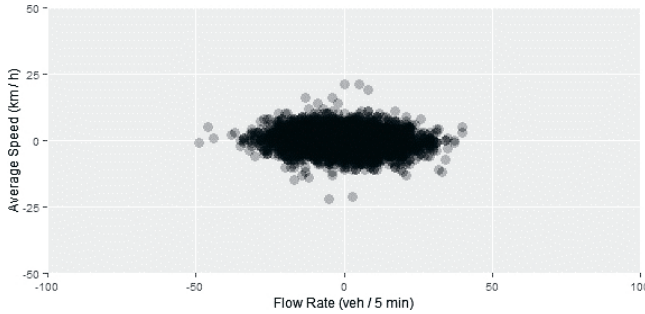

Figure B.12: Before adaptive M-distance outlier detection, one hourly differential dataset in differential speed-flow diagram (10 AM to 11 AM).

hourly differential outliers. When being compared to non-hourly differential outliers (in Figure B.13), hourly differential outliers in differential speed-flow diagram (Figure B.14) shows improvement that the threshold ellipse is more reasonable.

B.4.5 Different Outliers in Time Series

We can see outliers in speed-flow diagrams, but it is hard for analysts to use. Time-series plot is necessary for further analysis. Figure B.15 displays several hours traffic situation on a holiday. The hourly outliers spread over this period. Instead, hourly differential detection is more

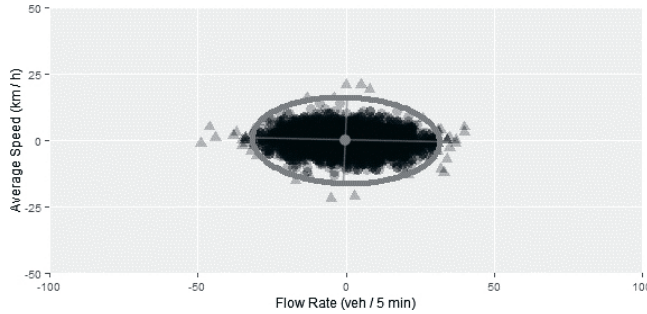

Figure B.13: After using all data (non-hourly data) in adaptive M-distance outlier detection. Non-hourly differential outliers are marked as triangles in differential speed-flow diagrams (10 AM to 11 AM).

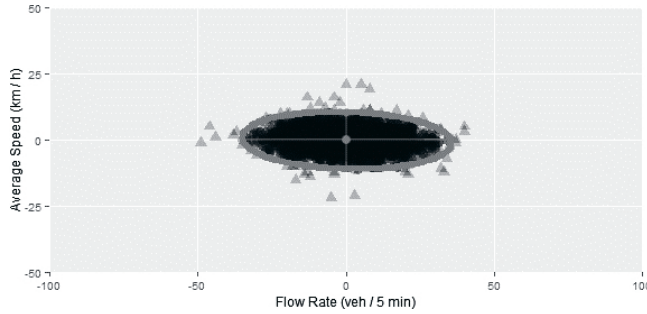

Figure B.14: After using hourly data in adaptive M-distance outlier detection. Hourly differential outliers are marked as triangles in differential speed-flow diagrams. Quality is slightly improved from non-hourly differential outliers (10 AM to 11 AM).

stable and accident is detected correctly.

In our system, one month length is used in updatable algorithm to make sure there is enough data to be analysed to update centroids and thresholds. One important thing to notice is that n_1 will be limited to maximum 360 which is product of 12 instances per hour and 30 days per month, i.e., the old result has less weighting than the new one.

Performance of the updatable algorithm is usually similar as the original one, but it performs better when there is a change for monthly traf-

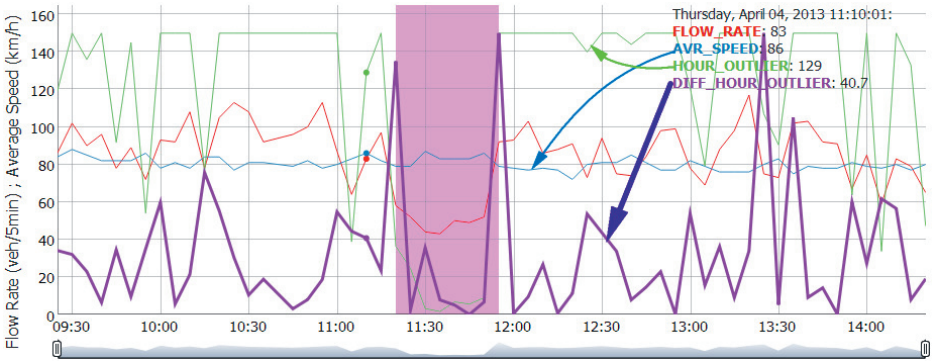

Figure B.15: Hourly differential outlier can detect accident correctly. It is robust to extreme high traffic during holidays. Detected accident duration is shaded. Outlier threshold is set to 100.

fic data. For example, when the biggest annual festival came earlier (the festival is based on lunar calendar), updatable detection algorithm adjusts to this change and can stay below threshold, while the original hourly outlier detection algorithm cannot get used to this situation and gives many outliers (Figure B.16). Outlier displayed threshold is enlarged from 1 to 100 as well as the quotient of data's M-distance and threshold to ease visualisation. For the same reason, displayed maximum quotient values are limited to 150.

Besides, updatable differential hourly outlier performs well even during abnormal fluctuation in holidays (Figure B.17).

B.5 Interactive User Interface

On one hand, R [152] provides a multiplatform commonly used environment [153,154] to perform state-of-the-art data analysis [155] (R environment is used throughout this paper). On the other hand, the speed-flow diagrams (Figure B.9 to Figure B.14) are hard to understand and use, it is necessary to have a rich-media user interface to provide illustrative information for analysts. Web-based system is widely used for data

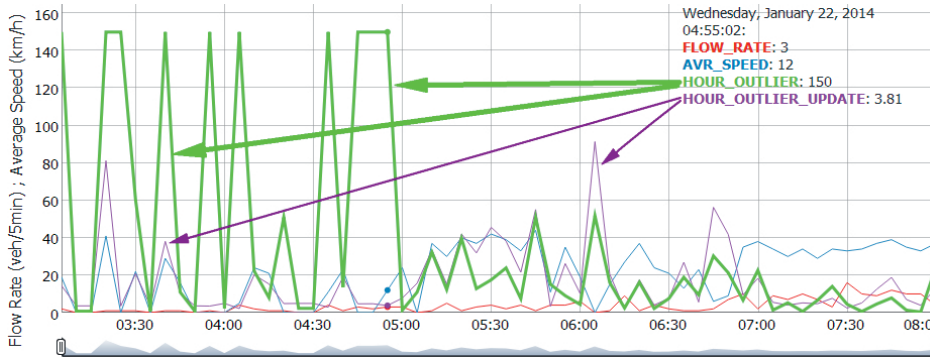

Figure B.16: There is no accident in the plotted duration and direction. Updatable hourly outlier is adopting to traffic situation change and more accurate than normal hourly outlier. Outlier threshold is set to 100.

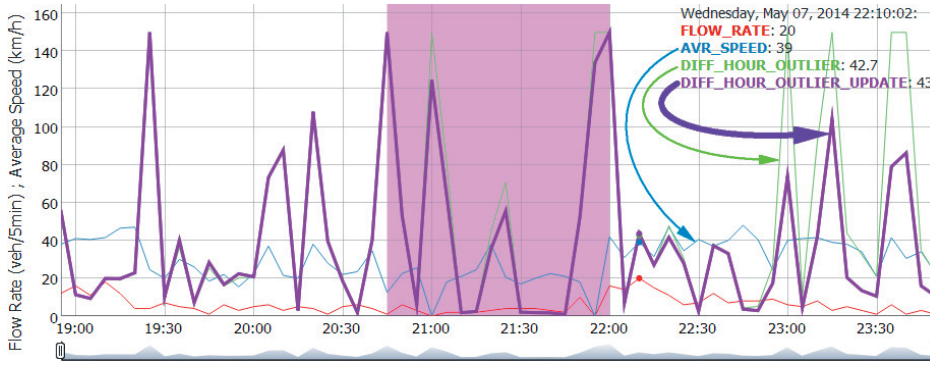

Figure B.17: Updatable differential hourly outliers can detect the accident, and stays stable after the accident compared with non-updatable ones. Detected accident duration is shaded. Outlier threshold is set to 100.

Figure B.18: Main panel. It contains four areas. Details are explained in the following zoomed in figures.

monitoring and visualization due to its excellent display effect and user-friendly interface [156, 157, 158]. Shiny [159] is R's web framework and both of them are available as open source software under GNU General Public License. Based on the proposed algorithms, we developed an interactive system for analysts using R and Shiny.

As shown in Figure B.18, the main panel has four areas. The first area is used to select which panel to display.

The source area contains three subareas (Figure B.19). In metric selection subarea, the analyst selects some metrics that should be displayed. Flow rate, speed and two updatable types of outliers are selected by default, the other two types of non-updatable outliers are optional. Then the analyst selects one accident that needs to be investigated from accident selection subarea. According to the selected accident, the system finds out the nearest four devices for both directions. In addition, two buttons named "previous accident" and "next accident" can be used to

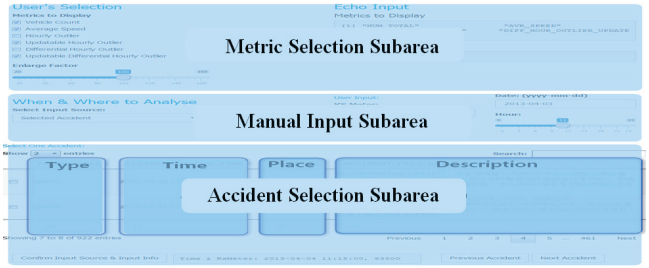

Figure B.19: The Source Area structure. It contains three subareas and the details are shown in the following three figures.

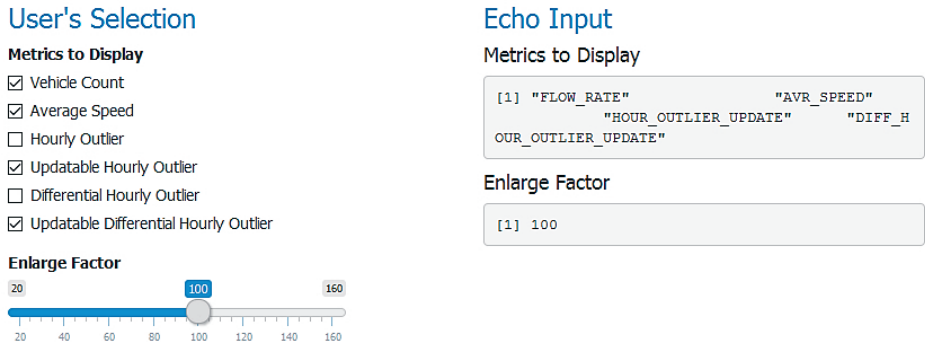

Figure B.20: Metric Selection Subarea for Data Source Area.

navigate among accidents when it is necessary to go through the accidents one by one. Besides using accidents, the analyst can also manually fill a time point and a place in manual input subarea for the system to pick related devices.

On the top left of Metric Selection Subarea, there are several metrics that are ready to be displayed in the plot. As mentioned in the previous chapter, different lanes carry different level of flow, to ease the view of outlier displayed threshold, an enlarge factor is used. All the selections are echoed from server to make sure the actions are right.

The figure below shows Manual Input Subarea. For "user manual input source" mode, the system requires a rough milestone position and

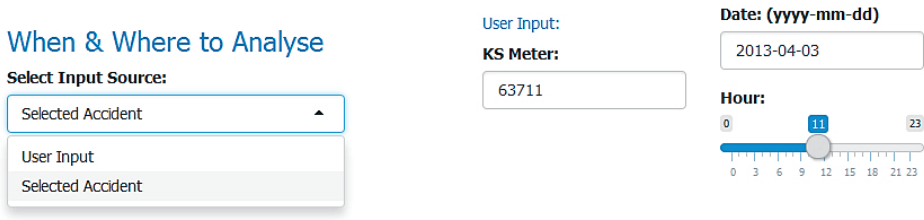

Figure B.21: Manual Input Subarea for Data Source Area. The left side is the switch of source "user manual input" or "selected accident". The right side is the manual input of milestone and time.

a rough time.

For "selected accident source" mode, Figure B.22 below shows the accidents to be selected. The accident information includes the type of accident (archived in different tables), raw recorded accident time, milestone position and brief accident fact (which is blurred for privacy).

Given the time and four devices from previous step, related data is analysed by the system and the results are displayed in the plot area (Figure B.20). The data from two devices in upstream direction is shown in the top subarea, one device is ahead of accident point and the other behind. The data from downstream devices is displayed in the bottom subarea. Then the selected accident is displayed in the middle to ease the analyst's comparison with plots.

Taking one plot as an example (Figure B.24), the system suggests a new occurring time and duration for the selected accident shown as the shaded area. The analyst can check metric values by moving and hovering mouse. Then the analyst can tune the system suggested information and confirm it. Hence, there is an accurate accident occurring time. Two new attributes are accident duration and direction. Based on the four accident indicators, the analyst can choose the biggest one which means the most obvious detection. Accident information is updated when the analyst confirms the analysis results. When the system promotes confirmation window, those attributes are stored in addition to the existing

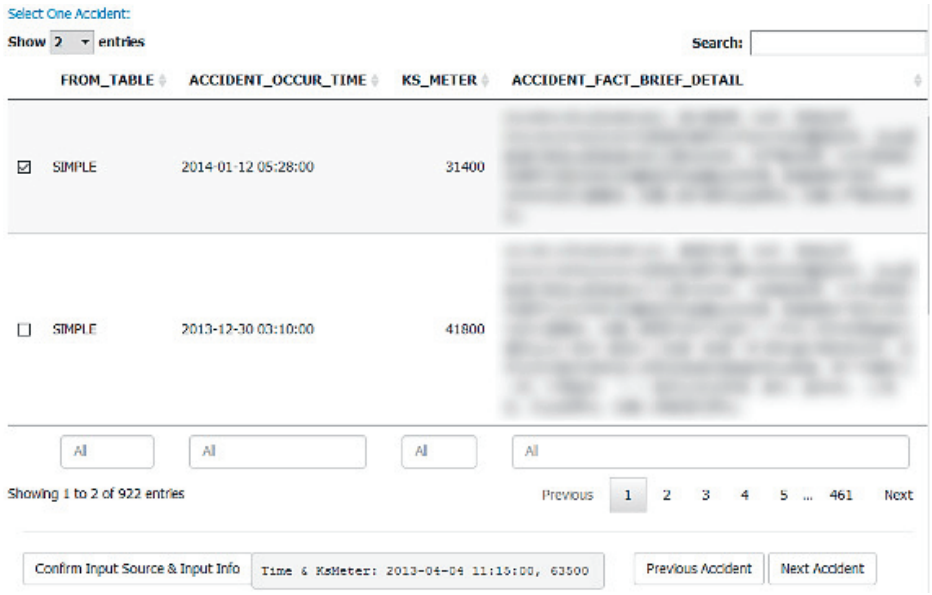

Figure B.22: Accident Selection Subarea for Data Source Area. It is the most important subarea as the selection leads to four related devices. The information shown here includes accident's type, raw time, position and description, which helps users to idenfy the correct accident.

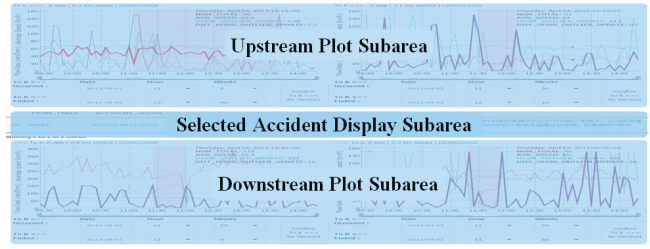

Figure B.23: Plot Area. It contains three subareas. The selected accident is displayed in the middle, surrounded with related traffic data from four related devices. The Selected Accident Display Subarea shows the same information as Accident Selection Subarea from source as a confirmation, and the plot is shwon in detail in the next figure.

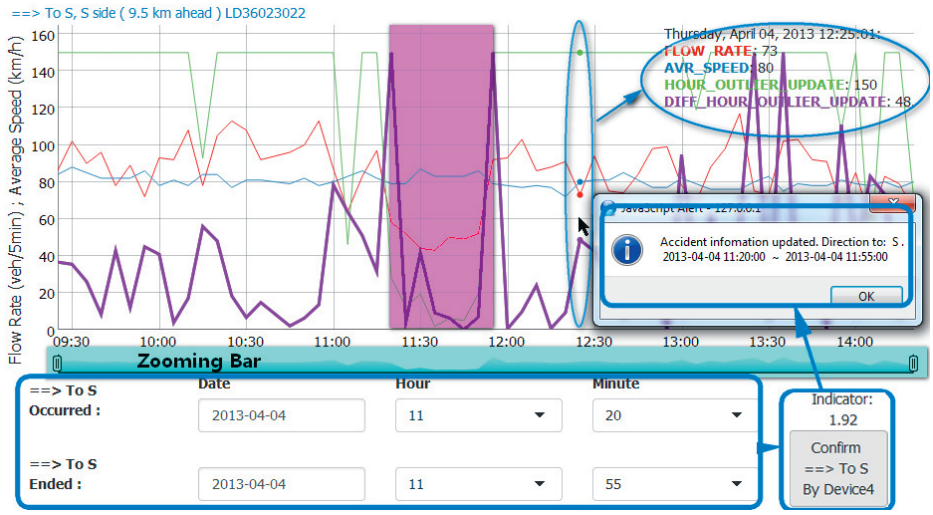

Figure B.24: One device's analysis result plot, part of plot area. Detected accident duration is shaded (11:20 to 11:55). Top-right values follows mouse hovering position. Zooming in or out can be done using zooming bar or direct mouse-drag selection. The four plots are syncronized for the zooming rate, which means zooming one plot will also let other three plots zoom to the same rate.

ones.

The plot title indicates only device distance and device ID, more detailed information such as number of monitored lanes, device type, installation place and surrounding environment is shown in the related devices area below the plot area.

During the above procedure, if the analyst wants to analyse the data deeply, "details" panel is useful. There are four areas in this panel (Figure B.25). The top area provides extra metrics such as monitored lane, occupancy, average space and flow rate of different types of vehicles. The analyst can select a specific device and time in the second area. The plot area then visualises time series which is similar with the plot area in the main panel but with more details using extra metrics as well

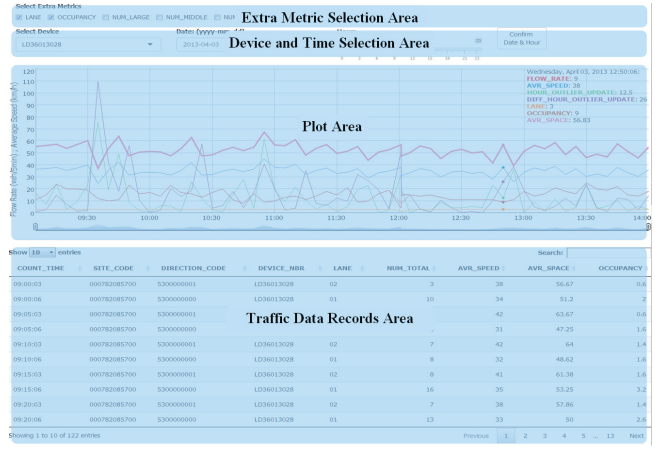

Figure B.25: Details panel. It contains four areas, including a detailed plot for the selected device and un-aggregated raw traffic statistic records. The plot is simplified from the main panel by removing accident time span, but with more metrics like lane, occupancy, average space, types of vehicles that are selected from Extra Metric Selection Area on the top. Traffic Data Records Area is zoomed in and shown in details in the next figure.

as the main panel's basic metrics. The bottom area shows raw traffic statistic records before aggregation which can be checked when there is suspicious device malfunctioning.

B.6 Conclusion and Future Work

In this research, we propose to use multi-metric data instead of commonly used single metric data for traffic analysis. We also introduce a general method to pre-process traffic data to be suitable for multivariate M-distance based algorithm. In this process, we introduce the importance of differential distance. Then we modify the algorithm to be updatable and describe the methodology to detect accident and correct and complement accident data. Finally, based on proposed algorithm, we develop a system with illustrative and interactive user interface to visualize different outliers in time domain and help to fix accident infor-

Show 5 entries Search:

COUNT_TIME	SITE_CODE	DIRECTION_CODE	DEVICE_NBR	LANE	NUM_TOTAL	AVR_SPEED	AVR_SPACE	OCCUPANCY
09:00:03	000782085700	5300000001	LD36013028	02	3	38	56.67	0.6
09:00:06	000782085700	5300000000	LD36013028	01	10	34	51.2	2
09:05:03	000782085700	5300000001	LD36013028	02	3	42	63.67	0.6
09:05:06	000782085700	5300000000	LD36013028	01	8	31	47.25	1.6
09:10:03	000782085700	5300000001	LD36013028	02	7	42	64	1.4

Showing 1 to 5 of 122 entries Previous 1 2 3 4 5 ... 25 Next

Figure B.26: Traffic Data Records Area. One record contains several fields like timestamp, ID (including numbering of site, device, direction and lane), flow rate, speed and occupancy. More fields such as statistics for different type of vehicles can be shown instead of ID's considering display space.

mation efficiently.

One issue should be concerned during data organisation, which is that the hourly datasets may cluster as ovals instead of eclipses. In that situation, evaluation of eclipse shape and appropriate transformation, such as logarithmic transformation or exponential transformation, are recommended. In addition, due to the lack of accurate flow-accident data, we are not able to statistically compare fluctuation estimation and accident detection performance with other distance types and algorithms. The usages and issues of proposed methodology and procedure will be investigated in future work, for instance traffic analysis and prediction using supervised learning.

Acknowledgements

We want to thank the editors of Technical Gazette for their efficient response, and reviewers for their time and contribution. We are thankful to get support from Florian Westphal with proof reading and advices.

We also thank authors and editors of referred papers for their important academic basis. Figure B.1 is reprinted by permission of Pearson Education, Inc., Upper Saddle River, New Jersey.

This work was supported by National Natural Science Foundation of China (NSFC), No. 61364019.

C

Anomaly-Aware Traffic Prediction Based on Automated Conditional Information Fusion

Bin Sun, Wei Cheng, Liyao Ma, Prashant Goswami. International Conference on Information Fusion (FUSION). IEEE: July 2018.

Abstract

Reliable and accurate short-term traffic prediction plays a key role in modern intelligent transportation systems (ITS) for achieving efficient traffic management and accident detection. Previous work has investigated this topic but lacks study on automated anomaly detection and conditional information fusion for ensemble methods. This works aims to improve prediction accuracy by fusing information considering different traffic conditions in ensemble methods. In addition to conditional information fusion, a day-week decomposition (DWD) method is introduced for preprocessing before anomaly detection. A *k*-nearest neighbours (*k*NN) based ensemble method is used as an example. Real-world data are used to test the proposed method with stratified ten-fold cross-validation. The results show that the proposed method with incident labels improves predictions up to 15.3% and the DWD en-

hanced anomaly-detection improves predictions up to 8.96%. Conditional information fusion improves ensemble prediction methods, especially for incident traffic. The proposed method works well with enhanced detections and the procedure is fully automated. The accurate predictions lead to more robust traffic control and routing systems.

Index terms— Intelligent Transportation Systems (ITS), Short-Term Traffic Prediction, Information Fusion, Time Series Decomposition, *k*-Nearest Neighbours (*k*NN)

C.1 Introduction

Intelligent transportation systems (ITS) are becoming more and more effective to avoid or resolve severe congestions and accidents [160, 161]. Reliable and accurate short-term traffic prediction is fundamental for modern ITS to achieve this target [2]. As a complex task, it has been studied in the past few decades using different schemes [4].

One widely used scheme is to fuse information according to different conditions to improve prediction accuracy [86, 87, 88, 89], However, finding the way to distinguish the traffic conditions is a problem as the related records for data labelling are messy. For messy labels or totally non-labelled data, some human resources may be needed such as labelling [118] or active learning [162]. In addition, there are different ways to categorize traffic conditions from either mathematical aspect or traffic engineering aspect and even more detailed aspects [163, 164, 165]. Normal vs. abnormal conditions are more suitable for a higher level analysis [166]. Though some previous methods are anomaly-aware, they are not automated ensemble methods and require manual parameter tuning, which are detailed in the next section. This work proposes the idea to predict traffic with the awareness of incidents or anomalies to enhance automated ensemble method. Besides, this work solves the issue of dynamic traffic fluctuation and multi-seasonality by introducing a day-week model for preprocessing before anomaly detection.

C.2 Related Work

C.2.1 Conditional Prediction

It has become consensus that different prediction methods or parameters should be used for different traffic conditions to improve accuracy, especially during extreme events [86]. For example, prediction-after-classification approaches [87] has been used, but it is considering one whole day as one data point without dynamic time series characteristics.

Many studies focus on choosing training data or nearest neighbours by clustering. For example, one work [88] found four different pattern changes, awareness of related patterns improves prediction accuracy. Theodorou et al. [89] considered typical vs. atypical conditions, but seasonal autoregressive integrated moving average (SARIMA) is their base method which is not accurate even with automated parameter tuning [116]. A flow-aware ensemble method weighted parameter tuples (WPT) is proposed targeting dynamic flow characteristics [116]. It has been compared with traditional time-aware predictions [105] and performs well. Ensemble methods have shown the ability to fuse information to improve prediction accuracy [126, 167] and to avoid manual parameter tuning such as WPT.

However, we have not seen any conditional ensemble methods with automated parameter tuning for short-term traffic prediction considering incidents or anomalies. Thus, based on the automated ensemble method WPT, this work proposes the idea to predict traffic with the awareness of incidents or anomalies.

C.2.2 Anomaly Detection

One group of anomaly detection methods are based on distribution, distance or density. This group includes traditional distribution based methods like dynamic Poisson distribution based detection [90], Chi-square test [91], and modern machine learning methods such as local

outlier factor [92, 93], one class support vector machine [94] and iForest [95]. However, distance/density-based methods require the time series should be transformed into points in a multidimensional space. This transformation loses the time series characteristics. Dynamic Poisson distribution uses more time-domain information, but it only detects local outliers.

Another way is to use regression methods [96] to build a model to get residuals and detect outliers according to a fixed or dynamic threshold. SARIMA [168] is widely used to build time series models. However, it is hard to automatically find suitable SARIMA parameters, and the performance is worse than kNN [36, 116]. Some automated versions of SARIMA [169] simply found "no suitable ARIMA model" for complex traffic data.

To get a better seasonality estimation, seasonal and trend decomposition with Loess (STL)-based methods are frequently used [170, 171]. One detection algorithm *TSoutliers* [170] uses STL and super smoother to build time series model and uses student-distribution based quantiles to detect outliers. This will detect additive outliers or pulses. Another recent algorithm is *SH-ESD* which is introduced by Twitter [172]. The basic algorithm is extreme studentized deviate test (ESD test) which is proposed by Roster [173]. The trend and seasonality should be removed before ESD. To remove it, seasonal ESD (S-ESD) [172] is using an improved version of STL [171] for time series decomposition. By using robust statistics such as median and median absolute deviation, S-ESD is improved to seasonal hybrid ESD (SH-ESD). SH-ESD has shown promising results [174, 175, 176] and is used as a base benchmark in this work together with TSoutliers.

For those regression-detection algorithms, one issue is that the dynamic traffic fluctuation is not considered. With higher traffic, comes higher fluctuation and deviation. Another issue is that the multi-seasonality characteristics in traffic time series are not considered. Thus, we propose to build a model for traffic by considering dynamic traffic fluctuation

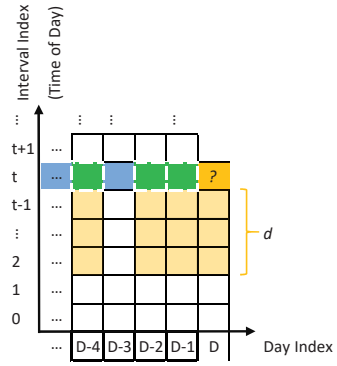

Figure C.1: Prediction for time t using basic kNN regression without window. History data last until time point $t-1$. Search step length (lag) is d.

and multi-seasonality.

C.3 Methodology

The proposed method improves ensemble kNN by conditional information fusion and enhanced anomaly detection.

C.3.1 kNN Regression for Prediction

The basic kNN regression algorithm for prediction is shown in Figure C.1. Suppose there is a traffic time series on time ticks: $\cdots, t-2, t-1, t, \cdots$ and we need to predict traffic flow at time t. The time interval between two neighbouring time ticks ranges from 1 minute to 15 minutes depending on different systems.

To predict the traffic at time t, two main steps are conducted. First, a state vector is constructed using the most recent data as a new query to represent current traffic state.

$$S_{[t]} = \begin{bmatrix} r_{t-1} & r_{t-2} & \cdots & r_{t-d} \\ s_{t-1} & s_{t-2} & \cdots & s_{t-d} \end{bmatrix} \tag{C.1}$$

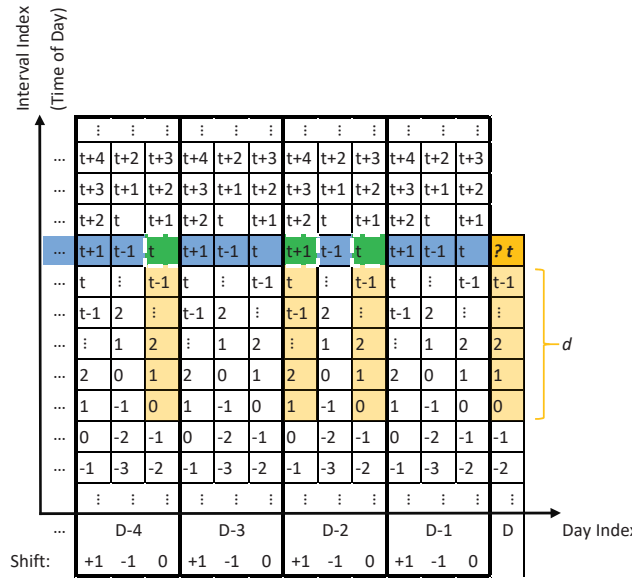

Figure C.2: Prediction for time t using kNN with window size $v = 1$ (shifts: -1,0,+1). History data last until time point $t - 1$. Search step length (lag) is d.

where r is flow rate and s is speed, d is search step length (lag), which means d data points from $t - 1$ until $t - d$ are used. Second, find the state vectors (nearest neighbours) from history data that are similar to current query state vector. For instance, in Figure C.1, state vectors from Day_{-1} (yesterday), Day_{-2} and Day_{-4} are the three nearest neighbours. Thus, the history values at time t of those three days are considered as neighbours' predictions and are averaged as the final prediction of traffic at time t of today.

This procedure has been improved by introducing window size (v), which is the maximum time shift during neighbour searching. If $v = 1$, all possible shifts are -1,0,+1. For instance, Figure C.2 shows $v = 1$ and the number of neighbours within each day increased to three. The last step is still averaging the values of all k nearest neighbours' predictions as the final prediction.

Now, there are three parameters in kNN regression to tune. The aforementioned method, WPT, is able to consider all three parameters at the same time and predict traffic automatically without manual parameter tuning. WPT is aware of the flow statistics, but not the flow conditions. The following content adds condition awareness to WPT and enhances the condition detection algorithms.

C.3.2 Conditional Information Fusion for kNN

As mentioned in Section C.2, WPT fuses information according to different flow rates. We modified it to fuse the information according to the flow conditions, i.e. normal vs. abnormal data, by not separating the data for different flow rates. This replaces the flow-awareness by anomaly-awareness. Abnormal history time points and normal data points are considered as two different groups. The time points in those two groups are used separately to train parameter tuples. Later, for the prediction of traffic flow, one of the two groups trained parameter tuples weights will be used respectively. Thus, the prediction is aware of normal vs. abnormal traffic.

C.3.3 DWD Preprocessing before Anomaly Detection

The purpose of preprocessing is to get normalized residuals to feed anomaly detection algorithms. The daily and weekly seasonality plays a key role in traffic engineering, so we propose a method named day-week-decomposition (DWD) to get rid of the daily and weekly seasonality. DWD will produce a *day-week model* which contains seven *day-of-week submodels*. The traffic flow data are averaged by a 15 min window to build the day-week model as it is the minimum time to get stable traffic flow [177]. To build the submodels, the data time points are divided into seven subsets according to their days of week. The median values of each time-of-day are used as one time point in a specific submodel. For example, if the time interval is 5 minutes, one day includes 288 time points. Each time point is the median value of the same time of day from one whole subset.

The basic idea to calculate residuals is subtracting the modelled values from the real values. Though for the same day of week, the traffic patterns are similar, the flow levels are different. Thus, for each day, the corresponding submodel is *scaled* and *shifted* to the day's data, hence *moved-submodels*. This transformation is done by robust fitting of nonlinear regression (*nlrob*) [178]. Consecutive zeros in this dataset are removed as they are from device malfunctioning or road closing and makes *nlrob* to fail. Then the residuals are calculated via subtracting the moved day-of-week submodels from the real values. Additionally, to smooth connections between neighbouring days during the scaling and shifting, the median occurring time of all the lowest points of the days are used as the beginnings and endings of each day. As higher flow rates produce higher fluctuations, the residuals are divided by their related standard deviations considering similar flow rates. As the residuals are derived from the data before 15 minutes smoothing, the residuals are smoothed. Finally, we get the normalized residuals.

Later, both the original data and the DWD generated data are sent to TSoutliers or SH-ESD to detect anomalies. Thus, the data are divided into two groups, normal data and abnormal data.

C.4 Experiments

C.4.1 Data Specification

The real world data are collected during April 2013 - May 2014 from a highway named Kunshi [118]. One device sends a monitored flow record at five-minute intervals. Each record contains some traffic statistics such as flow rate and average speed. This road carries undersaturated flow except in holidays noons.

Ground truth incident labels are generated by using the extended system mentioned in [118]. The data are imputed using the method from [179] before any processing.

C.4.2 Experimental Setup

The ensemble procedure is using the parameter generating rules from [10]. They developed an algorithm which is time-aware while this work develops an algorithm which is anomaly-aware. The values of k, d and v for the basic kNN are as follows: $\mathbb{K} = \{2, 4, 8, \cdots, 256\}$, $\mathbb{D} = \{2, 4, 8, \cdots, 256\}$, $\mathbb{V} = \{0, 4, 8, 16, 32\}$.

As the ensemble method consumes much time to run on a normal central processing unit (CPU), graphics processing unit (GPU) computing is used. The basic kNN experiments are conducted using CUDA [180] driver v8.0 and runtime v7.5 on a Nvidia Titan X Pascal. To get rid of the influence of missing value imputation, kNN ignores one neighbour if more than 10% steps in its searching steps or prediction steps contain incident data. The anomaly-awareness evaluation is done in R programming language version 3.4.3 and the results are analyzed in Python programming language version 3.5.2. TSoutliers is in R library *forecast* version 8.2 while SH-ESD is in *AnomalyDetection* version 1.0.

C.4.3 Evaluation and Measurement

Stratified ten-fold cross-validation (CV) is used for the evaluation of anomaly-awareness predictions according to detected conditions, i.e. anomaly vs. normality. To have fair results, the final measurement and analysis are done considering the ground truth incident labels, i.e. incident vs. non-incident. As the evaluation requires lables, only one dataset with enough ground truth incident labels is used while we did not find other large enough datasets with all labels or sufficient information for labelling.

Mean absolute error (MAE) is used to measure the performance of predictions. MAE is selected as it is frequently used in previous literature and it is easier for readers to compare with previous work. For instance, to measure the flow rate prediction results:

$$\mathrm{MAE} = \frac{\sum_{\delta=1}^{q} |\hat{r}_\delta - r_\delta|}{q} \tag{C.2}$$

where $\hat{r}$ is the predicted flow rate, r is the true flow rate, and q is the number of records. Predictions for missing values cannot be evaluated, and are excluded from evaluation.

C.5 Results and Analysis

C.5.1 Anomaly Detection Methods

The ground truth incident labels are used to evaluate anomaly detection performance. As shown in Table C.1, the DWD preprocessed methods are better on all aspects than the ones without DWD. DWD methods increase detection ratio (a.k.a true positive ratio, TPR) by 31% - 44%, and reduced false alarm ratio (a.k.a false positive ratio, FPR) by 69% - 74%. DWD methods improve F1-score by 84.3% compared to the other two methods. The corresponding normalized confusion matrices are shown in Figure C.3.

TABLE C.1
Detection performance measured by true positive ratio, false positive ratio, true negative ratio, false negative ratio, F1-score.

Method	TPR	FPR	TNR	FNR	F1
TSoutliers	28.235	1.973	98.027	71.765	0.281
SH-ESD	29.916	3.446	96.554	70.084	0.233
TSoutliers+DWD	40.538	0.522	99.478	59.462	0.507
SH-ESD+DWD	39.193	1.083	98.917	60.807	0.438

C.5.2 Anomaly-Aware vs. Unaware Predictions

Figure C.3: Normalized confusion matrices of anomaly detection methods. The anomaly detection results are evaluated using incident labels. All true positive and true negtive ratios are increased while all false alarms are decreased.

TABLE C.2
MAE of predictions. The unaware methods are similar and CV is robust. For incident data, the anomaly-awareness and DWD preprocessing are useful.

Awareness	Flow Condition	Detection Method	m=1	2	4	8
Aware	Incident	Ground Truth	12.662	13.944	15.822	17.548
		TSoutliers	15.983	16.476	17.281	18.258
		SH-ESD	14.598	15.329	16.626	17.950
		TSoutliers + DWD	13.705	14.695	16.205	17.766
		SH-ESD + DWD	13.611	14.575	16.172	17.723
	Non-Incident	Ground Truth	5.607	5.641	5.719	5.900
		TSoutliers	5.625	5.644	5.711	5.872
		SH-ESD	5.607	5.627	5.701	5.868
		TSoutliers + DWD	5.607	5.629	5.704	5.879
		SH-ESD + DWD	5.596	5.618	5.700	5.877
Unaware	Incident	Ground Truth	14.950	15.636	16.690	18.002
		TSoutliers	14.920	15.624	16.660	17.930
		SH-ESD	14.879	15.592	16.641	17.968
		TSoutliers + DWD	14.943	15.603	16.675	17.960
		SH-ESD + DWD	14.910	15.569	16.613	17.950
	Non-Incident	Ground Truth	5.607	5.632	5.702	5.878
		TSoutliers	5.606	5.630	5.701	5.877
		SH-ESD	5.606	5.630	5.703	5.880
		TSoutliers + DWD	5.607	5.629	5.704	5.879
		SH-ESD + DWD	5.607	5.629	5.700	5.873

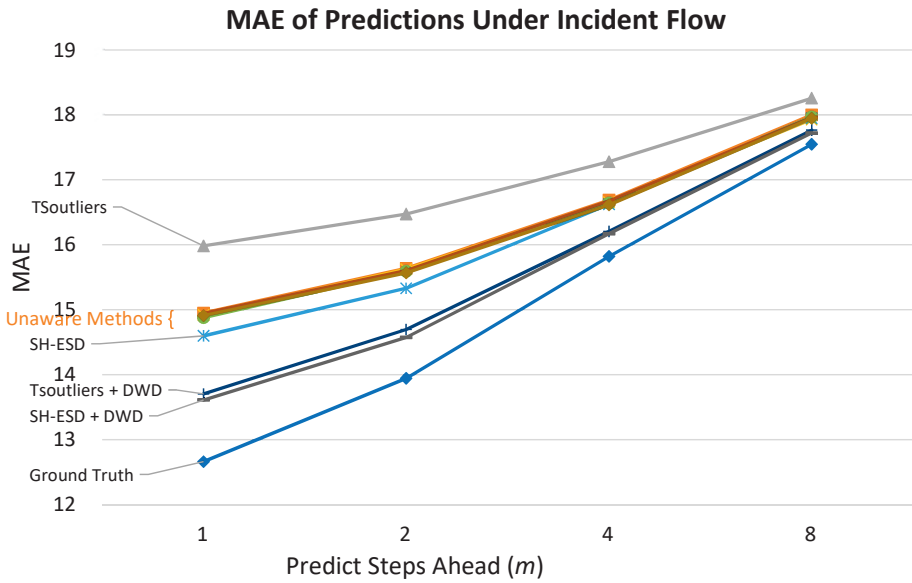

Figure C.4: Mean absolute error (MAE) of incident flow prediction. The incident-aware methods are distinguishable. The unaware methods are clustered together and not distinguishable which indicate cross-validated results are robust. The improvement of incident-aware methods with DWD are significant.

The prediction results are shown in Table C.2, there is no obvious difference among unaware results of different anomaly-unaware predictions. Different anomaly-unaware predictions are using different data divisions for 10-fold CV. They are similar and the maximum difference is no more than 0.48% compared to the ground truth. Thus, the CV results are robust.

The results patterns are easier to see in Figure C.4 and Figure C.5. Figure C.4 shows the prediction errors during incident traffic flow, while non-incident results are shown in Figure C.5. There are significant differences among aware methods during incident flow, but not during non-incident normal flow.

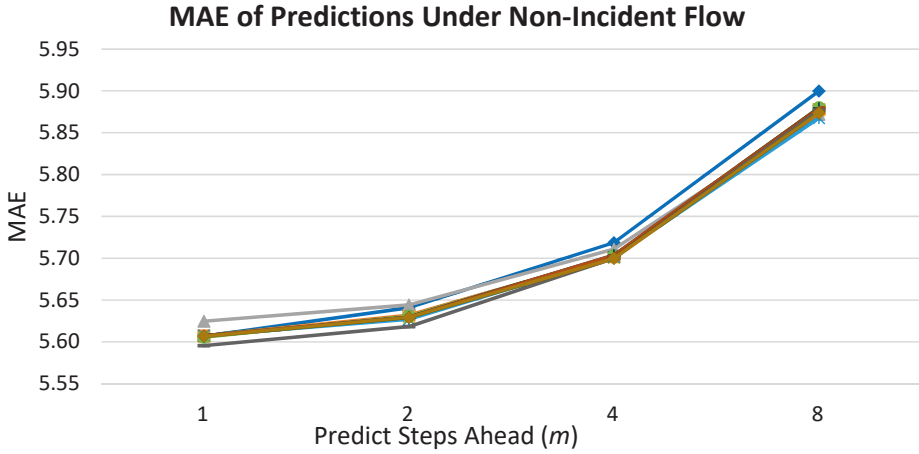

Figure C.5: MAE of non-incident flow prediction, both incident-aware and unaware. The differences among different methods are minor.

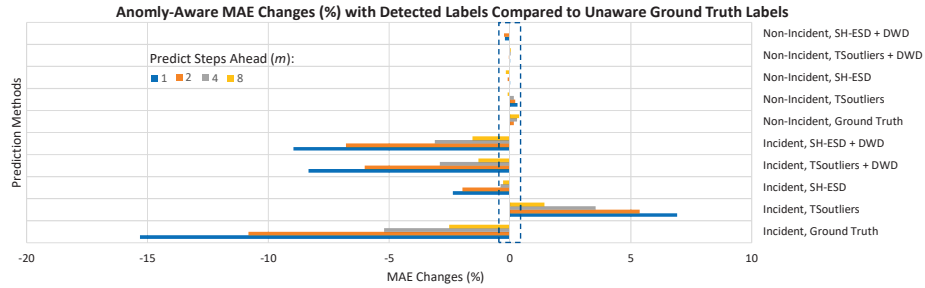

Figure C.6: Relative changes of anomaly-aware prediction MAE compared to ground truth unaware results. The changes during incident flow are significant, but not significant during non-incident flow. The values within the dash-circle is under the fluctuations of CV results (0.48%) and can be ignored. For incident flow, DWD based methods perform well and produce results similar to the ground truth. Results from other methods or conditions are not giving significant improvements, some even give worse results (higher MAE).

The relative changes of MAE are shown in Figure C.6 compared to ground truth unaware results. For non-incident data, the differences are ignorable, as the relative changes of MAE are minor considering the fluctuations of CV results (0.48%). For incident data prediction, the ground truth labels give the best results, up to 15.3% improvement, which is expected. The detection methods with DWD preprocessing preform well and reducing MAE by 4.87% on average and the maximum improvement is 8.96% for $m = 1$ by SH-ESD. The methods without DWD are giving less improvement (SH-ESD reduces MAE by only 2.36% during incident flow), or even increasing the prediction errors (TSoutliers increases MAE by 1.4%-6.9% during incident flow).

C.5.3 Analysis

One problem with the original methods (TSoutliers and SH-ESD) is that they treat high volume traffic flows during noon and holidays as abnormalities. It is due to the fact that they are not capturing daily and weekly seasonality correctly. Another problem is the high fluctuations and deviations during high traffic flow. DWD solves this problem by normalizing the residuals for different flow levels.

The results also motivate the improvement of anomaly detection algorithms. Improving the detection to be similar to the ground truth incident labels leads to incident-aware method and performs better. Currently, the detection ratio is not very high. While checking the results, two main reasons are found. First, some condition transition points as well as some points within incidents are similar to normal traffic flow volume and are not detected. Second, some minor incidents or the incidents that happened in other lanes are not detected.

The performance of anomaly detection algorithms influences the awareness prediction results significantly. The reason is that, removing information of time points that are from different traffic conditions leads to better fused results. For example, removing non-incident flow information when conducting incident flow prediction is actually removing

97.5% information, as only 2.5% data are incident data, according to the ground truth incident labels. However, removing incident flow information when conducting non-incident flow prediction is not giving significant improvements as the removed information is only 2.5%. Besides, the anomaly-aware methods require the underlying detection methods to perform well. If the detection algorithms are giving low accurate results, such as the ones without DWD, the prediction accuracy will be also low, even worse than the anomaly-unaware methods. This is due to information is fused into wrong conditions.

C.6 Conclusion and Discussion

The experiments results show that the proposed method of fusing information for traffic conditions separately to build different ensemble models can improve the prediction accuracy. Especially for incident conditions, the predictions are improved by 15.3%. The results are consistent with previous literature that different conditions should be treated separately for prediction. It also proves that our idea works well for fusing information in ensemble methods.

If no true labels are available to separate data conditions, anomaly detection is important. Our DWD enhanced methods act as alternatives to true labels and the predictions are improved by 8.96%. Although both TSoutliers and SH-ESD are supposed to be able to handle seasonal components, experiments show that our new DWD preprocessing provides better results. It increased detection ratio (TPR) by 31% - 44%, and reduced false alarm ratio (FPR) by 69% - 74%. The improvement comes from two reasons. First, the STL decomposition only removes one seasonality, while traffic data contains at least two main seasonality, daily and weekly. Second, the fluctuations and deviations on different flow levels are different, which are not considered in TSoutliers and SH-ESD.

Automation for anomaly detection is important due to the increasing amount of data. We also tried the classical SARIMA method. However, the automated SARIMA method from [170] cannot estimate the correct

parameters. Another alternative one [169] find "no suitable ARIMA model", which also indicate the problem of SARIMA parameter choices.

Prediction of coming flow conditions (incident vs. non-incident) is topic that covers a wide variety of alternative methods.We plan to investigate and improve the automated ones in the future work.

Acknowledgment

We would like to thank reviewers for their valuable suggestions. This work is supported by National Natural Science Foundation of China (NSFC) No. 61364019 and Shandong Provincial Natural Science Foundation No. ZR2018PF009.

D

Short-Term Traffic Forecasting Using Self-Adjusting *k*-Nearest Neighbours

Bin Sun, Wei Cheng, Prashant Goswami, Guohua Bai. IET Intelligent Transport Systems (ISSN: 1751-956X). 12(1), pp.41–48, February 2018.

Abstract

Short-term traffic forecasting is becoming more important in intelligent transportation systems. The *k*-nearest neighbours (*k*NN) method is widely used for short-term traffic forecasting. However, the self-adjustment of *k*NN parameters has been a problem due to dynamic traffic characteristics. This paper proposes a fully automatic dynamic procedure *k*NN (DP-*k*NN) that makes the *k*NN parameters self-adjustable and robust without predefined models or training for the parameters. A real-world dataset with more than one year traffic records is used to conduct experiments. The results show that DP-*k*NN can perform better than manually adjusted *k*NN and other benchmarking methods in terms of accuracy on average. This study also discusses the difference between holiday and workday traffic prediction as well as the usage of neighbour distance measurement.

D.1 Introduction

Increasing road traffic is nowadays causing more congestions and accidents which gain more attention from authorities and road users due to severe loss of life and property [161]. The intelligent transportation system (ITS) is expected to provide efficient traffic management and automatic accident detection [3]. The efficient control of traffic flow on motorways or freeways can lead to shorter travel time, fewer pollutant emissions, and increased road safety [1].

Efficient traffic management and accident detection rely on reliable and accurate short-term traffic forecasting [2]. Since the transportation system and traffic flow are results of the complex interplay among several aspects including the people, vehicles, roads, environment and information [24], predicting short-term traffic situation for ITS is a complex task, which has been an active research subject in the past few decades [4].

One frequently used method is *k*-Nearest Neighbour (*k*NN) [62] based prediction, but the self-adjustment of *k*NN parameters has been a problem due to dynamic traffic characteristics. The calibration or training of *k*NN parameters may discourage traffic management centres from using the algorithm [14]. We aim to make *k*NN parameters fully self-adjustable for traffic prediction to increase accuracy and to reduce human actions. We choose *k*NN because of the substantial increase in data availability [4] and its flexibility of solving nonlinear problems [36]. To have self-adjusting parameters, we develop dynamic procedure enhanced *k*NN (DP-*k*NN). DP-*k*NN is inspired by parameter space searching combined with *k*NN robustness [181]. When *k*NN parameters change, the prediction accuracy also changes. To get more accurate results, ensemble methods can be used to aggregate several prediction results [105]. The results show that DP-*k*NN can improve prediction accuracy compared to other methods, especially during holidays.

D.2 Related Work

Many studies have been conducted for short-term traffic forecasting. The existing methods can be divided into two categories which are parametric and non-parametric methods [4, 16, 18].

Typical parametric methods are algorithms designed using the seasonal autoregressive integrated moving average (SARIMA) [31]. Traditional subjective methods to choose values for SARIMA parameters include the usage of the autocorrelation function, partial autocorrelation function and extended autocorrelation function. Objective methods include Akaike information criterion and Bayesian information criterion, which can be used to automatize SARIMA [170]. Previous work has used SARIMA as benchmarking method but did not get the same conclusion when compared with *k*NN [58,64,71,182,183]. Thus, we compare automatised SARIMA with our proposed algorithm.

For non-parametric methods, most work focuses on neural networks [7, 8, 56, 57], as well as pattern searching [18, 58, 59, 60], besides other techniques, for example, Bayesian hierarchical model [184].

Within the pattern searching subcategory, one of the most popular algorithms is *k*NN. Some researchers attempted to improve this method for traffic forecasting from different aspects, such as spatial aspect [60], toll data aspect [66], complicated mathematical model aspect [67], among others.

D.2.1 Adaptive *k*NN Methods

One main parameter of *k*NN is the number of neighbours (k). If k is optimised, *k*NN can perform better [68]. There are many studies regarding optimisation of k. Ling et al. used *gap* of distance sequence to select neighbours where the gap implies the inherent boundary of a small region centred at new query instance. It is believed that the data within that region are densely gathered [69]. He et al. used an algorithm to adjust k from 1 to 10 according to the noise level (cache miss level)

during query [70], and more noise leads to bigger k. They investigated kNN from database performance aspect. Zhang and Song trained ANN to map from dataset characteristics to a suitable k value for classification [185]. Hong et al. optimised k by merging different values from similar sites and devices [65]. Dell et al. considered the flow with dynamic characteristics during different times of the day [64]. Singh et al. considered time series having piecewise linear nature and update k continuously [186]. There are also some studies that tried to optimise the selection of nearest neighbours from searching strategy aspect [187,188].

Though some research (including [71] and [189]) considered search step length (h) should be bigger than $2D + 1$ to be able to represent current state where D is number of input data dimensions. Other studies considered that it is possible to get best performance when $h < 2D + 1$ [190].

Some studies investigated both k and h. For example, [72] worked on both parameters, but the parameters are not optimised at the same time. [71] updated k periodically (daily, weekly, monthly) and they concluded that k and h should be optimised simultaneously. Some researchers used a part of the data as *estimation dataset* to evaluate the parameter combinations and used the best combination as the optimal settings [106]. This is part of their Pattern-kNN algorithm which overcomes memoryless issue in kNN regression. Pattern-kNN is similar to our idea and can handle missing data. Thus, it is used as one benchmarking algorithm in this work together with SARIMA.

For different traffic data, it is necessary to choose suitable parameter values differing from case-to-case and time-to-time. Three typical methods were used in previous work to make kNN adaptive. Those methods can have pre-defined models, for instance, holiday vs. workday and free flow vs. saturated flow, or calibrated/trained models [36, 182] or manual trial-and-error [64]. Based on the literature review, we found that fully automatic kNN parameter self-adjustment is lacking. Thus, we introduce a fully automatic dynamic procedure to choose parame-

ters and improve the robustness without extra information beyond the traffic data itself.

D.2.2 Distance Measurement

Distance measurement during neighbour selection for multi-variable should be considered. Among many distance measurements, Euclidean distance [117] is a widely used traditional and ordinary distance for finding nearest neighbours [72, 116, 143, 191]. It is easy to understand, implement and fast to calculate [145] but cannot represent a concept-related distance, as it does not consider the shape of distribution (scatter) [146]. Mahalanobis distance (M-distance) [108] shows advantages when handling multivariate problems. When compared with conventional thresholds, Figure D.1 from [147] shows that M-distance is suitable for multivariate outlier detection and can provide better thresholds by considering the shape of distribution. However, M-distance is rarely used in traffic data analysis. One possible reason is that few researchers are considering more than one metric together, so multivariate oriented M-distance is not necessary for previous work. Clark proposes to use multivariate distance for this, and they are using all the three fundamental traffic metrics (flow rate, speed, occupancy) with traditional Euclidean distance [1]. In this work, M-distance [108] based analysis is compared with Euclidean distance.

D.3 Methodology

In this section, we introduce the basic *k*NN algorithm and describe DP-*k*NN (abbreviated as DP-*k*NN) so that the parameters can be self-adjusting. The overview of algorithm logic is shown in Figure D.2. In summary, DP-*k*NN contains four stages. The preprocessing stage generates parameter options automatically according to the size of history data. The DP-*k*NN Level-1 evaluates all pairs of parameters using normal *k*NN. The second level evaluates combinations of the pairs and balances the weights between flow rate measurement and speed measurement. According to those evaluations, the final stage can use the

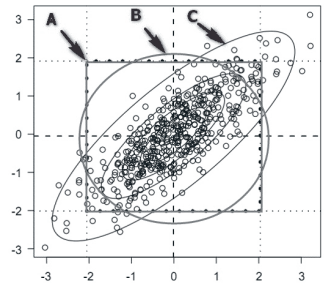

Figure D.1: Comparison of different distances. A: dotted square is the threshold considering each metric separately. B: threshold considering two metrics together. C: M-distance threshold considers two metrics together as well as the shape of the distribution. [147]

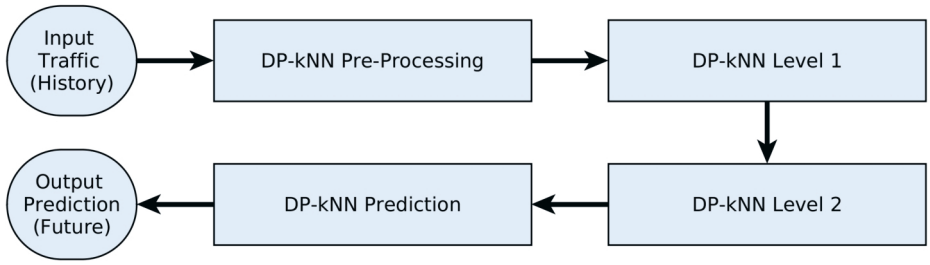

Figure D.2: DP-*k*NN algorithm logic overview. Every newly added record (time point) will trigger DP-*k*NN to go through those four stages. The preprocessing stage generates options for parameters which are evaluated in the first and second level. The prediction stage uses the evaluations to get final predictions.

optimal settings to conduct dynamic predictions. Level-1 and Level-2 need the parameters from preprocessing. Level-2 depends on the results from Level-1. The prediction stage needs to check evaluations from both Level-1 and Level-2. Later in this section, we present an application of M-distance measurement in multivariate difference traffic data.

D.3.1 Features and Metrics Selection

Flow rate, speed and density are three fundamental metrics in traffic engineering [24]. It is possible to calculate one metric given the other two, so two metrics should be considered at the same time. Previous research prefers to consider only one of them within time domain, usually flow rate or speed. We compare the usages of only flow rate with the usages of both flow rate and speed within time domain.

D.3.2 Basic kNN and Problem Settings

A basic kNN algorithm can be described as follows. Suppose that there is time series data until but not including now and the current time is $t-1$. New data arrives every time interval which differs among systems and it can be 1 minute, 5 minutes or 15 minutes and so on. The system interval is also the unit for search step length (h) and predict step length (m). If we consider one time interval as one basic time unit, the data is time series with data on time points: $\ldots, t-2, t-1, t, \ldots$. To predict the traffic, three steps are needed: constructing state vector, selecting nearest neighbours, conducting prediction according to the selected nearest neighbours' future (yellow parts after dashed squares in the figure).

Firstly, we select the latest data to construct state vector or matrix. Here we use h as the search step length which means h data points backwards from the current time $t-1$ until $t-h$ are selected as a new query in kNN algorithm. The step length is the number of intervals. The search step length is also known as a *window* in some research areas. To distinguish it from another definition when using shifting search step length, we will use the phrase *search step length* in this work. If traffic on time t is to be predicted, the state vector of flow rate is:

$$\boldsymbol{r}_{[t-1]} = (r_{t-1}, r_{t-2}, \ldots, r_{t-h}) \tag{D.1}$$

Similarly, the state vector for speed is:

$$\boldsymbol{s}_{[t-1]} = (s_{t-1}, s_{t-2}, \ldots, s_{t-h}) \tag{D.2}$$

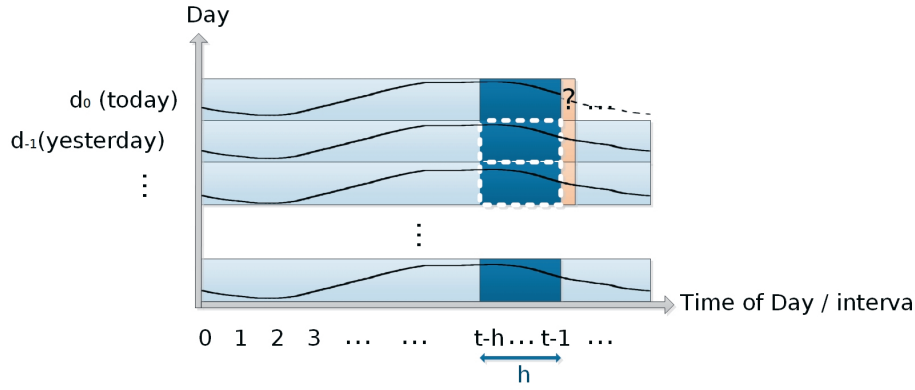

Figure D.3: Traffic prediction for time t using basic kNN. History data until and including the time point $t-1$. Search step length (lag) is h. Dashed white squares are selected nearest neighbours.

For the distance measurement which considers two variables, the state matrix is:

$$\begin{pmatrix} \boldsymbol{r}_{[t-1]} \\ \boldsymbol{s}_{[t-1]} \end{pmatrix} = \begin{pmatrix} r_{t-1} & r_{t-2} & \dots & r_{t-h} \\ s_{t-1} & s_{t-2} & \dots & s_{t-h} \end{pmatrix} \tag{D.3}$$

Secondly, we compare the latest data (state vector/matrix) with the same time range for every previous day, say time series from $t-h$ to $t-1$ of day d_{-1} (yesterday), from $t-h$ to $t-1$ of day d_{-2} (the day before yesterday), and so on. By comparison, we mean the calculation of distance between the latest data and previous each day's data. Therefore, the smallest distances of previous k days indicate the k nearest neighbours. For instance, in Figure D.3, shadowed part of time sequence from d_{-1} and d_{-2} are selected as nearest neighbours.

Finally, the averaged result of the values from the data at time points t of all the nearest neighbours' corresponding days is calculated. The result is the prediction. The whole basic kNN algorithm is shown in Figure D.3

Although the basic kNN can be used, traffic flow has varied and non-

stable statistic characteristics, so kNN needs different k and h pairs under different situations. As we can see later that manual adjusting is hard to give the best solution. DP-kNN provides a way to fully automatically adjust k and h and their ensemble results for the system.

D.3.3 DP-kNN Preprocessing

When traffic data is provided to DP-kNN, or a new record is appended to existing data, the preprocessing phase firstly generates the values' options to be assigned to DP-kNN parameters. The principles and procedure of generating possible values of parameters are as follows. Most parameters are directly or indirectly dependent on the number of history instances (m_1), exponential growth is suitable. For the number of neighbours k, it starts with $k = 2$ which grows exponentially with power two (double the value every time), until k can cover half of potential neighbours. For the search step length h, it starts with $h = 2$ which grows exponentially with power two until h can cover half of a day. Then the set of p ($\mathbb{P}$) is generated as all possible pairs of k and h, i.e., Cartesian product. On the second level of DP-kNN, for the number of p to choose (k'), it starts with $k' = 2$ which grows exponentially with power two, until k' can cover half of the items in $\mathbb{P}$. For the speed error weighting w', as its range is from 0 to 1 and it is not related to the number of history instances, an arithmetic sequence 0, 0.2 ..., 1 is used. Those principles are trying to cover bigger parameter space with less samples. Beyond the parameters, for the ensemble and aggregation of previous evaluations, n_1 and n_2 should cover the last one hour. The reason to cover one hour is as below. Short-term prediction is to predict future traffic no more than one hour [192], so traffic data can be divided hourly and hypothesis testing is used to check differences among hourly traffic datasets. Previous work has shown that, according to the p-values, one hour before a time point is the maximum safe range to get similar traffic [118].

D.3.4 DP-*k*NN Level 1

The first level of DP-*k*NN evaluates different pairs of *k*NN parameters. Another *k*NN procedure (the second level) is used to decide which parameter pairs from the first level should be, and how to be, aggregated.

In order to predict traffic at time t, we will find several best (k, h) pairs according to the recent-history situation, and use the averaged results of those pairs to reduce the influence of variance. The content below explains how to predict flow rate and speed at time t.

We have two sets here which contain available values of k and h: $\mathbb{K}$ which contains N_K values and $\mathbb{H}$ which contains N_H values. We need a measurement of how well the k and h pairs work. Firstly, there is a set of the pairs:

$$\mathbb{P} = \mathbb{K} \times \mathbb{H} = \{(k_{i_k}, h_{i_h}) | k_{i_k} \in \mathbb{K}; h_{i_h} \in \mathbb{H}\} \tag{D.4}$$

where $i_k = 1, 2, \ldots, N_K; i_h = 1, 2, \ldots, N_H$ and $\mathbb{P}$ contains $N_P = N_K \cdot N_H$ pairs of k and h.

For each $p_{i_p} \in \mathbb{P}, (i_p = 1, 2, \ldots, N_P)$, we use the paired values to set up *k*NN and do prediction. After getting the predicted data, we can measure flow rate prediction error (ε_R) for each p_{i_p} at time t:

$$\varepsilon_{R[t-1]i_p} = f(\hat{r}_{[t-1]}(p_{i_p}), r_{[t-1]}) \tag{D.5}$$

where $\hat{r}_{[t-1]}(p_{i_p})$ is the predicted flow rate at time $t-1$ using p_{i_p} and f is prediction error measurement function. The same measurement is used for speed prediction error (ε_S):

$$\varepsilon_{S[t-1]i_p} = f(\hat{s}_{[t-1]}(p_{i_p}), s_{[t-1]}) \tag{D.6}$$

where $\hat{s}_{[t-1]}(p_{i_p})$ is the predicted speed at time $t-1$ using p_{i_p}. A typical error measurement used is mean absolute error (MAE), which is used here. Another widely used measurement is mean absolute percentage error (MAPE) which is not always suitable because flow rate can sometimes be zero, especially during nights. The measurement of flow rate

prediction error from time $t+1$ till time $t+q$ (q instances) is:

$$\text{MAE} = \frac{\sum_{\delta=1}^{q} \left| \hat{r}_{[t+\delta]} - r_{[t+\delta]} \right|}{q} \tag{D.7}$$

where $\hat{r}$ is the predicted flow rate, and r is the real flow rate respectively.

Then we calculate the averaged measurements of each p_{i_p} for the latest n_1 intervals for flow rate and speed:

$$\overline{\varepsilon_{R[t-1]}}_{i_p} = \frac{1}{n_1} \sum_{\delta=1}^{n_1} \varepsilon_{R[t-\delta]i_p} \tag{D.8}$$

$$\overline{\varepsilon_{S[t-1]}}_{i_p} = \frac{1}{n_1} \sum_{\delta=1}^{n_1} \varepsilon_{S[t-\delta]i_p} \tag{D.9}$$

where n_1 is the number of instances per hour.

Now we construct a matrix using p, $\overline{\varepsilon_{R[t-1]}}$, $\overline{\varepsilon_{S[t-1]}}$ as Level-1 DP-kNN evaluation result matrix (before normalisation):

$$\hat{\boldsymbol{E}}_{[t-1]} = \begin{pmatrix} p_1 & \overline{\varepsilon_{R[t-1]}}_1 & \overline{\varepsilon_{S[t-1]}}_1 \\ p_2 & \overline{\varepsilon_{R[t-1]}}_2 & \overline{\varepsilon_{S[t-1]}}_2 \\ \vdots & \vdots & \vdots \\ p_{i_p} & \overline{\varepsilon_{R[t-1]}}_{i_p} & \overline{\varepsilon_{S[t-1]}}_{i_p} \\ \vdots & \vdots & \vdots \\ p_{N_P} & \overline{\varepsilon_{R[t-1]}}_{N_P} & \overline{\varepsilon_{S[t-1]}}_{N_P} \end{pmatrix} \tag{D.10}$$

To reduce the influence of random error, we will use several different p's to make the prediction at time t and calculate the average. However, the new challenge now is determining how many of pairs (p) to choose (i.e. to choose the value of k') and define the goodness metric of p.

We will firstly describe normalisation for flow rate and speed, then introduce weights (w') to have a weighted measurement. To have the

fair influence from both flow rate and speed, we need to unify the measurements (normalisation) in $\hat{\boldsymbol{E}}_{[t-1]}$:

$$u = \frac{\text{mean}(\text{col}_2\hat{\boldsymbol{E}}_{[t-1]})}{\text{mean}(\text{col}_3\hat{\boldsymbol{E}}_{[t-1]})} \tag{D.11}$$

$$\begin{aligned}\boldsymbol{E}_{[t-1]} &= \left(\text{col}_1\hat{\boldsymbol{E}}_{[t-1]},\quad \text{col}_2\hat{\boldsymbol{E}}_{[t-1]},\quad u\cdot\text{col}_3\hat{\boldsymbol{E}}_{[t-1]}\right)\\ &= \begin{pmatrix} p_1 & \overline{\varepsilon_{R[t-1]}}_1 & u\,\overline{\varepsilon_{S[t-1]}}_1\\ p_2 & \overline{\varepsilon_{R[t-1]}}_2 & u\,\overline{\varepsilon_{S[t-1]}}_2\\ \vdots & \vdots & u\vdots\\ p_{i_p} & \overline{\varepsilon_{R[t-1]}}_{i_p} & u\,\overline{\varepsilon_{S[t-1]}}_{i_p}\\ \vdots & \vdots & u\vdots\\ p_{N_P} & \overline{\varepsilon_{R[t-1]}}_{N_P} & u\,\overline{\varepsilon_{S[t-1]}}_{N_P}\end{pmatrix}\end{aligned} \tag{D.12}$$

where u is unifying factor, $\boldsymbol{E}_{[t-1]}$ is unified $\hat{\boldsymbol{E}}_{[t-1]}$, $\text{col}_1\hat{\boldsymbol{E}}_{[t-1]}$ is the first column of $\hat{\boldsymbol{E}}_{[t-1]}$ and so on. $\boldsymbol{E}_{[t-1]}$ is the final Level-1 DP-kNN evaluation result matrix with normalisation.

Now, two new parameters compose the second layer of our DP-kNN of kNN algorithm which are k' and w'.

D.3.5 DP-kNN Level 2

For notation, suppose possible values of k' and w' are in two sets: $\mathbb{K}'$ which contains $N_{K'}$ values and $\mathbb{W}'$ which contains $N_{W'}$ values, then we can define a set containing all (k', w') pairs as:

$$\mathbb{P}' = \mathbb{K}' \times \mathbb{W}' = \{(k'_{i_{k'}}, w'_{i_{w'}}) | k'_{i_{k'}} \in \mathbb{K}'; w'_{i_{w'}} \in \mathbb{W}'\} \tag{D.13}$$

where $i_{k'} = 1, 2, \ldots, N_{K'}, i_{w'} = 1, 2, \ldots, N_{W'}$ and $\mathbb{P}'$ contains $N_{P'} = N_{K'} \cdot N_{W'}$ pairs of (k', w').

Given a specific time point $(t-1)$, similarly like $\mathbb{P}$, we measure the performance of each $p'_{i_{p'}} \in \mathbb{P}', (i_{p'} = 1, 2, \ldots, N_{P'})$.

The indexes of w', i.e. i'_w, are repeated in $\mathbb{P}'$, but it is needed to distinguish each pair's w' from other pairs. Thus, for a specific pair $p'_{i_{p'}}$, consider the corresponding specific index of w' as $i_{p'}$. Similar to $w'_{i_{p'}}$, for the specific pair $p'_{i_{p'}}$, the corresponding specific index of k' is also considered as $i_{p'}$, hence $k'_{i_{p'}}$. Thus, $\mathbb{P}'$ and its elements can also be noted as:

$$\mathbb{P}' = \{(k'_{i_{p'}}, w'_{i_{p'}}) | i_{p'} = 1, 2, \ldots, N_P\} \tag{D.14}$$

After noting those elements, the evaluation of $p'_{i_{p'}}$ begins. For time point $t-1$, we add weighted measurement as a new column into $\boldsymbol{E}_{[t-1]}$, which becomes:

$$\boldsymbol{E}^{+}{}_{[t-1]} = \left(\boldsymbol{E}_{[t-1]}, \quad (1 - w'_{i_{p'}}) \cdot \mathrm{col}_2 \boldsymbol{E}_{[t-1]} + w'_{i_{p'}} \cdot \mathrm{col}_3 \boldsymbol{E}_{[t-1]}\right) \tag{D.15}$$

The rows in $\boldsymbol{E}^{+}{}_{[t-1]}$ is then sorted by the last column (i.e. weighted measurement) in increasing order. Mathematically speaking, a row permutation is applied to $\boldsymbol{E}^{+}{}_{[t-1]}$ so that the values in the last column are monotonically non-strict increasing when the index i_p is increasing. For the sake of convenience, the symbolic representation of the matrix ($\boldsymbol{E}^{+}{}_{[t-1]}$) is not changed. After permutation, use each p from the 1st row until $k'_{i_{p'}}{}^{\text{th}}$ row for basic kNN parameters to predict flow rate and speed. The results of those p are averaged to get predictions noted with $p'_{i_{p'}}$: $\hat{r}'_{[t]}(p'_{i_{p'}})$ and $\hat{s}'_{[t]}(p'_{i_{p'}})$. Thus, the measurements of DP-kNN Level-2 flow rate prediction error (ε'_R) and speed prediction error (ε'_S) of pair $p'_{i_{p'}}$ i.e. $(k'_{i_{p'}}, w'_{i_{p'}})$ at time $t-1$ are:

$$\varepsilon'_{R[t-1]i_{p'}} = f(\hat{r}'_{[t-1]}(p'_{i_{p'}}), r_{[t-1]}) \tag{D.16}$$

$$\varepsilon'_{S[t-1]i_{p'}} = f(\hat{s}'_{[t-1]}(p'_{i_{p'}}), s_{[t-1]}) \tag{D.17}$$

Now, we will choose the best p' (pair of k' and w'). Similar to the first level of DP-kNN, we need to calculate the average performance of each $p'_{i_{p'}} \in \mathbb{P}'$ for the last n_2 intervals:

$$\overline{\varepsilon'_{R[t-1]_{i_{p'}}}} = \frac{1}{n_2}\sum_{\delta=1}^{n_2}\varepsilon'_{R[t-\delta]i_{p'}} \tag{D.18}$$

$$\overline{\varepsilon'_{S[t-1]_{i_{p'}}}} = \frac{1}{n_2}\sum_{\delta=1}^{n_2}\varepsilon'_{S[t-\delta]i_{p'}} \tag{D.19}$$

We can consider p', $\overline{\varepsilon'_{R[t-1]_{i_{p'}}}}$, $\overline{\varepsilon'_{S[t-1]_{i_{p'}}}}$ as columns to construct Level-2 DP-*k*NN evaluation result matrix:

$$\boldsymbol{E'}_{[t-1]} = \begin{pmatrix} p'_1 & \overline{\varepsilon'_{R[t-1]_1}} & \overline{\varepsilon'_{S[t-1]_1}} \\ p'_2 & \overline{\varepsilon'_{R[t-1]_2}} & \overline{\varepsilon'_{S[t-1]_2}} \\ \vdots & \vdots & \vdots \\ p'_{i_{p'}} & \overline{\varepsilon'_{R[t-1]_{i_{p'}}}} & \overline{\varepsilon'_{S[t-1]_{i_{p'}}}} \\ \vdots & \vdots & \vdots \\ p'_{N_{p'}} & \overline{\varepsilon'_{R[t-1]_{N_{p'}}}} & \overline{\varepsilon'_{S[t-1]_{N_{p'}}}} \end{pmatrix} \tag{D.20}$$

This gives us the evaluation of all $p'_{i_{p'}}$. The Level-2 evaluation result ($\boldsymbol{E'}_{[t-1]}$) can then be used together with Level-1 evaluation result ($\boldsymbol{E}_{[t-1]}$) for parameter selections during prediction.

D.3.6 DP-*k*NN Prediction

To predict the flow rate at time t, the following steps are taken.

$\boldsymbol{E'}_{[t-1]}$ is sorted by the last column, i.e., a row permutation is applied to $\boldsymbol{E'}_{[t-1]}$ so that the values in the column $\mathrm{col}_2\boldsymbol{E'}_{[t-1]}$ (flow rate prediction error) are monotonically non-strict increasing. The fist row of the sorted $\boldsymbol{E'}_{[t-1]}$ indicates the least prediction error. The p' in the 1st row is selected as the best choice of p' and noted as p'^*. The corresponding k' and w' in p'^* are the best choices and noted as k'^* and w'^*. By using w'^* to weight columns $\mathrm{col}_2\boldsymbol{E}$ and $\mathrm{col}_3\boldsymbol{E}$, a new column is added into $\boldsymbol{E}_{[t-1]}$ to become

$\boldsymbol{E^*}_{[t-1]}$.

$$\boldsymbol{E^*}_{[t-1]} = \left(\boldsymbol{E}_{[t-1]}\,, \quad (1 - w'^*) \cdot \mathrm{col}_2\boldsymbol{E}_{[t-1]} + w'^* \cdot \mathrm{col}_3\boldsymbol{E}_{[t-1]}\right) \tag{D.21}$$

Lastly, after sorting $\boldsymbol{E^*}_{[t-1]}$, each p (i.e., k, h pair) from the 1st row until the k'^* th row are used to predict flow rate at time t. By averaging the results, we get final prediction of flow rate.

For predicting the speed at time t, similar procedure is used. The only difference is applying a row permutation to $\boldsymbol{E'}_{[t-1]}$ so that the values in the last column (speed prediction error) are monotonically non-strict increasing.

D.3.7 Difference Mahalanobis Distance

As there are many ways of applying M-distance, here is a brief description of M-distance and how it is used with difference data. Suppose i $(1 \leq i \leq m_1)$ is instance index and j $(1 \leq j \leq m_2)$ is variable index in dataset $\boldsymbol{X}$ (with elements x_{ij}) that contains m_1 observations of m_2 variables. The covariance between the j_1th variable and the j_2th variable is:

$$\sigma_{j_1 j_2} = \frac{1}{m_1 - 1} \sum_{i=1}^{m_1} (x_{ij_1} - \mu_{j_1})(x_{ij_2} - \mu_{j_2}) \tag{D.22}$$

where μ_{j_1} and μ_{j_2} are variables' expected values respectively.

Thus the covariance matrix of dataset $\boldsymbol{X}$ can be expressed as a m_2-by-m_2 matrix $\boldsymbol{\Sigma}$ (with element σ_{lm}), $(1 \leq l \leq m_2, 1 \leq m \leq m_2)$. This is calculated from history data.

Finally, M-distance to centroid is the distance between instance $\boldsymbol{x_i}$ and centroid $\boldsymbol{\mu}$:

$$MD_i = \sqrt{(\boldsymbol{x_i} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\boldsymbol{x_i} - \boldsymbol{\mu})^T} \tag{D.23}$$

Although M-distance can also be used to measure distance between any two points, it stands for *M-distance to centroid* in our work.

A nature of traffic time sequence is the difference characteristic. This is noticed in [193] and they also used first derivative of speed but in a different way. We can get one difference data from two consecutive instances. That is, the difference data is a result of subtracting one data instance with its previous neighbour in a consecutive time series.

If we note an arbitrary instance with both flow rate and speed as $\boldsymbol{x_i} = (r_i, s_i)$, we can get the difference data as:

$$\dot{\boldsymbol{x}}_{\boldsymbol{i}} = (\dot{r}_i, \dot{s}_i) = \boldsymbol{x_{i+1}} - \boldsymbol{x_i} = (r_{i+1} - r_i\ , s_{i+1} - s_i) \tag{D.24}$$

As there are originally m_1 instances in $\boldsymbol{X}$, we can construct difference dataset ($\dot{\boldsymbol{X}}$) which have $m_1 - 1$ difference instances ($\dot{\boldsymbol{x}}_{\boldsymbol{i}}$). Though the covariance matrix is calculated hourly, the algorithm does not need to know the exact time of day. Thus, the calculation is still automatic.

One problem during distance measurement is the curse of dimension which is caused by the sparse sampling space due to the big value of h [120]. To avoid this problem, we treat the time domain as one dimension instead of multi-dimension by averaging the distance values from all search step. Thus, the final distance of state vector (or matrix) at time t when considering an arbitrary day d is:

$$\frac{\sum_{i_h=1}^{h}(MD_{[t-i_h,d]})}{h} \tag{D.25}$$

where MD is M-distance. This equation is also applied to other distance measurements.

The whole DP-*k*NN algorithm is illustrated in the pseudocode Algorithm 1 (see Figure D.4) which can be considered as a detailed description of Figure D.2. Note that the cache is a part of random access memory (RAM) that stores many different matrices and values. The time

procedure DP
  automatically generate $\mathbb{K}, \mathbb{H}, \mathbb{P}, \mathbb{K}', \mathbb{W}', \mathbb{P}'$ ▷ Start DP-kNN Preprocessing
  initialise all measurements with 0
  while given a new instance $(r_{[t-1]}, s_{[t-1]})$ **do**
    append the new instance to history data
    delete the earliest history instance (if needed)
    update $\mathbb{K}, \mathbb{H}, \mathbb{P}, \mathbb{K}', \mathbb{W}', \mathbb{P}'$ according to generating rules (if needed)
    for each $p_{i_p} \in \mathbb{P}$ **do** ▷ Start DP-kNN Level-1
      $\hat{r}_{[t]}(p_{i_p})$ and $\hat{s}_{[t]}(p_{i_p}) \leftarrow$ normal kNN prediction using p_{i_p}
      save $\hat{r}_{[t]}(p_{i_p})$ and $\hat{s}_{[t]}(p_{i_p})$ to cache for later usages
      get predicted $\hat{r}_{[t-1]}(p_{i_p})$ and $\hat{s}_{[t-1]}(p_{i_p})$ from the cache
      calculate $\varepsilon_{R[t-1]i_p}$ and $\varepsilon_{S[t-1]i_p}$, then $\overline{\varepsilon_{R[t-1]}}_{i_p}$ and $\overline{\varepsilon_{S[t-1]}}_{i_p}$
    end for
    construct $\hat{\mathbf{E}}_{[t-1]}$, then $\mathbf{E}_{[t-1]}$
    for each $p'_{i_{p'}} \in \mathbb{P}'$ **do** ▷ Start DP-kNN Level-2
      $\mathbf{E}^{+}_{[t-1]} \leftarrow$ apply $w'_{i_{p'}}$ to $\mathbf{E}_{[t-1]}$
      sort($\mathbf{E}^{+}_{[t-1]}$)
      a list of selected p (i.e., k, h pairs) (number of pairs: $k'_{i_{p'}}$) $\leftarrow$ select 1st to $k'_{i_{p'}}$ th rows from $\mathrm{col}_1\mathbf{E}^{+}_{[t-1]}$
      a list of prediction ($\hat{r}_{[t]}(p)$, $\hat{s}_{[t]}(p)$) $\leftarrow$ get cached Level-1 predictions w.r.t p
      $\hat{r}'_{[t]}(p'_{i_{p'}})$ and $\hat{s}'_{[t]}(p'_{i_{p'}}) \leftarrow$ average from the list of ($\hat{r}_{[t]}(p)$, $\hat{s}_{[t]}(p)$)
      save $\hat{r}'_{[t]}(p'_{i_{p'}})$ and $\hat{s}'_{[t]}(p'_{i_{p'}})$ to the cache for later usages
      get predicted $\hat{r}'_{[t-1]}(p'_{i_{p'}})$ and $\hat{s}'_{[t-1]}(p'_{i_{p'}})$ from the cache
      calculate $\varepsilon'_{R[t-1]i_{p'}}$ and $\varepsilon'_{S[t-1]i_{p'}}$, then $\overline{\varepsilon'_{R[t-1]}}_{i_{p'}}$ and $\overline{\varepsilon'_{S[t-1]}}_{i_{p'}}$
    end for
    construct $\mathbf{E}'_{[t-1]}$
    $p'^{*} \leftarrow$ sort($\mathbf{E}'_{[t-1]}$) and select the p' from the 1st row ▷ Start DP-kNN Prediction
    final prediction for time t: use p'^{*} to get $\hat{r}'_{[t]}(p'^{*})$ and $\hat{s}'_{[t]}(p'^{*})$ from the cache
  end while
end procedure

Figure D.4: Algorithm 1 DP-*k*NN.

point indicator (subscripts) should be checked carefully during implementation. Recent measurement and evaluation results are temporarily cached so that DP-*k*NN does not need to recalculate them. Due to the usage of cache, the pseudocode for implementation looks a little different to the mathematical description, but the logic is the same.

D.4 Experiments

D.4.1 Data Collection

The data is collected by the traffic management centre of Yunnan province, China. The data from April 2013 to May 2014 of one monitoring device is cleaned using a tool from [118]. The road usually carries under-saturated flow except for holiday noons.

A new data record is generated every five minutes, i.e., the interval is five minutes, and one predict step ($m = 1$) is equivalent to five minutes. Each record contains an identifier and some statistically measured values. Among those attributes, we select the total number of vehicles passing the monitoring point (*flow rate*) and the average speed of those vehicles (*speed*) as mentioned in Section D.3.1.

D.4.2 Experimental Design

During neighbour searching, any neighbour is discarded if it has an overlap with query instance's search step length or predict step length or any possible evaluation results. Thus, the error is out-of-sample test error. Besides, if more than ten percent data in search or prediction steps are missing or influenced by events according to authority's information, the results are ignored during analysis.

As SARIMA processes and predicts one-dimension data, to compare with DP-*k*NN, we use two types of Euclidean distances. The phrase *Euclidean distance*, in our case, is used for prediction using only flow rate (the errors becomes absolute values), and *E2d* for prediction using

both flow rate and speed. As DP-kNN is using one-hour data to find parameter settings, the same data is used as *estimation set* (as defined in [106]) for Pattern-kNN.

The experiment environment for the preprocessing stage is Matlab R2016b. DP-kNN Level-1 is conducted in CUDA version 8.0 (runtime version 7.5) [180] on a Graphics Processing Unit (GPU), Nvidia Titan X Pascal. The second level of DP-kNN and experiment result analysis is done using R programming language version 3.3.3 and R-studio software version 1.0.143 running in Windows 10 on Intel Core i7-6850K with 32GB memory.

D.5 Results

We plot the results for holidays and workdays separately only for analysis purpose because they have different characteristics. As the results contain quite a lot of data, part of results are plotted. Only flow rate prediction is shown because speed prediction results have similar trends and SARIMA has only flow rate output. To plot two-dimension figures, $k = 8$ and $h = 4$ is used while changing another parameter. This can be considered as slicing of the multi-dimension results data. The patterns of data beyond those plots are not changing much unless otherwise mentioned. The parameters in DP-kNN are not fixed but self-adjusted and aggregated, so the values are displayed as straight horizontal lines.

According to the generating principles in DP-kNN preprocessing, the potential values for parameters are automatically generated as: $\mathbb{K}$ = {2, 4, 8, ..., 256}, $\mathbb{H}$ = {2, 4, 8, ..., 256}. $\mathbb{P}$ contains $N_P = 64$ pairs of (k, h), so $\mathbb{K}'$ = {2, 4, 8, 16, 32}, $\mathbb{W}'$ = {0, 0.2, 0.4, 0.6, 0.8, 1}. Thus, $\mathbb{P}'$ contains $N_{P'} = 30$ pairs of (k', w'). In addition to the parameters, $n_1 = n_2 = 12$ is used to cover one hour as the interval of data is five minutes.

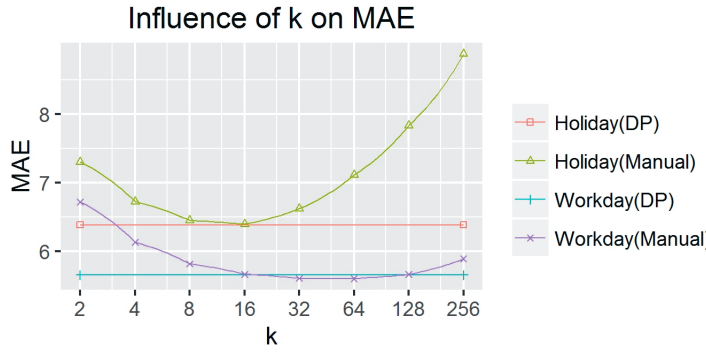

Figure D.5: Influence of number of nearest neighbours k on performance (mean absolute error), when search step length $h = 4$ (20 minutes). Two horizontal lines are DP-kNN results from self-adjust parameters. The horizontal lines are near the minimum of manually adjusted kNN curves. In this plot, DP-kNN is below all the MAE values on the manually adjusted kNN's error curve on holidays. This means the manual adjustment of k cannot be better than DP-kNN due to the very small h.

D.5.1 Influence of k and h

The plotted search step length h is 4, which means that the search time is 20 minutes. The plots below, Figure D.5 is showing the comparison between DP-kNN and manual tuned kNN on holidays and workdays. The distance is E2d and other distances have similar patterns. On holidays, DP-kNN performs well and can touch the curve's minimum or below the minimum depending on h. On workdays, DP-kNN is also near the minimum, but cannot go below the curve's minimum. When compared with holiday, working day's manual tuning curve has a higher k value of turning point.

The influence of search step length h is shown in Figure D.6. Patterns of influence of h are similar to k, and the workday curve also has a higher value of turning point.

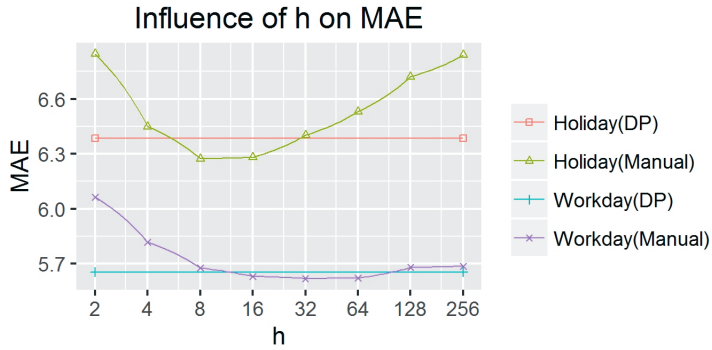

Figure D.6: Influence of search step length h on performance (error) with the number of nearest neighbours $k = 8$. One step ($h = 1$) is equivalent to 5 minutes. Two horizontal lines are DP-kNN results from self-adjusting parameters. The horizontal lines are near the minimums of manually adjusted kNN curves.

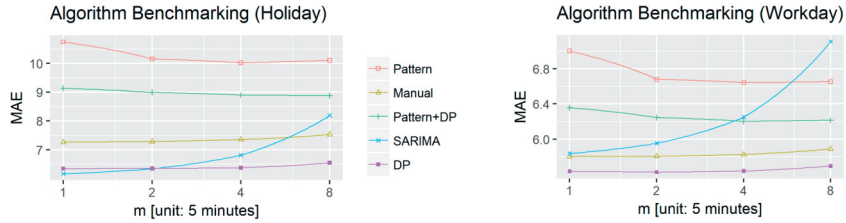

Figure D.7: Algorithm benchmarking results. One predict step ($m = 1$) is equivalent to 5 minutes. Pattern-kNN is providing less accurate results for most situations. SARIMA performs similar or worse than Manual-kNN and is not stable. On workdays, DP-kNN always gives the highest accuracy. kNN-based algorithms are more stable than SARIMA in both situations.

D.5.2 Benchmarking

DP-*k*NN is compared with the benchmarking algorithms. DP-*k*NN, Pattern-*k*NN, Pattern+DP-*k*NN, and SARIMA have automatic self-adjusted parameters. As SARIMA uses only one variable (flow rate), Euclidean distance based DP-*k*NN is shown here instead of E2d.

The left plot in Figure D.7 shows results on holidays. Pattern-*k*NN gives less accuracy. Applying DP to Pattern-*k*NN can improve the accuracy by 12%. Averaged results from Manual-*k*NN are giving a similar incremental pattern to DP-*k*NN, but are 15% less accurate. SARIMA initially gives a low error than DP-*k*NN for predict step $m = 1$ and 2, i.e., 5 to 10 minutes. Though SARIMA performs slightly better at the beginning, its error increases quickly from 6.17 to 8.19 (33% increment) when m becomes bigger, which makes SARIMA 7% to 25% worse than DP-*k*NN. The MAE of DP-*k*NN only increases from 6.37 to 6.55 (3% increment).

On workdays, DP-*k*NN gives the most accurate results than others, as shown in Figure D.7, right side plot. Pattern-*k*NN and SARIMA give the highest error. Applying DP to Pattern-*k*NN can improve the accuracy by 7%. SARIMA's error increases from 5.84 to 7.11 (22% increment) when m becomes bigger while DP-*k*NN stays less than 5.7 (1% increment). SARIMA is initially 4% worse than DP-*k*NN and then it becomes 25% worse when the predict step length increases from 1 to 4. Manual-*k*NN has a similar incremental pattern to DP-*k*NN, but 3% less accurate.

D.5.3 Neighbour Distance Measurement

There are three different metrics used in DP-*k*NN to measure the distance between the query instance and potential nearest neighbours. When compared with Mahalanobis, Euclidean and E2d distances are traditional and widely used.

The results on holidays show that Euclidean and E2d are performing similar while Mahalanobis is giving higher error as shown in Figure D.8,

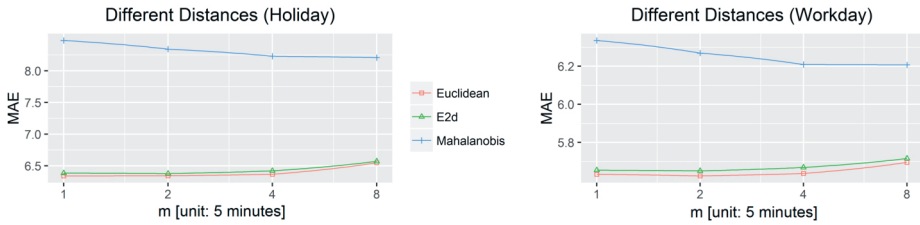

Figure D.8: The performance of distances in DP-kNN. One predict step ($m = 1$) is 5 minutes. On holidays, Euclidean and E2d are similar, while Mahalanobis's error is higher. On workdays, all three distances have lower MAE compared with holidays, but Mahalanobis still has a higher error than Euclidean and E2d.

left. The MAE of Euclidean and E2d are increasing a little when m increases. However, Mahalanobis gives less error for larger values of m. On workdays, Mahalanobis gives smaller MAE than holidays. Euclidean and E2d have the same pattern as shown in Figure D.8, right.

D.6 Analysis and Discussion

Even though the algorithms are working adaptively without the knowledge of day being holiday or not, the analysis here is done separately due to different patterns in the results. We found that the holiday prediction error usually decreases first and then increases when either number of neighbours k or search step length h becomes bigger and then increases. This means that there are turning points for both k and h. On the first level of DP-kNN, when k or h is too small, MAE varies a lot and has many fluctuations at one arbitrary time. Thus, k and h should not be too small. Longer search step length is better on workdays, because the number of workdays is more than holidays.

When compared DP-kNN with SARIMA, DP-kNN is robust and stable on both holidays and workdays when m increases, while SARIMA is not. In holidays, SARIMA can be slightly better when m is small, but its performance drops quickly and becomes much worse than DP-kNN. On workdays, DP-kNN is always better and SARIMA still gives as much as

24% more error. If we extend the value ranges for Pattern-*k*NN parameters k and h by using DP preprocessing results ($\mathbb{K}, \mathbb{H}, \mathbb{P}$), Pattern-*k*NN is 12% better on holidays and 4% better on workdays. Though Pattern-*k*NN was showing good performance in literature, it is not showing obvious advantages here. The reason could be that the previous literature was using one day data to test [106], while this work uses more than one year data. Another possible reason is that the manual-*k*NN method that has been compared with Pattern-*k*NN in [106] is from [63] and were using too small values for k and h when compared with recent literature. One more reason could be that the traffic flow in this paper contains more fluctuations and noise than the flow in [106] and it could be harder for Pattern-*k*NN to recognise patterns. The above discussion shows that the choice of *k*NN parameters plays an important role and we should look more at the parameters.

DP-*k*NN sometimes gives decreasing MAE when m increases. The reason could be that no weights are assigned on distance calculation of neighbours, i.e., the very recent traffic is not given extra focus. Though M-distance has shown good performance previously, it does not perform well in this work. Hence, it should be carefully analysed and optimised before application.

D.6.1 Complexity and Efficiency

Suppose the number of history instances is m_1, the time complexity of Algorithm 1 is $O(m_1^2)$. If the earliest history instances are being discarded when adding new instances, m_1 becomes a static value and the complexity becomes $O(1)$.

DP-*k*NN Level-1 is running on the Nvidia GPU, and each new instance costs less than 14 milliseconds. The time is not changing as the history data size is fixed. DP-*k*NN Level-2 and prediction are running on the Intel CPU and it costs 144 milliseconds to measure previous predictions and get final predictions for next time point. As the data instances come every 5 minutes, DP-*k*NN Level-1 can process data from 22 thou-

sand monitoring devices and Level-2 can handle more than two thousand devices. With the bottleneck on Level-2, DP-*k*NN can deal with 2 thousand devices simultaneously. The total delay from receiving a new instance to get prediction is 158 milliseconds. However, the two levels of DP-*k*NN are asynchronous so they should be considered separately when considering the bottleneck.

D.6.2 More about Holidays and Workdays

Both Manual-*k*NN and DP-*k*NN give better results for workday traffic prediction because there are more workdays which means more similar data. However, dynamic algorithms have more difficulties to catch the minimums of manually adjusted *k*NN curves on holidays. The reason could be that the parameter space contains different fluctuation styles on workdays when compared with holidays. We have found that the evaluation results of parameter space fluctuate more on workdays than holidays.

D.7 Conclusion and Future Work

In this paper, we successfully modified *k*NN to DP-*k*NN with fully automatic self-adjustment for the parameters without calibration or training. The results show that DP-*k*NN gives 9% to 40% improvement than benchmarking methods on average when considering both holidays and workdays. It usually performs the best, with few exceptional situations on holidays. Accurate predictions lead to more confident traffic management decisions and can be used to detect accidents more correctly. In addition, DP-*k*NN is a general methodology that can be applied to similar algorithms and systems which are in need of self-adjustable parameters.

This paper focuses on the number of nearest neighbours k and the search step length h. Even though some other parameters like window size are potentially adjustable, they have not been considered here. The type of days (holidays vs. workdays) matters, but how to make use

of the differences from the traffic flow itself without manual work is a challenge. A better way to handle the missing data, such as imputation [179], should be considered. These topics will be investigated in our future work.

Acknowledgements

We thank all anonymous reviewers for their thorough review and detailed comments which have significantly improved the quality of this paper. We owe great thanks to Niklas Lavesson and Bengt Aspvall for constructive comments and suggestions. We are also thankful to get support from Siva Krishna Dasari and Florian Westphal for proof reading and advice.

This work was supported by National Natural Science Foundation of China (NSFC), under grant agreement number 61364019. Figure D.1 is reprinted from publication [147] with permission from Elsevier, license number 4045330384711.

E

Flow-Aware WPT *k*-Nearest Neighbours Regression for Short-Term Traffic Prediction

Bin Sun, Wei Cheng, Prashant Goswami, Guohua Bai. ACM International Conference Proceeding Series ICITT/ICSET. ACM: October 2017.

Abstract

Robust and accurate traffic prediction is critical in modern intelligent transportation systems (ITS). One widely used method for short-term traffic prediction is *k*-nearest neighbours (*k*NN). However, choosing the right parameter values for *k*NN is problematic. Although many studies have investigated this problem, they did not consider all parameters of *k*NN at the same time. This paper aims to improve *k*NN prediction accuracy by tuning all parameters simultaneously concerning dynamic traffic characteristics. We propose weighted parameter tuples (WPT) to calculate weighted average dynamically according to flow rate. Comprehensive experiments are conducted on one-year real-world data. The results show that flow-aware WPT *k*NN performs better than manually tuned *k*NN as well as benchmark methods such as extreme gradient boosting (XGB) and seasonal autoregressive integrated moving

average (SARIMA). Thus, it is recommended to use dynamic parameters regarding traffic flow and to consider all parameters at the same time.

Index terms— Flow-Aware, Weighted Parameter Tuples, k-Nearest Neighbours Regression, Short-Term Traffic Prediction

E.1 Introduction

Increasing road traffic is nowadays causing more congestion and accidents which gain more attention from public and authorities due to severe loss of life and property [160, 161]. Efficient traffic management and automatic accident detection are key requirements in modern intelligent transportation systems (ITS). Reliable and accurate short-term traffic forecasting is necessary for achieving efficient traffic management and accident detection [2]. Predicting short-term traffic is a complex task, which has been a research subject of many studies in the past few decades [4].

The existing short-term traffic forecasting methods can be divided into two categories which are parametric and non-parametric methods [4, 16, 18]. A typical parametric method is seasonal autoregressive integrated moving average (SARIMA) [31]. Moving average considers that the near-future data is similar to the latest data and uses a weighted average of latest data as predictions. An autoregressive model is based on interdependent observations of stationary time series. Values of one stationary time series are in a range with a constant variance considering existing data and future data with a constant average. Within a stationary series, interdependency between history and future can be used to make a prediction. If the original time series is non-stationary, an integrated part is needed in the model to make it stationary by conduct differencing. Traffic data usually meet the peak during noon time and bottom after midnight. Besides, weekends have a different traffic scenario when being compared with workdays. Those patterns are modelled as seasonal part in SARIMA.

Within the non-parametric category, decision trees have been widely used. Gradient boosting is one way to improve decision trees. It improves tree models by focusing on badly predicted instances iteratively. Extreme gradient boosting (XGB) is a fast and regularised gradient boosting implementation with higher calculation speed and more robustness against overfitting. As a state-of-the-art method, XGB has shown outstanding performance and efficiency which is comparable with or better than random forest [51,52]. Thus, XGB is used as a benchmark method in this study.

Another important non-parametric algorithm is *k*-Nearest Neighbours (*k*NN) [194]. Both classification and regression tasks can be handled using *k*NN. For time series regression, the key idea in *k*NN is to find history data with similar patterns of most recent data. This paper uses *k*NN because of the substantial increase in data availability [4], the flexibility of *k*NN for solving non-linear problems and easiness of understanding and implementation [55]. Some studies show that *k*NN is better than traditional methods (such as Kalman filter and SARIMA) [71,183]. However, some others reported that *k*NN has similar performance with traditional methods [58,64]. Thus, SARIMA is also used as a benchmark method in this study.

Three parameters of *k*NN are the number of nearest neighbours (k), search step length (d) (also known as lag) and window size (v) (also known as constraint) [58]. Though distance measurement of neighbours can also be considered as a dynamic parameter, it is beyond this paper's scope and will be addressed separately. Besides, when m (number of predict steps ahead) changes, the parameters should also be tuned. Many studies tried to tune the parameter k and some work also tried to tune d while few researchers considered v.

Previous work has focused on some of the parameters, but the value assignment for those three parameters at the same time is still a problem. On one aspect, we propose to use weighted parameter tuples (WPT) to improve *k*NN by considering all parameters together. On another

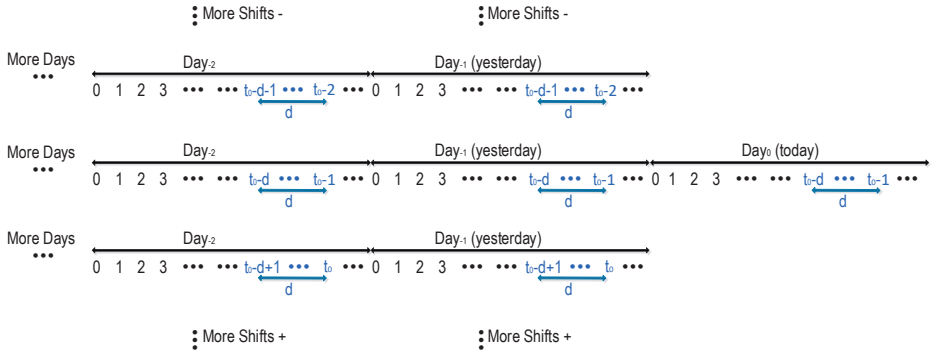

Figure E.1: Prediction for time t using kNN with window size $v = 1$ (shifts: -1,0,+1). History data last until time point $t - 1$. Search step length (lag) is d.

aspect, we focus on flow-aware parameter tuning, while previous studies focused on improving kNN from temporal aspect (time-aware) [52, 64], such as the hour of day or day of week etc.

E.2 Background and Problem Settings

This section contains some background definitions and formulate the problem in this paper mathematically.

E.2.1 kNN Regression for Prediction

Suppose there is a time series with data on time points: $\cdots, t-2, t-1, t, \cdots$ where t is the near-future to predict. The time interval can differ from system to system and is usually between 1 minute and 15 minutes.

The following steps are employed to predict the traffic at time t as shown in Figure E.1.

Step one is to construct state vector as the query to represent current traffic state by selecting the latest data. If we choose search step length

as d, the state vector of current flow rate is:

$$\boldsymbol{r}_{[t]} = [r_{t-1}, r_{t-2}, \cdots, r_{t-d}] \tag{E.1}$$

while the state vector of current speed is:

$$\boldsymbol{s}_{[t]} = [s_{t-1}, s_{t-2}, \cdots, s_{t-d}] \tag{E.2}$$

For the distance measurement with two variables, the state vector is:

$$\boldsymbol{S}_{[t]} = \begin{bmatrix} r_{t-1} & r_{t-2} & \cdots & r_{t-d} \\ s_{t-1} & s_{t-2} & \cdots & s_{t-d} \end{bmatrix} \tag{E.3}$$

The search step length is sometimes referred as "window" in some areas, though we have another kNN parameter named window. To distinguish them, the phrase "search step length" is used in this work.

Step two is to find the most similar vectors (nearest neighbours) of history days when being compared with current query state vector. For instance, data from time point $t-d$ to $t-1$ of yesterday (Day_{-1}), from $t-d$ to $t-1$ of Day_{-2} etc. are the neighbours to compare.

In kNN, window size (constraint) (v) is used to describe the maximum time point shift when searching for neighbours. If $v = 2$, the maximum shift of time point is two, therefore, all possible shifts are -2,-1,0,+1,+2. For instance, Figure E.1 is showing situation with $v = 1$ and the number of neighbours within yesterday is three instead of one. The time points of neighbours in yesterday are: $[t_0 - d - 1, \cdots, t_0 - 2]$ (with shift -1), $[t_0 - d, \cdots, t_0 - 1]$ (with shift 0) and $[t_0 - d + 1, \cdots, t_0]$ (with shift +1). Thus, if there are n days to search, then there are $n \times (2v + 1)$ neighbours to search to find k nearest neighbours.

Finally, the average value of all k nearest neighbours' predictions is calculated as the final prediction.

E.2.2 Mathematical Problem Settings

The three critical parameters of kNN include the number of neighbours k, search step length d and window size v. Suitable values should be as-

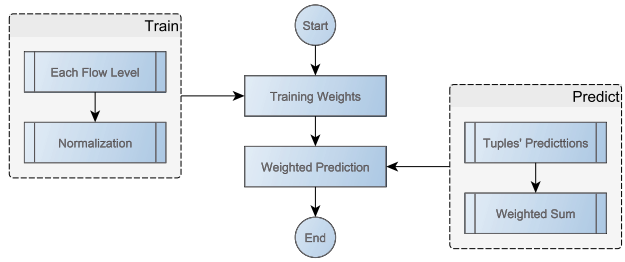

Figure E.2: Overview of the proposed WPT algorithm.

signed to them. If k has M_K options, d has M_H and v has M_V options, the possible number of tuples of those parameters is huge: $M_K \cdot M_H \cdot M_V$. When three parameters are considered together, a legitimate combination is called a *tuple* which is a 3-tuple in this work. For instance, (k = 8, d = 4, v = 0) is a tuple that can be used by the three kNN parameters.

E.3 Methodology

The kNN regression algorithm is introduced in the previous section. This section describes how the usage of WPT can enhance kNN and how the parameter-tuples' predictions can be employed to generate a weighted average. Later on, we explain how the weights are generated and present several weighting methods to assign weights.

The general idea is to conduct predictions using different values of (k, d, v) values for all training time points, and the performance of tuples can be measured. According to the performance, weighting functions generate weights for each configuration of (k, d, v). Later, selected (k, d, v) tuples are used to make prediction and a weighted average is calculated as the final result.

An overview of the proposed WPT algorithm is shown in Figure E.2.

E.3.1 Traffic Metrics Selection

Traffic engineering contains three fundamental metrics which are flow rate, speed and density [24]. Given two metrics, it is just enough to calculate the remaining metric. Most previous studies use only one of them, which is usually flow rate, sometimes speed. This work uses both flow rate and speed.

E.3.2 Training Weights

The following content explains how to train weights and to predict traffic using all parameter tuples.

We conduct multi steps ahead prediction using kNN with all given parameter values for each training time point. A tuple set ($\mathbb{P}$) is a Cartesian product of three sets which contains values of k, d and v: $\mathbb{K}$ contains M_K values, $\mathbb{D}$ contains M_D values and $\mathbb{V}$ contains M_V values. The performance of each tuple (k, d, v) is measured. The set of the tuples to be measured is:

$$\begin{gathered}\mathbb{P} = \mathbb{K} \times \mathbb{D} \times \mathbb{V} = \\ \{(k_{i_k}, h_{i_d}, v_{i_v}) | k_{i_k} \in \mathbb{K}; h_{i_d} \in \mathbb{D}; v_{i_v} \in \mathbb{V}\}\end{gathered} \tag{E.4}$$

where $i_k = 1, 2, \cdots, M_K; i_d = 1, 2, \cdots, M_D; i_v = 1, 2, \cdots, M_V$ and $\mathbb{P}$ contains $M_P = M_K \cdot M_D \cdot M_V$ tuples of k, d and v.

For each $p_{i_p} \in \mathbb{P}, (i_p = 1, 2, \cdots, M_P)$, three values are used to set up kNN and make prediction. Later, prediction error (ε) is measured. Error of flow rate prediction for each p_{i_p} is measured as $\varepsilon(r_{[t]})_{i_p} = f_e(\hat{r}_{[t]}(p_{i_p}), r_{[t]})$ where $\hat{r}_{[t]}(p_{i_p})$ is the predicted flow rate at t using p_{i_p} and f_e is the function to measure prediction error. For error of speed prediction, the same measurement is used as $\varepsilon(s_{[t]})_{i_p} = f_e(\hat{s}_{[t]}(p_{i_p}), s_{[t]})$ where $\hat{s}_{[t]}(p_{i_p})$ is the predicted speed at t using p_{i_p}.

Flow-Aware Weights

Previous research has shown that the traffic has time-variant dynamic characteristics in the different hour of day, day of the week, holiday

versus non-holiday etc. [52, 64]. However, if records are separated by time of day, it is not guaranteed that the flow situation is the same, neither the day of week, etc. Instead, records are separated by flow rate levels. A practical method is to separate the flow into ten levels equally from the lowest flow rate to the highest flow rate. For instance, for a road with peak flow of one hundred cars per time unit and zero during night, from 0 to 10 will be the lowest level while from 90 to 100 is the highest level. WPT adapts the weights of tuples to the flow according to separated flow levels. The present flow rate is determined by averaging flow rate of the last 15 minutes, as it is the minimum time to have stable traffic flow [177].

Weighting Function

It is worth mentioning that the weights are for tuples, not for neighbours or search steps (lags). *Rank-based weighting* generates scores according to the ranks of candidates when sorted according to increasing order of distances from the subject profile [35]. Previous work shows that rank-based weighting is better than inverse weighting [195]. When using weight dispersion measure of 1, the score generated by rank-based weighting (c_{rnk}) is as below (Equation E.5).

$$c_{rnk} = M_P - \gamma + 1 \tag{E.5}$$

where γ is the rank of the candidate ($\gamma = 1, 2, \cdots, M_P$). The candidate with $\gamma = 1$ has the lowest ε. The scores for all p's form an arithmetic sequence: $[M_P, M_P - 1, \cdots, 1]$.

Each p_i is tested and corresponding scores are calculated and added to the p_i's total score. The tuples with highest scores (e.g., top 25%) are used on each flow level. Finally, a normalisation procedure is conducted for weights to make sure the sum of weights is 1 for each flow level. An easy-to-understand logic will be presented in experimental design later. The final weights are as below:

$$\boldsymbol{W} = [w_1, w_2, \cdots, w_{M_P}] \tag{E.6}$$

E.3.3 Predicting using Weights

Instead of assigning weights to nearest neighbours, WPT assigns them to the parameter tuples (p_i).

To predict the flow rate, the following steps are taken.

Firstly, all tuples of (k, d, v) in $\mathbb{P}$ are used to calculate predictions. The predicted flow rate values are noted as below (Equation E.7):

$$\hat{\boldsymbol{R}} = [\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_{i_p}, \cdots \hat{r}_{M_P}]; i_p = 1, 2, \cdots, M_P \tag{E.7}$$

As the weights are $\boldsymbol{W}$, the weighted flow rate prediction is:

$$\hat{r} = \boldsymbol{W} \cdot \hat{\boldsymbol{R}} \tag{E.8}$$

To predict the speed, a similar procedure is used.

Distance of Neighbours

Euclidean distance (ED) is a widely used traditional and ordinary distance in transportation data for finding nearest neighbours [72]. It is easy to understand, implement and fast to calculate [196].

Here is a brief description of ED. Suppose $i(1 \leq i \leq n_1)$ is instance index and $j(1 \leq j \leq n_2)$ is variable index in dataset $\boldsymbol{X} = [[x_{ij}]]$ that contains n_1 observations of n_2 variables.

ED_i is the distance between query instance $\boldsymbol{x_i}$ and a neighbour instance $\boldsymbol{x_i'}$. Both instances have d intervals. As:

$$\boldsymbol{x_i} = \begin{bmatrix} \boldsymbol{r}_{[t]} \\ \boldsymbol{s}_{[t]} \end{bmatrix}; \boldsymbol{x_i'} = \begin{bmatrix} \boldsymbol{r'}_{[t]} \\ \boldsymbol{s'}_{[t]} \end{bmatrix} \tag{E.9}$$

$$ED_i = |\boldsymbol{x_i} - \boldsymbol{x_i'}| = \frac{\sum_{i_d=1}^{d} \sqrt{(r_{t-i_d} - r'_{t-i_d})^2 + (s_{t-i_d} - s'_{t-i_d})^2}}{d} \tag{E.10}$$

Big values of d may cause the curse of dimension problem for distance measurement, which is caused by the sparse space [120]. To avoid this problem, we consider all search steps in the time domain as one dimension instead of d dimensions. This is done by averaging the distances of all search steps. Some other types of distances are available, for example, Mahalanobis distance [65, 197], which is beyond this paper's scope.

E.4 Experiments

E.4.1 Data Specification

The real world data is collected by a traffic management centre who deployed dozens of devices along a highway named Kunshi. Each device sends one statistical record at five-minute intervals. Each record contains a timestamp and some statistical values such as flow rate and average speed. Part of the data from one monitoring device on the road is used. The time range of data is from April 2013 to May 2014. This road is under-saturated except during holidays.

E.4.2 Experimental Design

The beginning 80% data is used as training data to train weights, the remaining 20% is left as test data. To get rid of the influence of incidents, kNN ignores one neighbour if more than 10% steps in its searching steps or prediction steps contain incident data.

For the values of k, d and v, exponential incremental values are used as follows: $\mathbb{K} = \{2, 4, 8, \cdots, 256\}$, $\mathbb{D} = \{2, 4, 8, \cdots, 256\}$, $\mathbb{V} = \{0, 4, 8, 16, 32\}$. v starts from 0 so that the results can be compared with

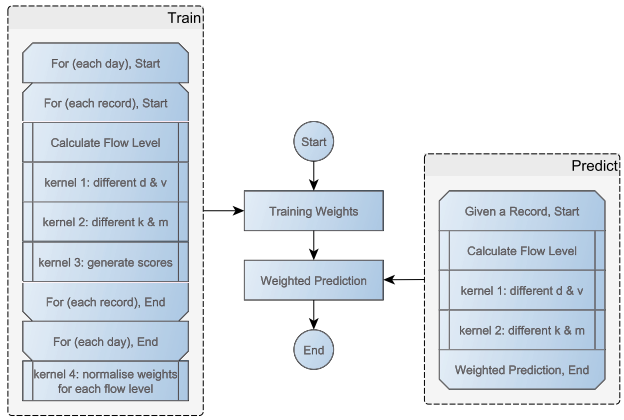

Figure E.3: WPT logic diagram with GPU kernels.

previous studies which did not use window. As 256 days is more than half, which is supposed to be big enough for k when v is zero. For d, 256 is the maximum value because one day contains $288 \cdot 5$ minutes. For v, the values start from 0 to compare with general no window situation. Thus, $\mathbb{P}$ contains $M_P = 320$ pairs of k, d and v.

To solve the huge calculation load, graphics processing unit (GPU) is used to reduce calculation time especially for training stage. GPU makes it practical to conduct a huge amount of calculations by accelerating them significantly. The detailed algorithm logic diagram with GPU kernels is shown in Figure E.3. The analysis of one year data using non-optimized R implementation would consume more than a year to conduct experiments on a central processing unit (CPU). GPU reduces this computational time to hours. As the original traffic data is only around one megabyte, it is divided into smaller chunks which are copied into shared memory from global memory and shared by threads. This further accelerates calculations. Four kernels are developed for the experiments and the first two kernels are used in both training and predicting stage.

The experiment is conducted using CUDA [180] driver v8.0 and run-

time v7.5 on GeForce 690 card. The analysis environment is R programming language [198] v3.3. XGB implementation is from R library *xgboost* v0.4.4. One parameter of XGB should be set, which is the number of iterations. In this paper, 16 can produce the best results. SARIMA implementation is from R library *forecast* v7.2 in which a fully automatic and objective SARIMA parameter value assignment is implemented.

E.4.3 Performance Measurement (f_e)

To measure the performance of prediction, mean absolute error (MAE) is used as f_e. For the flow rate prediction measurement:

$$\text{MAE} = \frac{\sum_{\delta=1}^{q} |\hat{r}_\delta - r_\delta|}{q} \tag{E.11}$$

where $\hat{r}$ is the predicted flow rate, r is the grant truth flow rate, and q is the number of records. Mean absolute percentage error (MAPE) is not used because the flow rate is sometimes zero after midnight before morning.

As mentioned before, the records are separated to ten dataset $\boldsymbol{X'}$ ($\boldsymbol{x_i}$) according to ten evenly separated flow levels. Each flow level contains about ninety thousand records in average for training, which are used to generate scores and weights.

E.5 Results

This section firstly provides results regarding the impact of flow rate. Later comes the comparison of different benchmark methods (XGB and SARIMA) in terms of the accuracy.

E.5.1 Impact of Flow Rate

The weighted averages of parameter values are shown in Figure E.4. Though WPT uses weights to calculate prediction, the weights can also be applied to parameter options to analyse how flow impact parameters.

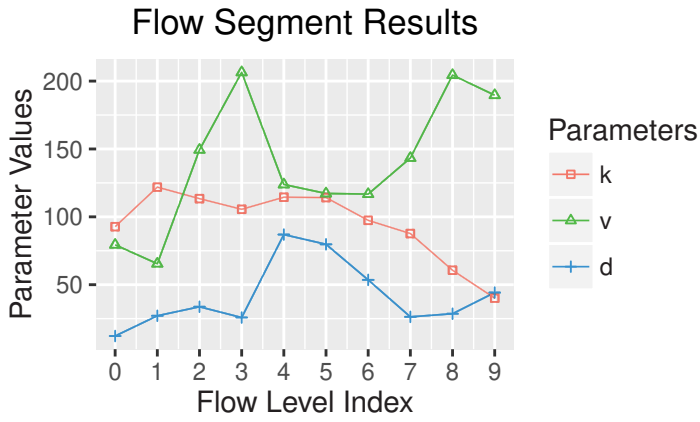

Figure E.4: Parameters impacted by flow rate when the flow rate is increasing during morning time. Values of v are squared to ease display on the same scale.

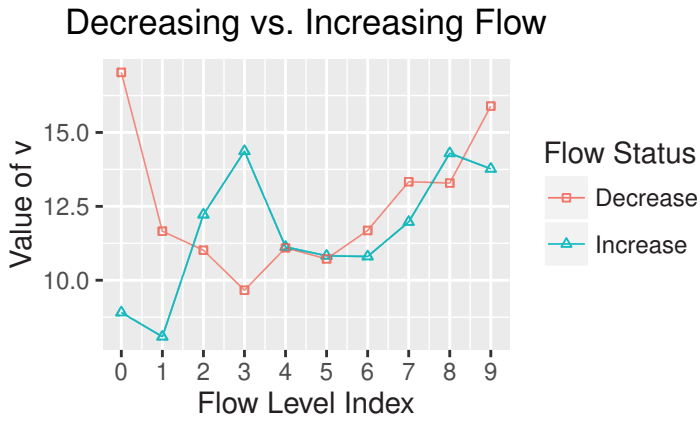

Figure E.5: Values of v for increasing vs. decreasing flows.

The results show that the parameter values are not linearly correlated with flow rate. There is no obvious similarity among patterns of three parameters when flow changes.

Besides, those patterns may differ when comparing decreasing flow with increasing flow. The traffic flow rate increases from 3 am to 15 pm

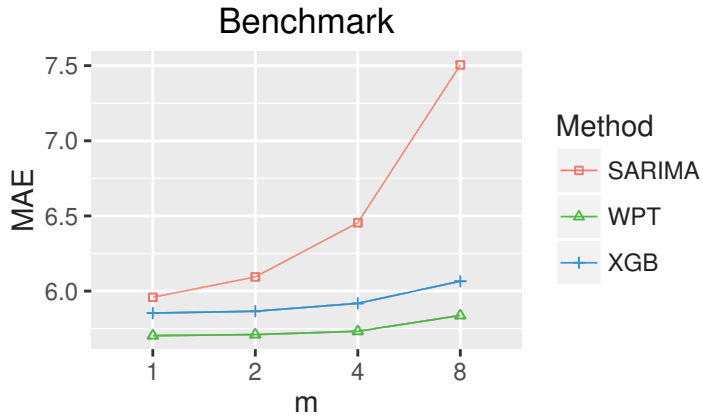

Figure E.6: Results of benchmarking WPT with SARIMA and XGB.

and decreases after 15 pm until the next early morning. The patterns of weighted k and d are similar, but the pattern of v values is different when flow level is lower than 4. Thus, only v's values are plotted as shown in Figure E.5.

E.5.2 Benchmarking

The benchmark results are shown in Figure E.6. When m increases, the error of SARIMA grows from 5.959 to 7.505 which is 25.95%. The error of XGB grows from 5.853 to 6.064 (3.61%). The error of WPT only grows from 5.702 to 5.836 (2.34%). Thus, WPT is more robust. Besides, WPT is giving 0.5% higher accuracy when being compared with the best possible result (5.772) of manually tuned parameter tuple p for any m.

E.6 Analysis of Results

Previous work used relatively small values for parameters by default (e.g. $k = 8$ or $10, d = 3$ or 4 and $v = 0$), which is hard to produce optimal results.

Considering flow rate (Figure E.4), the higher the flow rate the less the neighbours needed. The reason is high flow occurs in holidays and number of holidays are much less than workdays. For d and v, the trend is not monotonous. None of the relations between any of the three parameters and flow rate is linear. Thus, considering different flow levels separately is a good idea, especially when there is enough data to train weights. For incremental and decremental flow, as v has different relations with flow rate, it is necessary to treat incremental and decremental flows separately.

The benchmark results show that WPT gives not only the most accurate but also the most robust result when m changes. Considering all m choices, in average, WPT gives flow rate prediction accuracy 5.744, which is 3.05% improvement when compared with 5.925 given by XGB, and 11.7% improvement in comparison to 6.503 given by SARIMA.

To reduce the calculation time further, we used top 25% of the tuples on each flow level. If the GPU kernels are modified accordingly, it can accelerate prediction stage by four times. The experiment shows it also increases accuracy by discarding bad tuples.

E.7 Conclusion

This paper proposes to use WPT to make kNN dynamically tuned regarding dynamic flow rate levels. WPT gives the performance that cannot be achieved using the manual tuning. Besides, WPT is 3.05% better than XGB and 11.7% better than SARIMA. WPT is not only accurate but also space efficient. Only weights are saved which is one hundred kilobyte in the experiment, and one-year history traffic data uses less than one-megabyte space. Additionally, WPT is more robust when predicting multi-step ahead.

One problem we are facing is that the real-world data is usually dirty which has to be handled when designing and implementing WPT algorithm. Cleaning data in the pre-processing stage is not easy since man-

ual registration work and testimony of incident eyewitnesses often lack accuracy. Besides, the distance measurement of neighbours is also important. We plan to investigate these topics further in the future work.

Acknowledgment

We would like to thank reviewers for their valuable suggestions. This work was supported by National Natural Science Foundation of China (NSFC), under grant agreement 61364019.

F

An Overview of Parameter and Data Strategies for *k*-Nearest Neighbours Based Short-Term Traffic Prediction

Bin Sun, Wei Cheng, Prashant Goswami, Guohua Bai. Symposium on Computers and Communication (ISCC). IEEE: July 2017.

Abstract

Modern intelligent transportation systems (ITS) requires reliable and accurate short-term traffic prediction. One widely used method to predict traffic is *k*-nearest neighbours (*k*NN). Though many studies have tried to improve *k*NN with parameter strategies and data strategies, there is no comprehensive analysis of those strategies. This paper aims to analyse *k*NN strategies and guide future work to select the right strategy to improve prediction accuracy. Firstly, we examine the relations among three *k*NN parameters, which are: number of nearest neighbours (k), search step length (d) and window size (v). We also analysed predict step ahead (m) which is not a parameter but a user requirement and configuration. The analyses indicate that the relations among parameters are compound especially when traffic flow states are considered. The results show that strategy of using v leads to outstanding accuracy improvement. Later, we compare different data

strategies such as flow-aware and time-aware ones together with ensemble strategies. The experiments show that the flow-aware strategy performs better than the time-aware one. Thus, we suggest considering all parameter strategies simultaneously as ensemble strategies especially by including v in flow-aware strategies.

Index terms— Short-Term Traffic Prediction, k-Nearest Neighbours Regression, Parameter and Data Strategies

F.1 Introduction

Reliable and accurate short-term traffic prediction is a key requirement in modern intelligent transportation systems (ITS) [3]. Short-term traffic prediction is essential for efficient traffic management and incident detection [2]. However, it is a complex task and has been a research subject for the past few decades [4]. The difficulty is due to the fact that traffic flow is influenced by many factors including people, vehicles, roads, environment and information [24].

There are two categories of methods for short-term traffic prediction: parametric and nonparametric [4, 16, 18]. Within the nonparametric methods, one widely used algorithm is k-Nearest Neighbours (kNN) [60, 71, 194]. With the substantial increment of available data [4], kNN is gaining attention due to its flexibility in solving nonlinear problems [36]. Besides, it is easy to understand and implement [199].

Strategies for improving kNN can be divided into two categories, parameter strategies and data strategies. Considering the parameter strategies, three parameters of kNN are critical: the number of nearest neighbours (k), the search step length (also known as lag d), and the window size (also known as lag constraint) (v) [36,58]. Although the way to measure the distance of neighbours is also important, it is beyond this paper's scope when considering parameters. One problem in basic kNN is that it has fixed parameters which do not consider time-varying and nonstable statistical characteristics of traffic flow. Thus, kNN needs

different parameter values for its three parameters under different flow situation. That is why data strategies matter. [18] compared *k*NN performance under four datasets. Their results show that datasets with different traffic characteristics require diverse parameter configurations to perform well and the parameters of *k*NN should be chosen carefully.

To the best knowledge of the authors, there is no analysis considering all strategies simultaneously. Two possible reasons are: much computation is required due to the compound parameter relations and there was not enough data to test data strategies. This paper analyses and compares strategies for all parameters at the same time to provide a comprehensive understanding. Also, we provide a guideline to choose strategies with limited and fixed amount of data for short-term traffic prediction while taking data strategies and other settings into consideration.

F.2 Strategies for *k*NN

This section introduces the basic *k*NN algorithm then the parameter and data strategies that are used to improve *k*NN.

F.2.1 The *k*NN Algorithm

*k*NN predicts the future using history traffic data which are similar to current one. Recent traffic is described by a traffic state vector using both flow rate and speed:

$$S_{[t]} = \begin{bmatrix} r_{t-1} & r_{t-2} & \cdots & r_{t-d} \\ s_{t-1} & s_{t-2} & \cdots & s_{t-d} \end{bmatrix} \tag{F.1}$$

where $t-1$ is the time point of the last received data, r is flow rate and s is speed. S is then compared with each day's data in database and the most similar k neighbours are selected. The dissimilarity is measured using Euclidean distance. While previous work considers d search steps as d dimensions [72], we consider all search steps as one dimension to

avoid the curse of dimension problem. The following equation is used to calculate distance for d search steps:

$$\frac{\sum_{i_d=1}^{d} \sqrt{(r_{t-i_d} - r'_{t-i_d})^2 + (s_{t-i_d} - s'_{t-i_d})^2}}{d} \tag{F.2}$$

where r' and s' are flow rate and speed of an arbitrary day in the history database. If we take window size into consideration (suppose $v = 1$), while fixing $\boldsymbol{S}$, the flow rate vector can be $[r'_{t-2} \cdots r'_{t-d-1}]$ and $[r'_t \cdots r'_{t-d+1}]$ in addition to $[r'_{t-1} \cdots r'_{t-d}]$. The speed vector s' changes together with r'. That is, there are not one but three potential neighbours in one history day when $v = 1$.

While taking all three parameters into consideration, kNN becomes complicated since there are many possible tuples when assigning values to parameters. Ensemble method can be used to solve this complicated problem as introduced in [116]. The basic idea is to use weighted average of predictions of parameter tuples as final prediction. The weights of tuples are generated using training dataset.

F.2.2 Parameters Strategies

Although there is no rule for the choice of kNN parameter values, some values are frequently used, and this configuration is used as baseline A during comparison. To have a good value assigned to k, [185] trained artificial neural network to map characteristics of a dataset to a good k value. [186] updates k continuously considering piecewise linear nature in time series. For search step length (d), [71] suggests it to be bigger than $2D + 1$ according to Takens theorem, where D is the number of features. However, some research [182, 190] shows opposite results. Though v is also an important parameter, only a few studies considered it. Using v was not showing promising improvement in [58], the reason can be that they determined v separately from k and d. In some studies, both k and d get optimised [72, 182]. [71] shows we should optimise k and d at the same time. [182] uses part of the data (like training data) to find best parameters tuple and then the tuple is used for prediction. We modified

this method to include v and use it as baseline B for comparison in our study.

Some research shows improved performance using ensemble kNN [65,200]. While they are only considering k, [116] uses ensemble strategy for all kNN parameters.

F.2.3 Data Strategies

A data strategy is a way to separate training dataset to suitable sub-datasets according to different characteristics of instances. Data strategies are usually domain specific.

A frequently used data strategy is time-aware separation (TA) as the traffic at similar time usually has similar patterns. Previous studies have pointed out the traffic has different patterns at different hour of the day, workday vs. holiday [64, 134, 191, 201]. Considering hour of the day is a typical time-aware strategy and considering workday vs. holiday is a workday-aware strategy. The literature has used the time-variant characteristics to improve traffic prediction [52], where one-day time is separated into four stages. We compare data segmentation strategies of four stages (TA4) and ten stages (TA10)

However, if records are separated by hour of the day, it is not guaranteed that the flow situation is the same, neither day of week, etc. We propose to use another data strategy which is flow-aware separation (FA). The kNN algorithm is adapting to different flow rate levels. We compare data segmentation strategies of four levels (FA4) and ten levels (FA10). The flow rate should be determined by averaging the last 15 minutes traffic which is the minimal averaging time to get stable traffic flow data [177].

For flow-aware strategies, traffic flow can have two trends which are increasing flow and decreasing flow. One way is to consider those two trends separately (up vs. down), noted as UvD in this work. Another way is to not distinguish those two trends, noted as U+D.

F.3 Experiments

The traffic data are from devices along a highway named Kunshi in China. The flow on the road is usually under-saturated. Based on a five-minute interval, each device sends a record to the central database. Every record contains statistical values such as flow rate and speed. Data from one device are cleaned using the method from [118] and then used in this paper. The time range of data are between early-April 2013 and mid-May 2014.

The collected data are divided into 80% and 20% for training and testing respectively, separated on date 20th February 2014. If more than 10% searching steps or prediction steps are influenced by any incident, the record will be discarded. Exponential incremental values are assigned to k, d and v. As a nonstrategy setting, predict step ahead (m) also impacts experiment results and also uses exponential incremental values.

The experiments are conducted with CUDA [180] 8.0 on GeForce GTX 690 graphics card. The analyses are conducted with R language [198] version 3.3.

For the measurement of prediction performance, mean absolute error (MAE) is used. Mean absolute percentage error (MAPE) is not suitable as the flow can be zero during night.

Three terms are used to distinguish different types of relations within and out of strategies as well as nonstrategy settings. *Influence*: Given a time point, all three parameters contribute to the prediction performance, i.e. MAE $= g(k, d, v)$. The influences g_k, g_d, g_v of three parameters are presented separately to make results clear. *Effect*: The influence (g) of parameters on the prediction is also affected by the other two parameters. For instance, the influence of k on MAE (g_k) is under the effect of d and v, i.e. g_k changes when d or v change.Those effects are presented after the influences. *Impact*: For different flow-/time-aware index and predict step ahead (m) etc., the best values of parameters are dif-

ferent. Those factors are nonstrategy settings, but still impact factors to kNN parameters.

F.4 Results

This section provides comprehensive and detailed results regarding influences, effects and impacts of different strategies in the accuracy aspect.

As there are hundreds of thousands of parameter tuples (p), only some results are plotted. Most patterns are general and can hold for values that are not plotted unless otherwise mentioned and explained. For instance, only flow rate prediction is shown because speed prediction results have similar trends. The parameter values for plots are set to k = 8, d = 4, v = 0, unless otherwise mentioned. Those values are used as they are similar to default or well-performed values in previous studied [71, 194]. This can be considered as slicing of the multi-dimension results from this study. The differences of patterns introduced by m within different strategies are ignorable unless otherwise mentioned.

F.4.1 Influence of k

If the results are sliced by $d = 4, v = 0$ (i.e. search time is 20 minutes, no window), the remaining is g_k, i.e. the influence of k on kNN performance (MAE) as shown in Figure F.1. The result is showing comparisons between parameter predictions and ensemble strategies, each with four different m values. The plot also shows that when k increases, MAE decreases first (when $k < 32$) and then increases (when $k > 64$) on g_k curves. Thus, the influences g_k contain turning points (the k value for lowest MAE on one curve) between $k = 32$ and $k = 64$. Ensemble strategies are giving the lowest MAE when being compared with any parameter tuple p for any m. This conclusion holds for any v and d values when considering the influence from k on performance.

The **effect of d on g_k** is shown in Figure F.2, all curves have turning

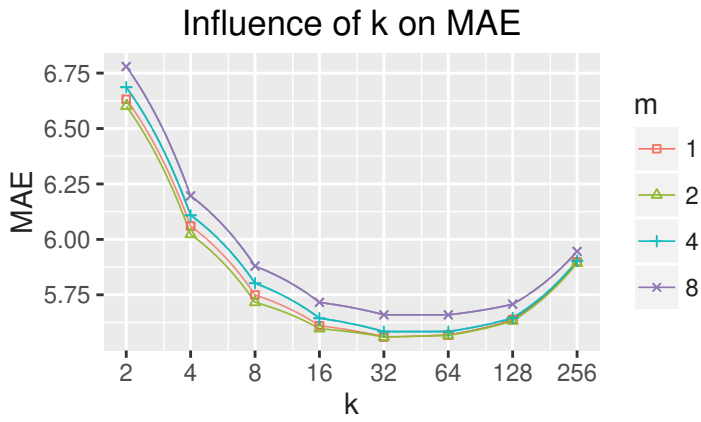

Figure F.1: Influence of k on MAE (g_k) when $d = 4, v = 0$. Results of predictions with different parameter values construct four curves with turning points between $k = 32$ and $k = 64$.

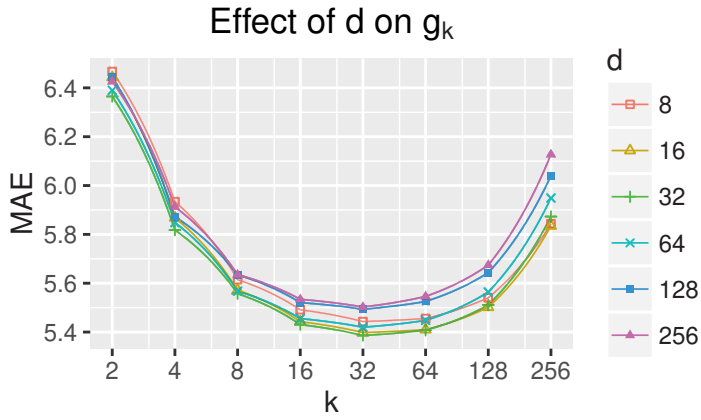

Figure F.2: Effect of d on g_k when $v = 0$. The value of d affects the shape and turning point of g_k curves. The gradients after turning points increase if d becomes bigger. The turning point is moving from range $[32, 64]$ to range $[16, 32]$.

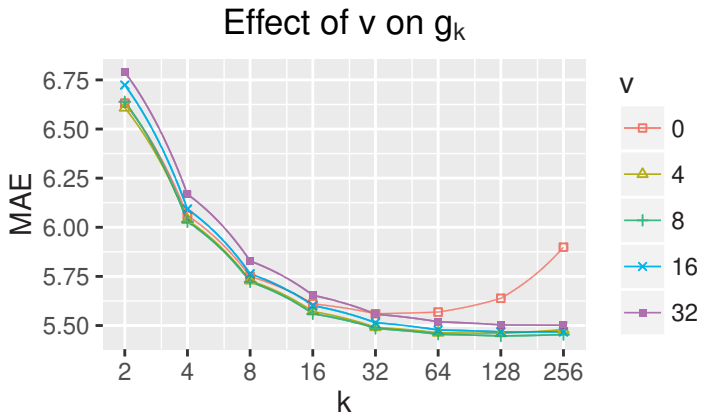

Figure F.3: Effect of v on g_k when $d = 4$. As v becomes bigger, turning points are moving from around $k = 32$ to higher k values and has potential and trend to go beyond 256 if $v > 32$. This plot is when $d = 4$.

points. The effect of d is, if d is big, MAE increases fast when k grows. Besides, the turning points occur at smaller k values, moving from $k > 32$ to $k < 32$. The **effect of v on g_k** is shown in Figure F.3, most curves have turning points around $k = 128$. However, when $v = 0$, the turning point is around $k = 32$. If v is small, say $v < 8$, there are turning points on g_k curves $k < 128$. If v is bigger than 8, the turning points of g_k move to higher k values $(k > 128)$. The above conclusion is much clearer when d is big.

F.4.2 Influence of d

If the results are sliced by $k = 8, v = 0$ (i.e. 8 nearest neighbours, no window), the remaining is g_d, i.e. the influence of d on MAE as shown in Figure F.4. The plot also shows that when d increases, MAE decreases first (when $d < 32$) then increase (when $d > 32$) for g_d curves. Thus, the influences (g_k) contain turning points around $d = 32$. A special pattern is that there are peaks when $d = 128$.

The **effect of k on g_d** is shown in Figure F.5, there are always turning

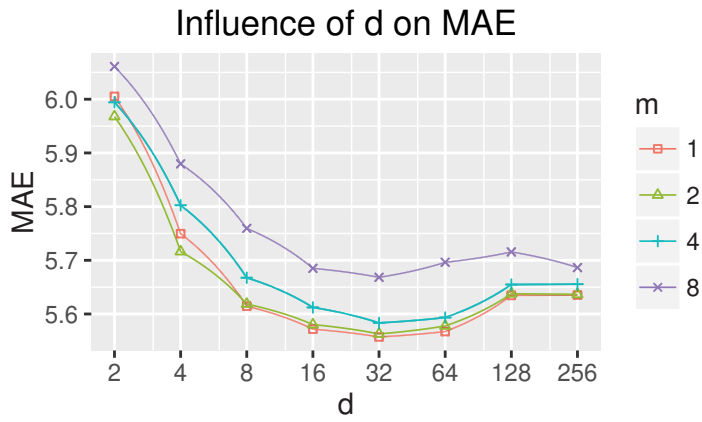

Figure F.4: Influence of d on MAE (g_d) when $k = 8, v = 0$. Results of parameter predictions construct four curves with turning points around $k = 32$ and. Ensemble strategies are giving the lowest MAE compared with any parameter tuple p for any m.

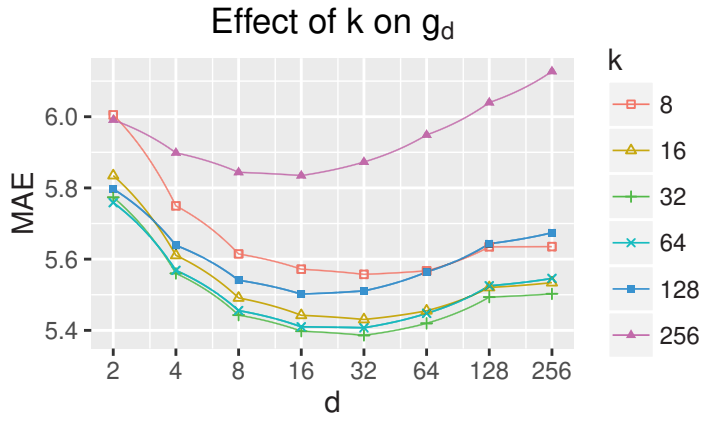

Figure F.5: Effect of k on g_d when $v = 0$. There are always turning points for all curves. If k increases, turning points will occur at a smaller d value and the curves becomes flatter before turning points.

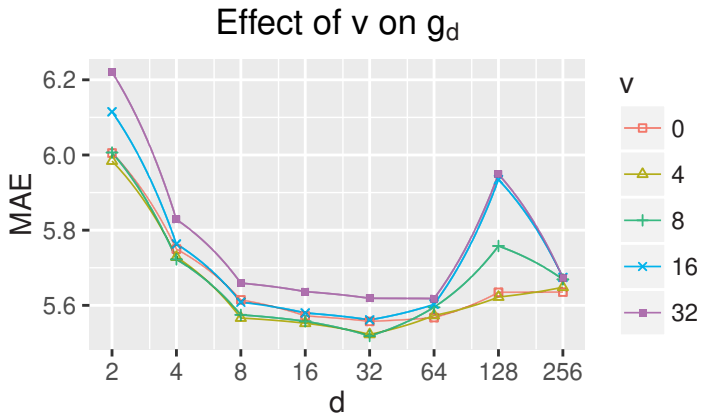

Figure F.6: Effect of v on g_d when $k = 8$. If v increases, g_d becomes steeper and turning points are clearer (around $d = 32$). When d value is close to the curve's turning point, v's effect is smaller.

points around $d = 16$ or 32 on all curves. If k increases, turning points are clearer and occur at a smaller d value, moving from around $d = 32$ to near $d = 16$. If k is small, the curves are steeper. For big values of k, the curves are flatter before turning points. The **effect of v on g_d** is shown in Figure F.6, if v increases, g_d becomes steeper both before turning points and after turning points, which makes the turning points clearer. If v is big enough ($v \geq 16$), there are some peaks at $d = 128$ (only when k is too small). When d's value is close to the curve's turning point (around $d = 32$), v's effect is smaller. There are turning points on all curves and are in the range of $d \in$[16,64], around 32. For the **effect of both k and v on g_d**, if window size (v) becomes big enough (e.g. 8), the turning points of g_d occur at higher values ($d = 32$) and not changing.

F.4.3 Influence of v

If the results are sliced by $k = 8, d = 4$, the remaining is g_v, i.e. the influence of v on MAE as shown in Figure F.7. For any m there are turning points around $v = 4$. When m increases, v values of turning points are slightly decreasing.

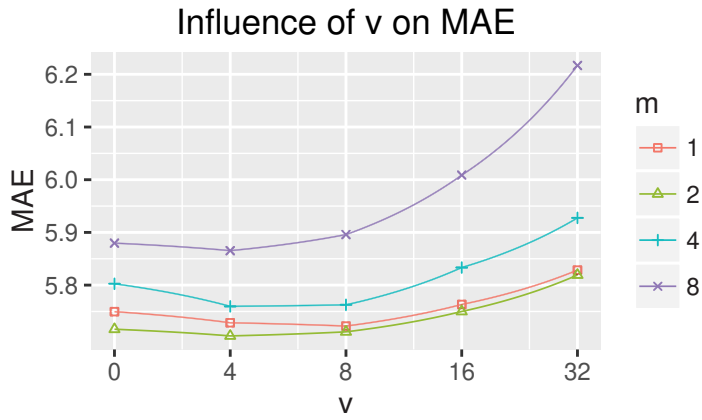

Figure F.7: Influence of v on MAE (g_v) when $k = 8, d = 4$. Results of parameter predictions construct four curves. For some m values, there are turning points at $v = 4$.

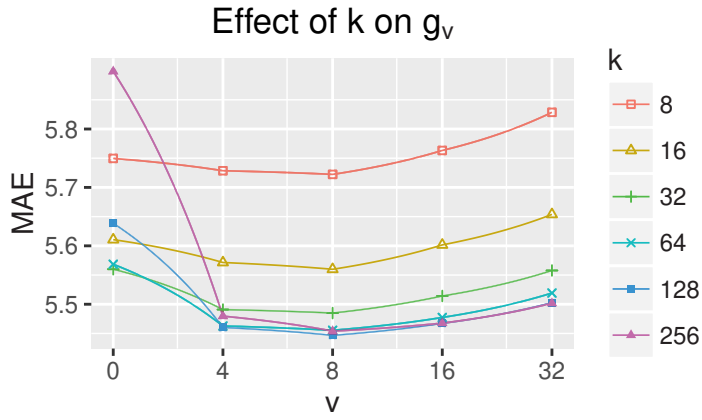

Figure F.8: Effect of k on g_v when $d = 4$. The g_v curves have turning points around $v = 8$. If k becomes bigger, the slopes before the turning points become steeper.

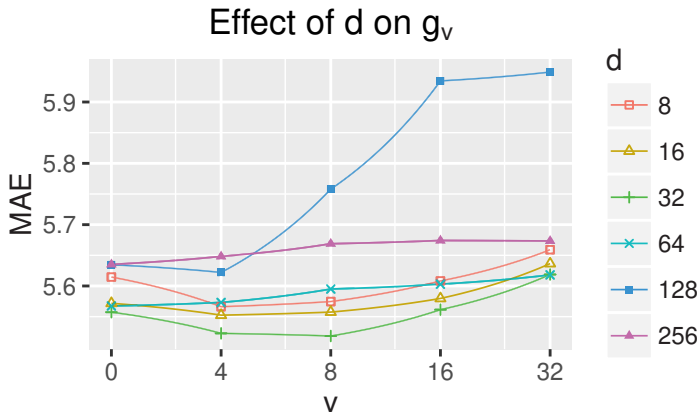

Figure F.9: Effect of d on g_v when $k = 8$. As shown in Figure F.9, when $k = 8$ there are usually turning points around $v = 4$ or $v = 8$ except for $d = 256$. When $d = 128$ (and k is too small), there is a MAE peak for big window sizes ($v > 4$). If d grows continuously after the peak, the turning points disappear.

The **effect of** k **on** g_v is shown in Figure F.8, when search step length is small ($d = 4$), if k is small, there are always turning points around at $v = 8$. If k becomes big, the effect of k on g_v is stable, the pattern of the curves are not changing, but just become more obvious. The **effect of** d **on** g_v is shown in Figure F.9, there are usually turning points around $v = 4$ or $v = 8$ except for $d = 256$. If k increases to 128 (not small any longer), the results indicate that v can reduce around 5% error. However, previous work reported that v is not so useful as it reduced the error by about only 0.05% as shown in the second figure of [58].

F.4.4 Impact of Flow Rate

To have a robust pattern analysis, the ensemble weights are used to calculate weighted parameter values as shown in Figure F.10. The results show that optimal parameter values are dependent on flow levels. Three parameters have different changing patterns when the flow changes. Besides, the pattern of v values under increasing flow is different from the pattern under decreasing flow.

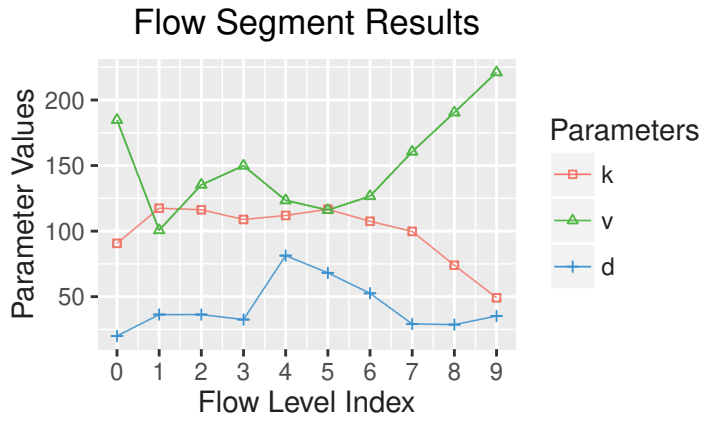

Figure F.10: Parameters impacted by flow rate when flow rate is decreasing and increasing (U+D). Higher index means higher flow rate. Values of v are squared to ease display on the same scale.

F.4.5 Impact of Predict Step Ahead

As shown in Figure F.11, when m increases, d is increasing slightly. Similar trend has been seen in previous literature [72], but it was not obvious. The result also shows that v is decreasing and k is constant, while [72] reported their k values were going up and down. It might be due to the limited data that had been used.

F.4.6 Impact of Data Strategies

Results of data strategies are shown in Figure F.12. All strategies have the same parameter options as in parameter strategies except baselines. Baseline A is commonly used (default) parameter tuple values: $k = 8, d = 4, v = 0$, which give MAE 6.2429. It is too high and not shown here. Baseline B is the best single tuple according to training data, potentially with the problem of overfitting. Workday-aware strategy has no segmentation situation, so it is shown as a dashed black line. The results show that all strategies that are considering window size parameter are better than the baseline A. Some flow-aware fixed segmentation strate-

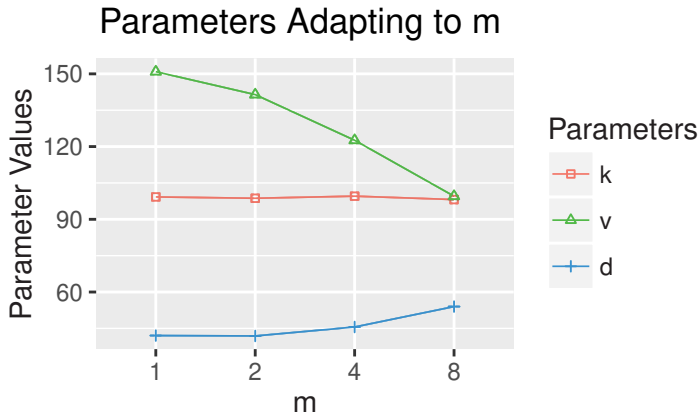

Figure F.11: Parameters impacted by m (1 step is 5 minutes). When m increases, k is constant, d is increasing slightly, v is decreasing. Values of v are squared to ease display on the same scale.

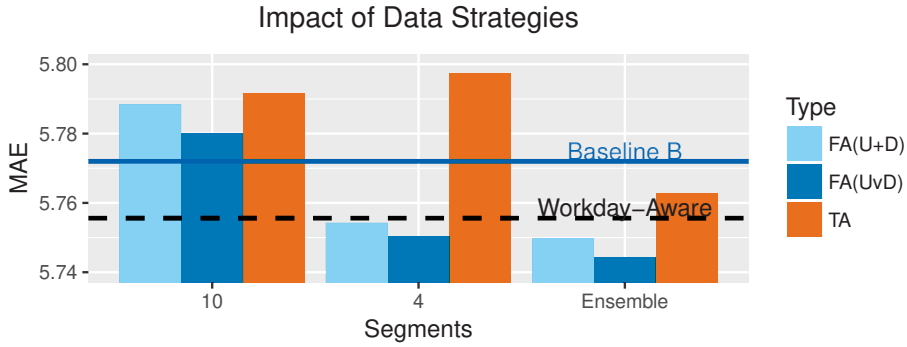

Figure F.12: MAE of data strategies, including flow-aware(FA), time-aware(TA) and workday-aware strategy. Three segmentation situations: 10 segments, 4 segments and ensemble(E). Baseline A: common values (not shown). Baseline B (solid blue line): best values from training data.

gies are better than baseline B. Ensemble strategies are always better than both baselines, and better than other nonensemble strategies.

F.5 Analysis of Results

This section summarizes and interprets the results, and analyses the influence, effects and impacts of strategies and settings.

Analysis of Parameter Strategies. A general pattern is that the error goes down first and then goes up when one parameter increase, i.e., when increasing the values, turning points will occur. This pattern is clearer especially near the optimal values. Small values were used for parameters in some literature (e.g. $k = 8$ or $10, d = 3$ or 4 and $v = 0$) which leads to lack of accuracy. To get higher accuracy, the values need to be increased to turning points. Although the turning points are the best choices for high accuracy, the weighted results from ensemble strategy are more accurate than any specific tuple in the experiments. Even if the peaks are not influencing final results a lot, they make the influence functions (g) not purely convex. Thus, it is necessary to consider all parameter strategies at the same time instead of using separately. It is also not possible to use simple gradient descent without step size tuned.

Analysis of Predict Step Ahead. As shown in Figure F.11, longer predict time (bigger m) needs longer search steps (bigger d). Besides, window size should be smaller. However, the number of actual selected nearest neighbours (k) is stable. Previous work indicates big m contributes to prediction error a lot, such as the ninth figure in [71] shows around 20% more error when m increases from 1 to 6. However, using our ensemble kNN, when m increases from 1 to 8, prediction error increases only 2%.

Analysis of Data Strategies. It is necessary to separate incremental and decremental flow data. Window size v gives different patterns for increasing and decreasing without doubt, which can be the actual reason that UvD strategies are better than U+D. The TA4 strategy here is

not giving a good result. A previous study showed good results with TA4 [52], the reason could be that the segmentation is suitable for their data. One need to be more cautious if a nonensemble method is used. Although we can see clear difference between workday and holiday traffic patterns, the workday-aware strategy is giving less improvement comparing with FA strategies. Thus, FA strategies are preferred, if possible. When the flow rate is low, kNN can benefit from more nearest neighbours. The reason is that low flow occurs in workdays and the number of workdays are much more than holidays. The patterns of d and v are not monotonous. All patterns of the three parameters are nonlinear. Thus, the algorithm should consider flow rate when applying parameter strategies.

F.6 Conclusion

This paper firstly analysed the compound relations when applying parameter strategies. The performance of ensemble kNN cannot be achieved using manual adjustment due to the compound relations, which make it important to consider parameter strategies of all parameters at the same time. It is important to investigate parameters in detail before applying any strategy. Our work is general and it covers previous work in literature where only part of parameter strategies has been considered. For instance, the prediction error decreases continuously when d increases in [71], but we found that there is turning point when d is big enough.

When using data strategies, it is better to separate increasing traffic from decreasing one. Too detailed separation (e.g. 10 segments instead of 4) is usually not good, because each segment has less training data. However, even the number of segments is good, the separation should also be optimised, otherwise different types of instances are in the same segment, TA4 is a negative example here.

In one sentence, we suggest considering all parameter strategies simultaneously as ensemble strategies especially by including v together with k and d in flow-aware strategies.

During the data pre-processing stage, we found the real world data are hard to use directly. Cleaning transportation data requires correct data imputation. We plan to investigate this problem in our future work.

Acknowledgements

This work was supported by National Natural Science Foundation of China (NSFC), under grant agreement 61364019.

References

[1] S. Clark. Traffic prediction using multivariate nonparametric regression. *Journal of Transportation Engineering*, 129(2):161–168, March 2003.

[2] Jianhua Guo, Wei Huang, and Billy M. Williams. Real time traffic flow outlier detection using short-term traffic conditional variance prediction. *Transportation Research Part C-Emerging Technologies*, 50:160–172, January 2015.

[3] Yuexian Zou, Guangyi Shi, Hang Shi, and He Zhao. Traffic incident classification at intersections based on image sequences by HMM/SVM classifiers. *Multimedia Tools and Applications*, 52(1):133–145, March 2011.

[4] Simon Oh, Young-Ji Byon, Kitae Jang, and Hwasoo Yeo. Short-term Travel-time Prediction on Highway: A Review of the Data-driven Approach. *Transport Reviews*, 35(1):4–32, January 2015.

[5] S. Vasantha Kumar and Lelitha Vanajakshi. Short-term traffic flow prediction using seasonal ARIMA model with limited input data. *European Transport Research Review*, 7(3):21, September 2015.

[6] Md Moniruzzaman, Hanna Maoh, and William Anderson. Short-term prediction of border crossing time and traffic volume for commercial trucks: A case study for the Ambassador Bridge. *Transportation Research Part C: Emerging Technologies*, 63:182–194, February 2016.

[7] 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*, 13(3):211–234, June 2005.

[8] Young-Ji Byon and Steve Liang. Real-Time Transportation Mode Detection Using Smartphones and Artificial Neural Networks: Performance Comparisons Between Smartphones and Conventional Global Positioning System Sensors. *Journal of Intelligent Transportation Systems*, 18(3):264–272, July 2014.

[9] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee D. Han. Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications*, 36(3, Part 2):6164–6173, April 2009.

[10] Bin Sun, Wei Cheng, Prashant Goswami, and Guohua Bai. Short-Term Traffic Forecasting Using Self-Adjusting k-Nearest Neighbours. *IET Intelligent Transport Systems*, 12(1):41–48, February 2018.

[11] Yuebiao Li, Zhiheng Li, and Li Li. Missing traffic data: Comparison of imputation methods. *IET Intelligent Transport Systems*, 8(1):51–57, February 2014.

[12] Sehyun Tak, Soomin Woo, and Hwasoo Yeo. Data-Driven Imputation Method for Traffic Data in Sectional Units of Road Links. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1762–1771, June 2016.

[13] Li Qu, Li Li, Yi Zhang, and Jianming Hu. PPCA-Based Missing Data Imputation for Traffic Flow Volume: A Systematical Approach. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):512–522, September 2009.

[14] Manoel M. Castro-Neto, Lee D. Han, Young-Seon Jeong, and Myong K. Jeong. Toward Training-Free Automatic Detection of Freeway Incidents Simple Algorithm with One Parameter. *Transportation Research Record*, 2278:42–49, 2012.

[15] Mohammed S. Ahmed and Allen R. Cook. Analysis of freeway traffic time-series data by using box-jenkins techniques. *Transportation Research Record*, 722, 1979.

[16] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, 43(1):3–

19, June 2014.

[17] E. I. Vlahogianni, J. C. Golias, and M. G. Karlaftis. Short-term traffic forecasting: Overview of objectives and methods. *Transport Reviews*, 24(5):533–557, September 2004.

[18] Lei Lin, Qian Wang, and Adel W. Sadek. Short-Term Forecasting of Traffic Volume - Evaluating Models Based on Multiple Data Sets and Data Diagnosis Measures. *Transportation Research Record*, 2013(2392):40–47, 2013.

[19] M. Papageorgiou, J. 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: Policy and Practice*, 24A(5), September 1990.

[20] JWC van Lint and CPIJ van Hinsbergen. Short-term traffic and travel time prediction models. *Artificial Intelligence Applications to Critical Transportation Issues*, 22, 2012.

[21] J.W.C. van Lint. *Reliable Travel Time Prediction for Freeways*. PhD thesis, Netherlands TRAIL Research School, May 2004.

[22] Roland Chrobok. *Theory and Application of Advanced Traffic Forecast Methods*. PhD thesis, Universität Duisburg-Essen, Fakultät für Physik, 2005.

[23] Dong-wei Xu, Yong-dong Wang, Li-min Jia, Hai-jian Li, and Guijun Zhang. Real-time road traffic states measurement based on Kernel-KNN matching of regional traffic attractors. *Measurement*, 94:862–872, December 2016.

[24] Roger P. Roess, Elena S. Prassas, and William R. McShane. *Traffic Engineering*. Prentice Hall, Upper Saddle River, NJ, 4 edition, 2010.

[25] Laura Wynter, Wanli Min, and Yasuo Amemiya. Short-, Medium- and Long-term Road Traffic Prediction with Spatio-Temporal Correlations. Technical report, IBM Research Division, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, 2006.

[26] Fei Su, Honghui Dong, Limin Jia, Yong Qin, and Zhao Tian. Long-term forecasting oriented to urban expressway traffic situation. *Advances in Mechanical Engineering*, 8(1):1687814016628397, January 2016.

[27] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The

MIT Press, Cambridge, MA, 1 edition edition, August 2012.

[28] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1st edition edition, October 1997.

[29] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[30] Yorgos J. Stephanedes, Panos G. Michalopoulos, and Roger A. Plum. Improved Estimation of Traffic Flow for Real-Time Control. In *Transportation Research Record*, 1981.

[31] Rob J Hyndman and George Athanasopoulos. Seasonal ARIMA models. In *Forecasting: Principles and Practice*. oTexts, 2015.

[32] Jui-Sheng Chou and Abdi Suryadinata Telaga. Real-time detection of anomalous power consumption. *Renewable & Sustainable Energy Reviews*, 33:400–411, May 2014.

[33] Tomas Singliar and Milos Hauskrecht. Learning to detect incidents from noisily labeled data. *Machine Learning*, 79(3):335–354, June 2010.

[34] 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*, 14(2):871–882, June 2013.

[35] Filmon G. Habtemichael and Mecit Cetin. Short-term traffic flow rate forecasting based on identifying similar traffic patterns. *Transportation Research Part C-Emerging Technologies*, 66:61–78, May 2016.

[36] Marcin Bernas, Bartlomiej Placzek, Piotr Porwik, and Teresa Pamula. Segmentation of vehicle detector data for improved k-nearest neighbours-based traffic flow prediction. *IET Intelligent Transport Systems*, 9(3):264–274, April 2015.

[37] L Breiman, JH Friedman, RA Olshen, and CJ Stone. *Classification and Regression Trees*. Wadsworth Books / CRC Press, Belmont, California, 1984.

[38] Soroush Rashidi, Prakash Ranjitkar, and Yuval Hadas. Modeling Bus Dwell Time with Decision Tree-Based Methods. *Transportation Research Record*, 2418:74–83, 2014.

[39] Leo Breiman. Bagging predictors. Technical Report 421, University of California, Berkeley, CA, 1994.

[40] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[41] Peter Flach. Model ensembles. In *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*, pages 330–342. Cambridge University Press, 2012.

[42] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[43] Michael Kearns. Thoughts on hypothesis boosting. Technical report, University of Pennsylvania, 1988.

[44] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[45] Dana Kaner. Bagging predictors and random forest. Technical report, Tel Aviv University, Tel Aviv, Israel, May 2017.

[46] Leo Breiman. Bias, variance, and arcing classifiers. Technical Report 460, University of California, Berkeley, CA, April 1996.

[47] Mohammed Elhenawy, Hao Chen, and Hesham A. Rakha. Dynamic travel time prediction using data clustering and genetic programming. *Transportation Research Part C: Emerging Technologies*, 42:82–98, May 2014.

[48] Ivana Semanjski. Potential of Big Data in Forecasting Travel Times. *Promet-Traffic & Transportation*, 27(6):515–528, 2015.

[49] Yi Hou, Praveen Edara, and Carlos Sun. Traffic Flow Forecasting for Urban Work Zones. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1761–1770, August 2015.

[50] Yi Hou, Praveen Edara, and Yohan Chang. Road Network State Estimation using Random Forest Ensemble Learning. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (Itsc)*, New York, 2017. IEEE.

[51] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *22nd SIGKDD Conference on Knowledge Discovery and Data Mining*, San Francisco, August 2016.

[52] D. Wang, Q. Zhang, S. Wu, X. Li, and R. Wang. Traffic flow forecast with urban transport network. In *2016 IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, pages 139–143, August 2016.

[53] S.R. Mousa, P.R. Bakhit, O.A. Osman, and S. Ishak. A Comparative Analysis of Tree-Based Ensemble Methods for Detecting Imminent Lane Change Maneuvers in Connected Vehicle Environments. *Transportation Research Record*, 2018.

[54] Harris Drucker, Chris J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support Vector Regression Machines. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, pages 155–161, Cambridge, MA, USA, 1996. MIT Press.

[55] J. Moysen, L. Giupponi, and J. Mangues-Bafalluy. On the potential of ensemble regression techniques for future mobile network planning. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 477–483, June 2016.

[56] Daniel Boto-Giralda, Francisco J. Diaz-Pernas, David Gonzalez-Ortega, Jose F. Diez-Higuera, Miriam Anton-Rodriguez, Mario Martinez-Zarzuela, and Isabel Torre-Diez. Wavelet-Based Denoising for Traffic Volume Time Series Forecasting with Self-Organizing Neural Networks. *Computer-Aided Civil and Infrastructure Engineering*, 25(7):530–545, October 2010.

[57] W. Z. Zheng, D. H. Lee, and Q. X. Shi. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. *Journal of Transportation Engineering-Asce*, 132(2):114–121, February 2006.

[58] Zuduo Zheng and Dongcai Su. Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. *Transportation Research Part C-Emerging Technologies*, 43:143–157, June 2014.

[59] Soroush Salek Moghaddam and Bruce Hellinga. Real-Time Prediction of Arterial Roadway Travel Times Using Data Collected by Bluetooth Detectors. *Transportation Research Record*, 2442:117–128, 2014.

[60] Shanhua Wu, Zhongzhen Yang, Xiaocong Zhu, and Bin Yu. Improved k-nn for Short-Term Traffic Forecasting Using Temporal and Spatial Information. *Journal of Transportation Engineering*, 140(7):04014026, July 2014.

[61] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[62] Gary A. Davis and Nancy L. Nihan. Nonparametric regression and short-term freeway traffic forecasting. *Journal of Transportation Engineering*, 117(2):178–188, March 1991.

[63] B. L. Smith and M. J. Demetsky. Traffic Flow Forecasting: Comparison of Modeling Approaches. *Journal of Transportation Engineering*, 123(4):261–266, July 1997.

[64] Pietro Dell'Acqua, Francesco Bellotti, Riccardo Berta, and Alessandro De Gloria. Time-Aware Multivariate Nearest Neighbor Regression Methods for Traffic Flow Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3393–3402, 2015.

[65] Haikun Hong, Xiabing Zhou, Wenhao Huang, Xingxing Xing, Fei Chen, Yu Lei, Kaigui Bian, and Kunqing Xie. Learning Common Metrics for Homogenous Tasks in Traffic Flow Prediction. In *14th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1007–1012, Miami, FL, December 2015. IEEE.

[66] Y. Ohba, H. Ueno, and M. Kuwahara. Travel time calculation method for expressway using toll collection system data. In *Proceedings of IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, pages 471–475, Tokyo, Japan, 1999.

[67] Sul.I. Bajwa, E. Chung, and M. Kuwahara. Performance evaluation of an adaptive travel time prediction model. In *Proceedings of IEEE Intelligent Transportation Systems*, pages 1000–1005, Vienna, Austria, September 2005.

[68] Juan Jesús Ruiz Aguilar, Ignacio J. Turias, José A. Moscoso Lopez, María J. Jiménez Come, and María M. Cerbán. Forecasting of short-term flow freight congestion: A study case of Algeciras Bay Port (Spain). *DYNA*, 83(195):163–172, February 2016.

[69] Ping Ling, Dajin Gao, Xifeng Zhou, Zhining Huang, and Xiangsheng Rong. Improve the diagnosis of atrial hypertrophy with the local discriminative support vector machine. *Bio-Medical Materials and Engineering*, 26:1813–1820, 2015.

[70] Z. He, B. S. Lee, and R. Snapp. Self-tuning cost modeling of user-defined functions in an object-relational DBMS. *Acm Transactions*

on Database Systems, 30(3):812–853, September 2005.

[71] Byoungjo Yoon and Hyunho Chang. Potentialities of Data-Driven Nonparametric Regression in Urban Signalized Traffic Flow Forecasting. *Journal of Transportation Engineering*, 140(7):04014027, July 2014.

[72] H. Chang, Y. Lee, B. Yoon, and S. Baek. Dynamic near-term traffic flow prediction: System-oriented approach based on past experiences. *IET Intelligent Transport Systems*, 6(3):292–305, September 2012.

[73] T. Thomas and E. C. Van Berkum. Detection of incidents and events in urban networks. *IET Intelligent Transport Systems*, 3(2):198–205, June 2009.

[74] Fred L. Hall, Yong Shi, and George Atala. On-line testing of the McMaster incident detection algorithm under recurrent congestion. *Transportation Research Record*, 1394:1–7, 1993.

[75] F. Yuan and R. L. Cheu. Incident detection using support vector machines. *Transportation Research Part C-Emerging Technologies*, 11(3):309–328, August 2003.

[76] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Anomaly Detection. In *Introduction to Data Mining*, pages 651–665. Pearson Addison Wesley, Boston, 2006.

[77] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Elsevier, Singapore, 3 edition, 2012.

[78] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Wavelet-Clustering-Neural Network Model for Freeway Incident Detection. *Computer-Aided Civil and Infrastructure Engineering*, 18(5):325–338, September 2003.

[79] Faisal Ahmed and Yaser E. Hawas. A Threshold-Based Real-Time Incident Detection System for Urban Traffic Networks. *Transport Research Arena*, 48:1713–1722, 2012.

[80] Hector Gonzalez, Jiawei Han, Yanfeng Ouyang, and Sebastian Seith. Multidimensional Data Mining of Traffic Anomalies on Large-Scale Road Networks. *Transportation Research Record*, 2215(1):75–84, 2011.

[81] W. H. Lin and C. F. Daganzo. A Simple Detection Scheme for

Delay-Inducing Freeway Incidents. *Transportation Research Part A-Policy and Practice*, 31(2):141–155, March 1997.

[82] Yorgos J. Stephanedes and Athanassios P. Chassiakos. Freeway incident detection through filtering. *Transportation Research Part C: Emerging Technologies*, 1(3):219–233, September 1993.

[83] Yonggang Wang and Chunbo Zhang. Alternative Route Strategy for Emergency Traffic Management Based on ITS: A Case Study of Xi'an Ming City Wall. *Tehnicki Vjesnik-Technical Gazette*, 20(2):359–364, April 2013.

[84] Jiyoun Yeon, Sarah Hernandez, and Lily Elefteriadou. Differences in Freeway Capacity by Day of the Week, Time of Day, and Segment Type. *Journal of Transportation Engineering-ASCE*, 135(7):416–426, July 2009.

[85] Jia Li and H. Michael Zhang. Fundamental Diagram of Traffic Flow - New Identification Scheme and Further Evidence from Empirical Data. *Transportation Research Record*, 2260:50–59, 2011.

[86] E. I. Vlahogianni, M. G. Karlaftis, and A. Stathopoulos. An extreme value based neural clustering approach for identifying traffic states. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, New York, SEP 13-16, 2005. IEEE.

[87] Ricardo Garcia-Rodenas, María L. López-García, and María Teresa Sánchez-Rico. An Approach to Dynamical Classification of Daily Traffic Patterns: An approach to dynamical classification of daily traffic patterns. *Computer-Aided Civil and Infrastructure Engineering*, 32(3):191–212, March 2017.

[88] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Temporal evolution of short-term urban traffic flow: A nonlinear dynamics approach. *Computer-Aided Civil and Infrastructure Engineering*, 23(7):536–548, October 2008.

[89] Traianos-Ioannis Theodorou, Athanasios Salamanis, Dionisis D. Kehagias, Dimitrios Tzovaras, and Christos Tjortjis. Short-Term Traffic Prediction under Both Typical and Atypical Traffic Conditions using a Pattern Transition Model. In *VEHITS*, 2017.

[90] Alexander Ihler, Jon Hutchins, and Padhraic Smyth. Adaptive Event Detection with Time-varying Poisson Processes. In *Proceed-*

ings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pages 207–216, New York, NY, USA, 2006. ACM.

[91] Manish Gupta, Jing Gao, Charu Aggarwal, and Jiawei Han. Outlier Detection for Temporal Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, September 2014.

[92] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying Density-based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 93–104, New York, NY, USA, 2000. ACM.

[93] Zhihua Li, Ziyuan Li, Ning Yu, and Steven Wen. Locality-Based Visual Outlier Detection Algorithm for Time Series. *Security and Communication Networks*, page UNSP 1869787, 2017.

[94] J. Ma and S. Perkins. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1741–1745 vol.3, July 2003.

[95] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data*, 6(1):3, March 2012.

[96] HyungJun Cho, Yang-jin Kim, Hee Jung Jung, Sang-Won Lee, and Jae Won Lee. OutlierD: An R package for outlier detection using quantile regression on mass spectrometry data. *Bioinformatics*, 24(6):882–884, March 2008.

[97] Yuzhe Liu and Vanathi Gopalakrishnan. An Overview and Evaluation of Recent Machine Learning Imputation Methods Using Cardiac Imaging Data. *Data*, 2(1):8, January 2017.

[98] Senniang Chen and Cindy L. Yu. Parameter estimation through semiparametric quantile regression imputation. *Electronic Journal of Statistics*, 10(2):3621–3647, 2016.

[99] Guoyou Qin, Jiajia Zhang, and Zhongyi Zhu. Simultaneous mean and covariance estimation of partially linear models for longitudinal data with missing responses and covariate measurement error. *Computational Statistics & Data Analysis*, 96:24–39, April 2016.

[100] Stefan Oehmcke, Oliver Zielinski, and Oliver Kramer. kNN Ensembles with Penalized DTW for Multivariate Time Series Imputation. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2774–2781, New York, 2016. IEEE.

[101] Yalda Rajabzadeh, Amir Hossein Rezaie, and Hamidreza Amindavar. Short-term traffic flow prediction using time-varying Vasicek model. *Transportation Research Part C: Emerging Technologies*, 74:168–181, January 2017.

[102] Yuebiao Li, Zhiheng Li, Li Li, Yi Zhang, and Maojing Jin. Comparison on PPCA, KPPCA and MPPCA Based Missing Data Imputing for Traffic Flow. In *International Conference on Transportation Information and Safety (ICTIS)*, Wuhan, China, June 2013. American Society of Civil Engineers.

[103] Panagiotis Loukopoulos, Suresh Sampath, Pericles Pilidis, George Zolkiewski, Ian Bennett, Fang Duan, and David Mba. Dealing With Missing Data for Prognostic Purposes. In M. J. Zuo, L. Xing, Z. Li, Z. Tian, and Q. Miao, editors, *2016 Prognostics and System Health Management Conference*, New York, 2016. IEEE.

[104] Panagiotis Loukopoulos, George Zolkiewski, Ian Bennett, Pericles Pilidis, Fang Duan, and David Mba. Dealing with missing data as it pertains of e-maintenance. *Journal of Quality in Maintenance Engineering*, 23(3):260–278, 2017.

[105] Bin Sun, Wei Cheng, Prashant Goswami, and Guohua Bai. An Overview of Parameter and Data Strategies for k-Nearest Neighbours Based Short-Term Traffic Prediction. In *ACM International Conference Proceeding Series ICITT/ICSET 2017*, volume 133326, pages 68–74. ACM, Sep.-Oct. 2017.

[106] Taehyung Kim, Hyoungsoo Kim, and David J Lovell. Traffic flow forecasting: Overcoming memoryless property in nearest neighbor non-parametric regression. In *8th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 965–969, Vienna, Austria, September 2005. IEEE.

[107] PeMS. Dodgers Loop Sensor Data Set from Performance Measurement System (pems.dot.ca.gov). Technical report, State of California, July 2017.

[108] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart. The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, January 2000.

[109] Brent Komer, James Bergstra, and Chris Eliasmith. Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn. In *ICML Workshop on AutoML*, pages 2825–2830, 2014.

[110] Matthew MacKay, Paul Vicol, Jimmy Ba, and Roger Grosse. Reversible Recurrent Neural Networks. In *2018 Conference on Neural Information Processing Systems*, Montréal, Canada, December 2018.

[111] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3918–3926, Stockholmsmässan, Stockholm Sweden, July 2018. PMLR.

[112] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-Keras: Efficient Neural Architecture Search with Network Morphism. *arXiv:1806.10282 [cs, stat]*, June 2018.

[113] Bo-Jian Hou and Zhi-Hua Zhou. Learning with Interpretable Structure from RNN. *arXiv:1810.10708 [cs]*, October 2018.

[114] Ming Zhong, Satish Sharma, and Pawan Lingras. Genetically Designed Models for Accurate Imputation of Missing Traffic Counts. *Transportation Research Record: Journal of the Transportation Research Board*, 1879:71–79, January 2004.

[115] Amanda N. Baraldi and Craig K. Enders. An introduction to modern missing data analyses. *Journal of School Psychology*, 48(1):5–37, February 2010.

[116] Bin Sun, Wei Cheng, Prashant Goswami, and Guohua Bai. Flow-Aware WPT k-Nearest Neighbours Regression for Short-Term Traffic Prediction. In *22nd IEEE Symposium on Computers and Com-*

munication (ISCC), pages 48–53, Heraklion, Greece, July 2017. IEEE.

[117] Brian Everitt. Euclidean distance. In *The Cambridge Dictionary of Statistics*, page 134. Cambridge: Cambridge University Press, 2002.

[118] Bin Sun, Wei Cheng, Guohua Bai, and Prashant Goswami. Correcting and Complementing Freeway Traffic Accident Data Using Mahalanobis Distance Based Outlier Detection. *Tehnicki Vjesnik-Technical Gazette*, 24(5):1597–1607, October 2017.

[119] Shu Xu, Bo Lu, Michael Baldea, Thomas F. Edgar, Willy Wojsznis, Terrence Blevins, and Mark Nixon. Data cleaning in the process industries. *Reviews in Chemical Engineering*, 31(5):453–490, October 2015.

[120] V. Garcia, E. Debreuve, and M. Barlaud. Fast k nearest neighbor search using GPU. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–6, Anchorage, AK, June 2008.

[121] Python Software Foundation. Python.Org, 2017.

[122] Elton Law. Impyute: Library of the different imputation algorithms; methods for dealing with ambiguity and handling missing data., 2017.

[123] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[124] Steffen Moritz and Thomas Bartz-Beielstein. imputeTS: Time Series Missing Value Imputation in R. *R Journal*, 9(1):207–218, June 2017.

[125] Julie Josse and Francois Husson. Handling missing values in exploratory multivariate data analysis methods. *Journal de la Societe Francaise de Statistique*, 153(2):79–99, December 2012.

[126] Liyao Ma, Bin Sun, and Li Ziyi. Bagging Likelihood-Based Belief Decision Trees. In *20th International Conference on Information Fusion (FUSION)*, pages 1–6, Xi-An, China, July 2017.

[127] Tawfik A. Moahmed, Neamat El Gayar, and Amir F. Atiya. Forward and Backward Forecasting Ensembles for the Estimation of Time Series Missing Data. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, Lecture Notes in Computer Sci-

ence, pages 93–104. Springer, Cham, October 2014.

[128] Wang Xuesong, Guo Qiang, Li Shanshan, and Cao Rongfei. Design and Implementation of School Hospital Information Analysis and Mining System. *Applied Science, Materials Science and Information Technologies in Industry*, 513:498–501, 2014.

[129] Narasimha Prasad, Prudhvi Kumar, and M. M. Naidu. An Approach to Prediction of Precipitation Using Gini Index in SLIQ Decision Tree. In *Fourth International Conference on Intelligent Systems, Modelling and Simulation*, pages 56–60, Bangkok, 2013.

[130] Advait S. Bhatt. Comparative Analysis of Attribute Selection Measures Used for Attribute Selection in Decision Tree Induction. In *International Conference on Radar, Communication and Computing*, pages 230–234, SKP Engineering College Tiruvannamalai, 2012.

[131] Taneja Shweta, Raheja Shipra, and Kaur Savneet. Using Supervised Learning and Comparing General and ANTI-HIV Drug Databases Using Chemoinformatics. In *Pattern Recognition and Machine Intelligence, Proceedings*, pages 177–183. Springer-Verlag, Berlin, 2009.

[132] Mingbo Zhao, Choujun Zhan, Zhou Wu, and Peng Tang. Semi-Supervised Image Classification Based on Local and Global Regression. *IEEE Signal Processing Letters*, 22(10):1666–1670, October 2015.

[133] Andrew Honig, Andrew Howard, Eleazar Eskin, and Sal Stolfo. Adaptive Model Generation: An Architecture for Deployment of Data Mining-based Intrusion Detection Systems. *Applications of data mining in computer security*, 8720:153–194, 2002.

[134] Yiannis Kamarianakis, H. Oliver Gao, and Poulicos Prastacos. Characterizing regimes in daily cycles of urban traffic using smooth-transition regressions. *Transportation Research Part C-Emerging Technologies*, 18(5):821–840, October 2010.

[135] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Statistical methods for detecting nonlinearity and non-stationarity in univariate short-term time-series of traffic volume. *Transportation Research Part C-Emerging Technologies*, 14(5):351–367, October 2006.

[136] Robert M Kunst. Multivariate forecasting methods. In *Econometric Forecasting*, pages 31–40. Institute for Advanced Studies, Vienna, September 2012.

[137] Sungbin Cho, Hyojung Hong, and Byoung-Chun Ha. A hybrid approach based on the combination of variable selection using decision trees and case-based reasoning using the Mahalanobis distance: For bankruptcy prediction. *Expert Systems with Applications*, 37(4):3482–3488, April 2010.

[138] Jagdish Krishnaswamy, Kamaijit S. Bawa, K. N. Ganeshaiah, and M. C. Kiran. Quantifying and mapping biodiversity and ecosystem services: Utility of a multi-season NDVI based Mahalanobis distance surrogate. *Remote Sensing of Environment*, 113(4):857–867, April 2009.

[139] Yong Zhang, Dan Huang, Min Ji, and Fuding Xie. Image segmentation using PSO and PCM with Mahalanobis distance. *Expert Systems with Applications*, 38(7):9036–9040, July 2011.

[140] J. M. Cunderlik and D. H. Burn. Switching the pooling similarity distances: Mahalanobis for Euclidean. *Water Resources Research*, 42(3):3409, March 2006.

[141] O. Farber and R. Kadmon. Assessment of alternative approaches for bioclimatic modeling with special emphasis on the Mahalanobis distance. *Ecological Modelling*, 160(1-2):115–130, February 2003.

[142] Richard J Larsen and Morris L Marx. Hypothesis Testing. In *An Introduction to Mathematical Statistics and Its Applications*. Pearson, Boston, 5 edition, 2012.

[143] Yuan Qi and Brian L Smith. Identifying nearest neighbors in a large-scale incident data archive. *Transportation Research Record*, 1879(1):89–98, 2004.

[144] Huang Zhijun and Xia Chuangwen. A Kind of Algorithms for Euclidean Distance-Based Outlier Mining and its Application to Expressway Toll Fraud Detection. In K. Loach, editor, *International Asia Conference on Informatics in Control, Automation and Robotics*, pages 414–417, Los Alamitos, 2009.

[145] C. R. Maurer, R. S. Qi, and V. Raghavan. A linear time algorithm

for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, February 2003.

[146] Jorma Laurikkala, Martti Juhola, Erna Kentala, N Lavrac, S Miksch, and B Kavsek. Informal identification of outliers in medical data. In *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pages 20–24, Berlin, Germany, May 2000. Citeseer Press.

[147] P. Filzmoser, R. G. Garrett, and C. Reimann. Multivariate outlier detection in exploration geochemistry. *Computers & Geosciences*, 31(5):579–587, June 2005.

[148] Peter J Rousseeuw and Bert C Van Zomeren. Unmasking multivariate outliers and leverage points. *Journal of the American Statistical Association*, 85(411):633–639, 1990.

[149] Peter J Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[150] Alfred Krzywicki and Wayne Wobcke. Exploiting Concept Clumping for Efficient Incremental E-Mail Categorization. *Advanced Data Mining and Applications Pt II*, 6441:244–258, 2010.

[151] Carlos C. Sun and Venkat Chilukuri. Dynamic Incident Progression Curve for Classifying Secondary Traffic Crashes. *Journal of Transportation Engineering*, 136(12):1153–1158, December 2010.

[152] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 1993.

[153] Yeunjoo E. Song, Catherine M. Stein, and Nathan J. Morris. Strum: An R package for structural modeling of latent variables for general pedigrees. *Bmc Genetics*, 16:35, April 2015.

[154] Javier Palarea-Albaladejo and Josep Antoni Martin-Fernandez. zCompositions - R Package for multivariate imputation of left-censored data under a compositional approach. *Chemometrics and Intelligent Laboratory Systems*, 143:85–96, April 2015.

[155] Crt Ahlin, Dasa Stupica, Franc Strle, and Lara Lusa. Medplot: A Web Application for Dynamic Summary and Analysis of Longi-

tudinal Medical Data Based on R. *Plos One*, 10(4):121760, April 2015.

[156] Simon Janos and Goran Martinović. Web based distant monitoring and control for greenhouse systems using the Sun SPOT modules. In *7th International Symposium on Intelligent Systems and Informatics*, pages 165–169, September 2009.

[157] J. R. Alder and S. W. Hostetler. Web based visualization of large climate data sets. *Environmental Modelling & Software*, 68:175–180, June 2015.

[158] Rui Wang, Denghua Zhong, Yuankun Zhang, Jia Yu, and Mingchao Li. A Multidimensional Information Model for Managing Construction Information. *Journal of Industrial and Management Optimization*, 11(4):1285–1300, October 2015.

[159] Winston Chang, Joe Cheng, J. J. Allaire, Yihui Xie, and Jonathan McPherson. Shiny: Web Application Framework for R. Technical report, Cran, February 2015.

[160] A. M. de Souza, R. S. Yokoyama, G. Maia, A. Loureiro, and L. Villas. Real-time path planning to prevent traffic jam through an intelligent transportation system. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 726–731, June 2016.

[161] Shi An, Tao Zhang, Xinming Zhang, and Jian Wang. Unrecorded Accidents Detection on Highways Based on Temporal Data Mining. *Mathematical Problems in Engineering*, 2014(852495):1–7, 2014.

[162] Liyao Ma, Sébastien Destercke, and Yong Wang. Online active learning of decision trees with evidential data. *Pattern Recognition*, 52(Supplement C):33–45, April 2016.

[163] Wilfredo Palma. Missing Values and Outliers. In *Time Series Analysis*, page 424. John Wiley & Sons, April 2016.

[164] S. I. Khan and S. G. Ritchie. Statistical and neural classifiers to detect traffic operational problems on urban arterials. *Transportation Research Part C-Emerging Technologies*, 6(5-6):291–314, OCT-DEC 1998.

[165] Yiming Gu, Zhen Qian, and Feng Chen. From Twitter to detector: Real-time traffic incident detection using social media data. *Transportation Research Part C-Emerging Technologies*, 67:321–342, June

2016.

[166] Marco A. F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, June 2014.

[167] Liyao Ma, Bin Sun, and Chunyan Han. Learning Decision Forest from Evidential Data: The Random Training Set Sampling Approach. In *4th International Conference on Systems and Informatics (ICSAI)*, Hangzhou, China, November 2017.

[168] Chung Chen and Lon-Mu Liu. Joint Estimation of Model Parameters and Outlier Effects in Time Series. *Journal of the American Statistical Association*, 88(421):284–297, March 1993.

[169] Javier Lopez-de Lacalle. Tsoutliers - R Package for Detection of Outliers in time Series, May 2017.

[170] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 27(3):1–22, July 2008.

[171] Rob J Hyndman and George Athanasopou los. STL decomposition. In *Forecasting: Principles and Practice*. OTexts, 1st edition, October 2013.

[172] Jordan Hochenbaum, Owen S. Vallis, and Arun Kejariwal. Automatic Anomaly Detection in the Cloud Via Statistical Learning. *arXiv:1704.07706 [cs]*, April 2017.

[173] Bernard Rosner. Percentage Points for a Generalized ESD Many-Outlier Procedure. *Technometrics*, 25(2):165–172, May 1983.

[174] Stephen Kelly and Khurshid Ahmad. Propagating Disaster Warnings on Social and Digital Media. In *Intelligent Data Engineering and Automated Learning – IDEAL 2015*, pages 475–484. Springer, Cham, October 2015.

[175] L. Bodrog, M. Kajo, S. Kocsis, and B. Schultz. A robust algorithm for anomaly detection in mobile networks. In *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–6, September 2016.

[176] K. Jensen, T. V. Do, H. T. Nguyen, and A. Arnes. Better Protection of SS7 Networks with Machine Learning. In *2016 6th International Conference on IT Convergence and Security (ICITCS)*, pages

1–7, September 2016.

[177] TRB. *Highway Capacity Manual (HCM)*. Transportation Research Board, Washington, DC, 2010.

[178] Andreas Ruckstuhl. Robust Fitting of Nonlinear Regression Models (nlrob) in Package robustbase: Basic Robust Statistics, November 2017.

[179] Bin Sun, Liyao Ma, Wei Cheng, Wei Wen, Prashant Goswami, and Guohua Bai. An Improved k-Nearest Neighbours Method for Traffic Time Series Imputation. In *Chinese Automation Congress (CAC)*, Jinan, China, October 2017. IEEE.

[180] Nvidia. *CUDA Programming Guide 8.0*. Nvidia, September 2016.

[181] Nitin Bhatia and Vandana Ashev. Survey of Nearest Neighbor Techniques. *International Journal of Computer Science and Information Security*, 8(2):302–305, July 2010.

[182] Hrvoje Markovic, Bojana Dalbelo Basic, Hrvoje Gold, Fang Dong, and Kaoru Hirota. GPS Data-based Non-parametric Regression for Predicting Travel Times in Urban Traffic Networks. *PROMET-Traffic & Transportation*, 22(1):1–13, 2010.

[183] 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 *18th IEEE International Conference on Intelligent Transportation Systems*, pages 2262–2267, Gran Canaria Canary Islands, Spain, September 2015. IEEE.

[184] Robert Kohn, Michael Smith, and David Chan. Nonparametric regression using linear combinations of basis functions. *Statistics and Computing*, 11(4):313–322, October 2001.

[185] Xueying Zhang and Qinbao Song. Predicting the number of nearest neighbors for the k-NN classification algorithm. *Intelligent Data Analysis*, 18(3):449–464, 2014.

[186] Himanshu Singh, Aditya Desai, and Vikram Pudi. PAGER: Parameter less, Accurate, Generic, Efficient kNN-Based Regression. In P. G. Bringas, A. Hameurlain, and G. Quirchmayr, editors, *Database and Expert Systems Applications, Pt 2*, volume 6262, pages 168–176. Springer-Verlag Berlin, Berlin, 2010.

[187] S. Oh, Y. J. Byon, and H. Yeo. Improvement of Search Strategy

With K-Nearest Neighbors Approach for Traffic State Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1146–1156, April 2016.

[188] Harshit Dubey and Vikram Pudi. CLUEKR: CLUstering Based Efficient kNN Regression. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 450–458, Gold Coast, Australia, April 2013. Springer.

[189] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, University of Warwick: Proceedings of a Symposium Held at the University of Warwick*, pages 366–381. Springer, 1981.

[190] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. Geometry from a Time Series. *Physical Review Letters*, 45(9):712–716, September 1980.

[191] S. Innamaa. Self-adapting traffic flow status forecasts using clustering. *IET Intelligent Transport Systems*, 3(1):67–76, March 2009.

[192] B. L. Smith and R. K. Oswald. Meeting Real–Time Traffic Flow Forecasting Requirements with Imprecise Computations. *Computer-Aided Civil and Infrastructure Engineering*, 18(3), May 2003.

[193] Dionysios Kehagias, Athanasios Salamanis, and Dimitrios Tzovaras. Speed pattern recognition technique for short-term traffic forecasting based on traffic dynamics. *IET Intelligent Transport Systems*, 9(6):646–653, August 2015.

[194] Bin Yu, Xiaolin Song, Feng Guan, Zhiming Yang, and Baozhen Yao. K-Nearest Neighbor Model for Multiple-Time-Step Prediction of Short-Term Traffic Condition. *Journal of Transportation Engineering*, 142(6):04016018, 2016.

[195] F Habtemichael, M Cetin, and K Anuar. Methodology for quantifying incident-induced delays on freeways by grouping similar traffic patterns. In *Transportation Research Board 94th Annual Meeting*, Washington DC, 2015.

[196] Pinlong Cai, Yunpeng Wang, Guangquan Lu, Peng Chen, Chuan Ding, and Jianping Sun. A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting. *Trans-*

portation Research Part C: Emerging Technologies, 62:21–34, January 2016.

[197] S Moghaddam, Reza Noroozi, and B Hellinga. Real Time Prediction of Roadway Travel Times for Traveller Information and Proactive Control. In *OTC & CITE Joint Conference, Kitchener, Ontario, Canada*, 2014.

[198] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[199] Elia Liitiainen, Michel Verleysen, Francesco Corona, and Amaury Lendasse. Residual variance estimation in machine learning. *Neurocomputing*, 72(16-18):3692–3703, October 2009.

[200] Ahmad Basheer Hassanat, Mohammad Ali Abbadi, Ghada Awad Altarawneh, and Ahmad Ali Alhasanat. Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach. *International Journal of Computer Science and Information Security*, 12(8):33–39, September 2014.

[201] Edward Chung. Classification of traffic pattern. In *10th World Congress on Intelligent Transport Systems*, volume 10, page 11, Brussels, November 2003.

ABSTRACT

Intelligent transportation systems (ITS) are becoming more and more effective. Robust and accurate short-term traffic prediction plays a key role in modern ITS and demands continuous improvement. Benefiting from better data collection and storage strategies, a huge amount of traffic data is archived which can be used for this purpose especially by using machine learning.

For the data preprocessing stage, despite the amount of data available, missing data records and their messy labels are two problems that prevent many prediction algorithms in ITS from working effectively and smoothly. For the prediction stage, though there are many prediction algorithms, higher accuracy and more automated procedures are needed.

Considering both preprocessing and prediction studies, one widely used algorithm is k-nearest neighbours (kNN) which has shown high accuracy and efficiency. However, the general kNN is designed for matrix instead of time series which lacks the use of time series characteristics. Choosing the right parameter values for kNN is problematic due to dynamic traffic characteristics. This thesis analyses kNN based algorithms and improves the prediction accuracy with better parameter handling using time series characteristics.

Specifically, for the data preprocessing stage, this work introduces gap-sensitive windowed kNN (GSW-kNN) imputation. Besides, a Mahalanobis distance-based algorithm is improved to support correcting and complementing label information. Later, several automated and dynamic procedures are proposed and different strategies for making use of data and parameters are also compared.

Two real-world datasets are used to conduct experiments in different papers. The results show that GSW-kNN imputation is 34% on average more accurate than benchmarking methods, and it is still robust even if the missing ratio increases to 90%. The Mahalanobis distance-based models efficiently correct and complement label information which is then used to fairly compare performance of algorithms. The proposed dynamic procedure (DP) performs better than manually adjusted kNN and other benchmarking methods in terms of accuracy on average. What is better, weighted parameter tuples (WPT) gives more accurate results than any human tuned parameters which cannot be achieved manually in practice. The experiments indicate that the relations among parameters are compound and the flow-aware strategy performs better than the time-aware one. Thus, it is suggested to consider all parameter strategies simultaneously as ensemble strategies especially by including window in flow-aware strategies.

In summary, this thesis improves the accuracy and automation level of short-term traffic prediction with proposed high-speed algorithms.

ISSN: 1653-2090
ISBN: 978-91-7295-360-4

2018:10