CROSS DOMAIN DATA ANALYTICS FOR URBAN COMPUTING

YUQI WANG

PhD

The Hong Kong Polytechnic University

2018

The Hong Kong Polytechnic University
Department of Computing

Cross Domain Data Analytics for Urban Computing

Yuqi WANG

A thesis submitted in partial fulfilment of the requirements

for the degree of Doctor of Philosophy

April 2018

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Yuqi WANG (Name of student)

Abstract

With the rapid development of information technologies, we are entering the era of big data. Large amount of data in urban spaces are collected from various domains such as transportation, logistics, Point of Interests (POI), etc. The data reflect different aspects of cities in various ways, offering great opportunities for better understanding of the city's operation, and optimization of the infrastructure. Effective data analytics is the key to unlock the power of these big data. Although previous works mostly focus on data from single domain, ***Cross Domain Data Analytics*** is attracting increasing attention and lies at the core of many urban problems and applications.

Cross domain data analytics offers two additional opportunities than traditional single domain data analytics. First, it provides a more comprehensive picture about the studied problems based on the information from different angles, which helps gain new insights by discovering the correlations among cross-domain datasets. Second, it improves decision making by complementing data sources for joint analysis, especially for the cases where data are insufficient in some domains. Meanwhile, urban computing aims at utilizing urban big data, typically from different domains, to facilitate important urban operations such as traffic management, energy reduction and so on. In this way, urban computing offers a perfect application scenario for cross domain data analytics. Thus, this thesis focuses on ***Cross Domain Data Analytics for Urban Computing***, studies the problem of jointly analyzing data from different domains to ***generate hidden insights*** and ***enable intelligent decision-making***, and proposes effective solutions to three important applications in urban computing for demonstration.

First, we study the problem of traffic congestion, and show how to jointly utilize data from three domains, namely GPS trajectories, road network and POI data

to generate insights. Previous work mainly focuses on the prediction of congestion and analysis of traffic flows, while the congestion correlation between road segments has not been studied yet. In this work, we propose a three-phase framework to explore the congestion correlation between road segments from multiple real world data. In the first phase, we extract congestion information on each road segment from GPS trajectories of over 10,000 taxis, define congestion correlation and propose a corresponding mining algorithm to find out all the existing correlations. In the second phase, we extract various features on each pair of road segments from road network and POI data. In the last phase, the results of the first two phases are input into several classifiers to predict congestion correlation. We further analyze the important features and evaluate the results of the trained classifiers through experiments. We found some important patterns that lead to a high/low congestion correlation, and they can facilitate building various transportation applications. In addition, we found that traffic congestion correlation has obvious directionality and transmissibility.

Second, we study the problem of order response time prediction to enable intelligent decision-making in logistics services by jointly considering both order historical records and driver GPS trajectories from two different domains. Accurate prediction of order response time would not only facilitate decision making on order dispatching, but also pave ways for applications such as supply-demand analysis and driver scheduling, leading to high system efficiency. In this work, we forecast order response time on current day by fusing data from order history and driver historical locations. Specifically, we propose Coupled Sparse Matrix Factorization (CSMF) to deal with the heterogeneous fusion and data sparsity challenges raised in this problem. CSMF jointly learns from multiple heterogeneous sparse data through the proposed weight setting mechanism therein. Experiments on real-world datasets demonstrate the effectiveness of our approach, compared to various baseline methods. The performances of many variants of the proposed method are also presented to show the effectiveness of each component.

Third, we extend the previous method to incorporate more context information

by proposing a Coupled Weighted Tensor-matrix Factorization (CWTF) for accurate prediction on order accepting probabilities of van drivers, which would facilitate efficient order dispatching and improve user experience. However, it is difficult to handle the inherent heterogeneous data fusion, sparsity and efficiency challenges simultaneously. In this work, we propose a three-stage framework with a Coupled Weighted Tensor-matrix Factorization method for order accepting probability prediction in logistics services. Specifically, orders are first grouped into clusters to enrich the sparse interactions between orders and drivers; then an accepting probability tensor with the three dimensions of driver, order cluster, and time is generated by a tensor-matrix factorization method that fuses order characteristics and driver behaviors in an efficient way; finally given a new order, the accepting probability of each driver is efficiently predicted by directly retrieving from the learned tensor. The experiment results on a large dataset from a famous app-based logistics platform, demonstrate the superiority of the proposed method against various baseline methods.

List of Publications

1. **Yuqi Wang**, Jiannong Cao, Lifang He, Wengen Li, Lichao Sun, Philip S. Yu, "Coupled Sparse Matrix Factorization for Response Time Prediction in Logistics Services", in CIKM, 2017.

2. **Yuqi Wang**, Jiannong Cao, Wengen Li, Tao Gu and Wenzhong Shi, "Exploring Traffic Congestion Correlation from Multiple Data Sources", Pervasive and Mobile Computing Journal (PMCJ), DOI: 10.1016/j.pmcj.2017.03.015, 2017.

3. **Yuqi Wang**, Jiannong Cao, Wengen Li and Tao Gu, "Mining Traffic Congestion Correlation between Road Segments on GPS Trajectories", in SMARTCOMP, 2016. [**Best Paper Award**]

4. **Yuqi Wang**, Wengen Li, Jiannong Cao, Lifang He and Senzhang Wang, "Coupled Weighted Tensor-matrix Factorization for Accepting Probability Prediction in Logistics Services", under first review, IEEE Transactions on Knowledge and Data Engineering (TKDE).

5. Zimu Zheng, **Yuqi Wang**, Quanyu Dai, Huadi Zheng and Dan Wang, "Automatic Multi-task Learning with Metadata Clustering", submitted to CIKM, 2018.

6. Lichao Sun, **Yuqi Wang**, Bokai Cao, Philip S. Yu, Witawas Srisa-an and Alex D Leow, "Sequential Keystroke Behavioral Biometrics for Mobile User Identification via Multi-view Deep Learning", in ECML-PKDD, 2017.

7. Jiannong Cao, Shailey Chawla, **Yuqi Wang** and Hanqing Wu, "Programming Platforms for Big Data Analysis", a chapter in Handbook of Big Data Technologies (Springer), Albert Y. Zomaya and Sherif Sakr (Eds.), 2017.

8. Hongliang Lu, Jiannong Cao, Shailey Chawla and **Yuqi Wang**, "A Cluster Based Approach for Task Scheduling Across Multiple Programming Systems", in CloudCom-Asia, 2016.

Acknowledgements

First of all, I would like to express the greatest gratitude to my supervisor, Professor Jiannong Cao. He offers a golden opportunity and many insightful advices to me to pursue my passion in research and broaden my horizon. His global outlook and perpetual enthusiasm on impactful research always inspire me when times were tough. His profound thoughts and critical judgement also motivate me to strengthen my mind and get smarter, more careful and considerate. It is really my honor and privilege to have his continuous support, guidance and patience.

I also would like to thank Professor Philip S. Yu for the constructive suggestions and warm encouragement during my visit to his lab. Meanwhile, I am obliged to Professor Xiaoli Ding for the significant help and kind care. I learned a lot from them in both research and life.

My special thanks go to Dr. Wengen Li and Dr. Lifang He. I benefited so much in the numerous meetings with them during my Ph.D. study. In addition, I thank Dr. Guanqing Liang, Dr. Lei Yang, Dr. Xuefeng Liu, Dr. Senzhang Wang, Ms. Man-Ching AU and Ms. Wing-Yan AU for their kind help.

My dear friends, Mr. Jiaxing Shen, Mr. Zimu Zheng, Mr. Zilin Liang, Mr. Fei Jiang, Mr. Jiangyu Ding, Mr. Rui Liu, Mr. Zhaoyan Shen, Mr. Yu Yang, Dr. Tao Li, Dr. Bo Tang, Prof. Junbing Liang, Dr. Xiulong Liu, Mr. Lichao Sun, Dr. Bokai Cao, Dr. Chun-Ta Lu, Mr. Yang Yang, Mr. Linchuan Xu, Mr. Yaguang Huangfu, Mr. Jiaqi Wen, Mr. Hanqing Wu, Mr. Shan Jiang, Mr. Ruosong Yang, Miss Jia Wang, Miss Yanni Yang, Miss Tingting Liang, Miss Guixiang Ma, Miss Jiating Zhu, Mr. Muhui Jiang, Mr. Chun-Tung Li, Mr. Yuvraj Sahni and many others, I truly appreciate all kinds of support from you. Thanks for brightening my world.

Finally, I want to thank my parents and other family members. Without your love and backing, this journey would not have been possible.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

As the data are collected from ever more ubiquitous sources in various types, at very high velocity and a great volume, we are entering the era of big data. It may be difficult for existing technologies to manage and process big data, but big data also brings a big revolution and enable a wide range of new applications, since patterns and knowledge of the physical world as well as human society are becoming more detectable than ever before.

Specifically, big data generated by heterogeneous sources in urban spaces will lead to a better understanding of the citys operation, and new possibilities for optimization of the infrastructure. In addition, such data provides enormous opportunities for the study of general social phenomena, such as crime, entertainment patterns, or energy usage.

As a matter of fact, big data analytics for urban computing [109] has already sparked a great deal of interest around the world. Early successes include the policing in New York City and monitoring the traffic status in Singapore. Above all, when urban big data is used properly, urban computing can help tackle the challenges cities faced like pollution, energy, transportation problems. Thus, it is beneficial for people, environment and cities.

However, data analytics for urban computing is a challenging task in the complex and changeable environment. In addition, most of current research works focus on analyzing the data collected from a single domain, while there are actually multi-domain data existing in the urban spaces which make the analysis even harder to perform. There is not yet a systematic work on processing and analyzing cross domains data sets. The mostly close related work is data fusion and correlation mining in specific applications, aiming to extract correlation relationship between variables to understand the dependency between them. We envision the research paradigm of big data analytics for urban computing being horizontal cross domain analysis based on data collected from various sources in different domains, which is extended from current vertical intra-domain single source analysis.

The joint analysis of urban big data of different domains from multiple sources is very useful. It helps us to gain the hidden insights and enable intelligent decision-making. Specifically, cross domain data analytics has two advantages over traditional single domain data analytics. First, it offers a more comprehensive picture about the studied problems based on the information from different angles, where valuable insights can be discovered by analyzing the cross domain big datasets. For example, when we try to find out the reason of having difficulty to get taxis in rainy days by GPS taxi trajectories and weather conditions, most people might think the unavailability of taxis in rainy days is due to the high demand for services. However, according to a report based on data study comparing two months of weather satellite data with taxi GPS records, the true reason was that many taxi drivers did not work during rainstorms, mainly due to the high probability for accident. Second, it improves decision making by complementing data sources for joint analysis, especially for the cases where data are insufficient in some domains. For example, when recommending restaurants to customers, only the historical data about the restaurants such as types and price, are insufficient to achieve a satisfactory accuracy, along

with the data about customers such as tastes and income, the accuracy would be improved.

Based on the above observations, we propose the topic ***Cross Domain Data Analytics for Urban Computing***, and study the problem of jointly analyzing data from different domains to ***generate hidden insights*** and ***enable intelligent decision-making*** with three important applications in urban computing as discussed in Sec 1.2.

1.2 Research Framework and Scope

This research topic will focus on three important applications in urban computing to show how cross domain urban data can be utilized to generate hidden insights and enable intelligent decision making. Particularly, Considering the heterogeneity and complexity of cross domain urban data, two unique challenging issues arise.

- The heterogeneous nature of cross domain data: multiple sources of data are collected in urban computing in different forms, format and from different domains, e.g., GPS and Point of Interest (POI). The analytics should be taken based on the heterogeneous data. How to effectively analyze cross domain urban big data remains an open and challenging problem, which has not received sufficient attention.

- The complex correlations: different from single source big data, cross domain data usually have correlations among one another. How to model and discover correlation between datasets from different domains is a grand challenge.

Corresponding to the two challenges, in this thesis, the contributions are:

- We model and discover correlations among cross domain data in an important urban application, showing how to deal with the complexity of correlations to

generate hidden insights.

- We provide practical frameworks and solutions in two new urban applications through the fusion cross domain data, showing how to handle heterogeneous data to enable intelligent decision making.

Specifically, the studied three important applications demonstrate how the challenges can be addressed in different urban computing scenarios.

First, we jointly utilize cross domain data: GPS trajectories, road network and POI data to conduct traffic congestions analysis. We design a new metric for congestion correlation between road segments, and propose a three-phase framework to explore the correlations from multiple real world data. In the first phase, we extract congestion information on each road segment from GPS trajectories of over 10,000 taxis, define congestion correlation and propose a corresponding mining algorithm to find out all the existing correlations. In the second phase, we extract various features on each pair of road segments from road network and POI data. In the last phase, the results of the first two phases are input into several classifiers to predict congestion correlation. We further analyze the important features and evaluate the results of the trained classifiers through experiments. We found some important patterns that lead to a high/low congestion correlation, and they can facilitate building various transportation applications. In addition, we found that traffic congestion correlation has obvious directionality and transmissibility.

Second, we study the problem of order response time prediction to enable intelligent decision-making in logistics services by jointly considering both order historical records and driver GPS trajectories from two different domains. Accurate prediction of order response time would not only facilitate decision making on order dispatching, but also pave ways for applications such as supply-demand analysis and driver scheduling, leading to high system efficiency. In this work, we forecast order response

time on current day by fusing data from order history and driver historical locations. Specifically, we propose Coupled Sparse Matrix Factorization (CSMF) to deal with the heterogeneous fusion and data sparsity challenges raised in this problem. CSMF jointly learns from multiple heterogeneous sparse data through the proposed weight setting mechanism therein. Experiments on real-world datasets demonstrate the effectiveness of our approach, compared to various baseline methods. The performances of many variants of the proposed method are also presented to show the effectiveness of each component.

Third, we extend the previous method to incorporate more context information by proposing a Coupled Weighted Tensor-matrix Factorization (CWTF) for accurate prediction on order accepting probabilities of van drivers, which would facilitate efficient order dispatching and improve user experience. However, it is difficult to handle the inherent heterogeneous data fusion, sparsity and efficiency challenges simultaneously. In this work, we propose a three-stage framework with a Coupled Weighted Tensor-matrix Factorization method for order accepting probability prediction in logistics services. Specifically, orders are first grouped into clusters to enrich the sparse interactions between orders and drivers; then an accepting probability tensor with the three dimensions of driver, order cluster, and time is generated by a tensor-matrix factorization method that fuses order characteristics and driver behaviors in an efficient way; finally given a new order, the accepting probability of each driver is efficiently predicted by directly retrieving from the learned tensor. The experiment results on a large dataset from a famous app-based logistics platform, demonstrate the superiority of the proposed method against various baseline methods.

Figure 1.1 shows framework of this topic. This framework first acquires and stores data from different domains in urban spaces like transportation, POI, road network and weather. Then features are extracted from datasets of different domains, which may or may not have correlations among each other. For different applications, data

with correlations are jointly analyzed to acquire hidden insights and enable intelligent decision-making.

The scope of this topic is focused on data analytics and application side. Specifically, I will study the requirements and issues on cross domain data analytics for urban applications such as traffic congestion analysis in transportation, and supply demand analysis in logistics.

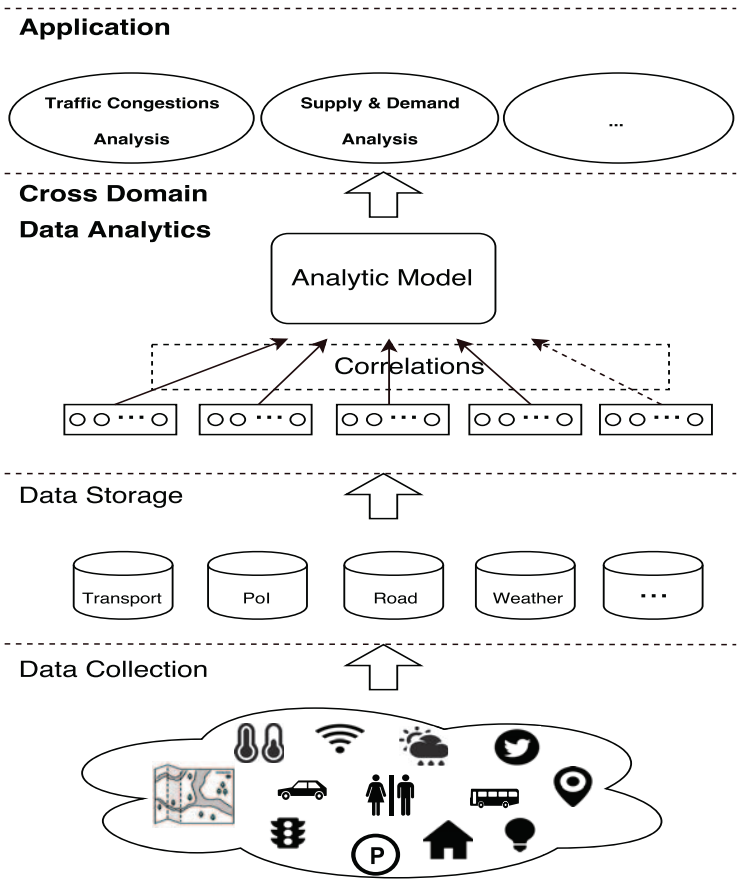


Figure 1.1: Research framework of cross domain data analytics for urban computing

1.3 Organization

As shown in Figure 1.2, this thesis is consisted of six chapters, where

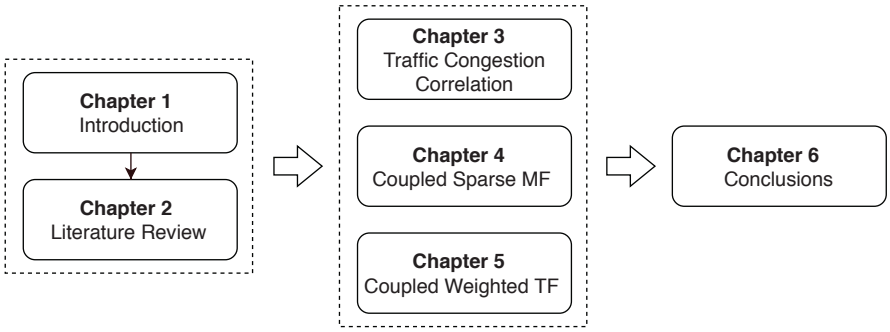


Figure 1.2: The organization of this thesis

- **Chapter 1** introduces the background, and highlights the research framework and scope of studying cross domain data analytics for urban computing;
- **Chapter 2** reviews the related work and discusses pros and cons the cross domain data analytics methods in different urban applications;
- **Chapter 3** presents the problem of traffic congestion correlations, and proposes a three-phase framework to generate hidden insights for congestions patterns based on data from GPS trajectories, POI and road network;
- **Chapter 4** proposes Coupled Sparse Matrix Factorization (CSMF) for order response time prediction in logistics services, and demonstrates its effectiveness in real-world data sets;
- **Chapter 5** proposes Coupled Weighted Tensor-matrix Factorization (CWTF) for order accepting probability prediction in logistics services, and shows its feasibility in real-world data sets;
- **Chapter 6** concludes the thesis.

Chapter 2

Literature Review

This chapter gives a review and categorization of existing works on cross domain data analytics for urban computing. Figure 2.1 shows the typical flowchart of the research topic. There are four stages of the processing, namely multi-domain raw data, feature representation, knowledge, applications. At the first stage, data from different domains are collected in different forms such as structured, semi-structured and unstructured; at feature representation stage, multi-domain raw data are pre-processed into different feature representation using different techniques such as data cleaning and feature extraction; at knowledge stage, different analytic models such as regression, classification and clustering can be applied to discover the knowledge and patterns in the data; at the last stage, the analytic results can be used by different applications for prediction, inference, profiling, etc. During this processing flow, the data from different domains can be jointly analyzed at both feature stage and knowledge stage. We categorize the existing works into feature-based and model-based methods based on where they jointly analyze multi-domain data. More details are presented in next sections.

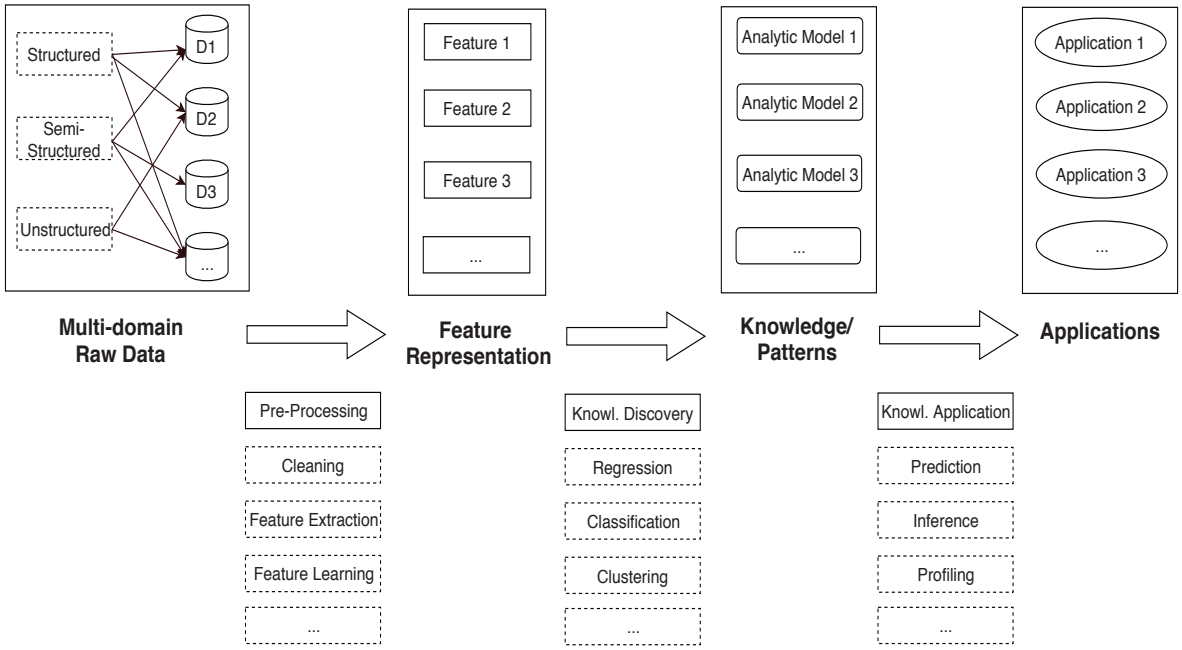


Figure 2.1: Flowchart of cross domain data analytics for urban computing

2.1 Feature-based

Feature-based methods jointly fuse data from different domains into a unified feature representation, and then the feature representation is used by the following analytic models. The existing works can be divided into three kinds, i.e., direct concatenation, domain knowledge-based and learning-based.

- **Direct concatenation:** As shown in Figure 2.2, the first kind of feature-based methods directly concatenate features from different domains into a unified feature representation. For example, Chen et al. [19] combine the indoor air quality data and weather data, and input them into neural network model to provide actionable suggestions to HVAC (heating, ventilation, and air conditioning) system. This kind of methods are easy to implement, but treating the features from different domains equally leads to loss of information on non-linear relationship as well as dependency and redundancy among the features.

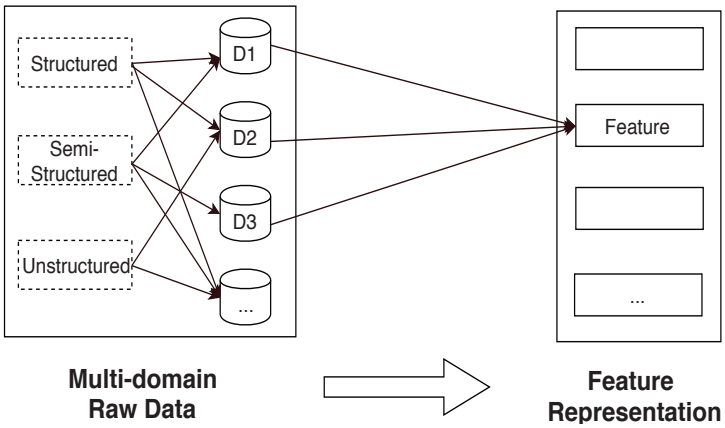


Figure 2.2: Direct concatenation

- **Domain knowledge-based:** As shown in Figure 2.3, the second kind of methods utilize domain knowledge when fusing data from different domains. For example, in [112], Beijing is divided into several grids using road network data, and then GPS trajectories of taxis are mapped into the grids for further analysis to detect the flawed road network planning. In this way, the two domains of data are combined into a unified representation. In [74], shops are partitioned into grids based on floor plan data, then WiFi data are mapped into the grids for network coverage analysis. This kinds of methods require some domain knowledge to improve the interpretability of the feature representation, and usually lead to better performance in following analytics.

- **Learning-based:** The third kind of methods are more advanced. As shown in Figure 2.4, they utilize a learning model to extract a unified representation from different domains of data. For example, Ngiam et al. [60] use a Deep Neural Network (DNN) based model to learn a unified feature representation from both audio and video features, and show its effectiveness in audio-visual speech classification tasks. Lin et al. [93] propose a network embedding model to

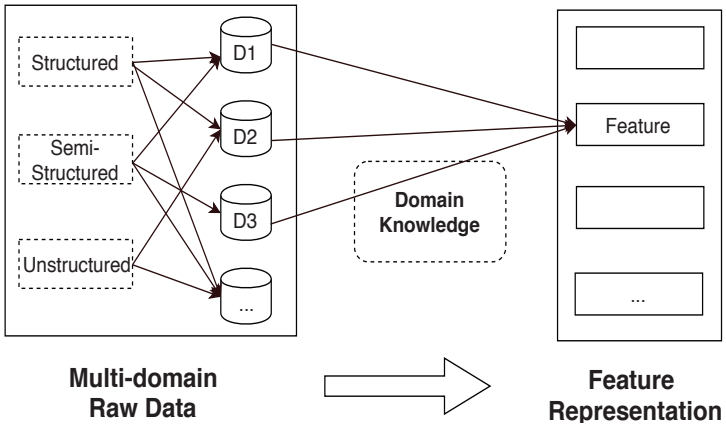


Figure 2.3: Domain knowledge-based

jointly combine both co-authorship network and word co-concurrence network into a unified representation for link prediction. This kinds of methods are able to discover the complex relations among the features and usually improve the efficiency by reducing the dimension of features. However, building the learning model require much efforts in both model designing and parameter tuning.

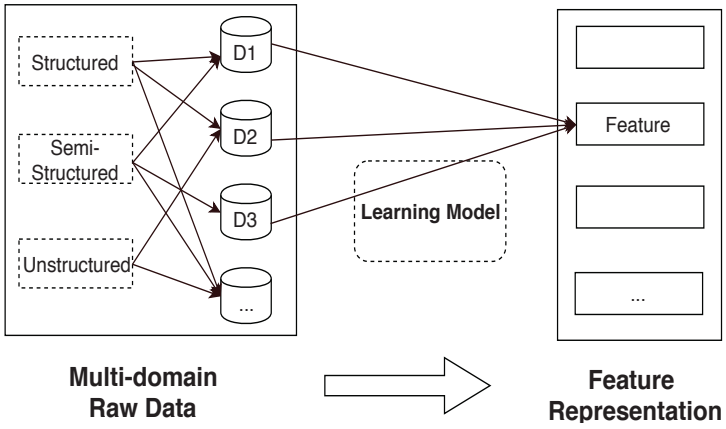


Figure 2.4: Learning-based representation

2.2 Model-based

Model-based methods fuse the analytic models and the features from different domains into a unified analytic model. There are mainly three kinds of existing works, i.e., multi-view learning-based, similarity-based and dependency-based.

- **Multi-view learning-based:** Multi-view learning-based methods treat data from different domains as different views on an object of interest. For example, the characteristics of an online user can be reflected in many aspects such as the browsing history, purchase records and click behaviors. Each of these aspects provides a view and some knowledge about the online user, and combing these complementary knowledge would lead to a more accurate user profiling. As show in Figure 2.5, different models are built and take features from different domains as input. Then these models are combined into an aggregated model for joint analysis. Several works are done on this regard. Zheng et al. [110] use a co-training based model to infer the air quality of Beijing in fine granularity based on data from transportation, road network, POIs, air quality and meteorology. They also predict the air quality in the next 48 hours with a Multi-Kernel Learning-based framework [113]. Liu et al. [52] propose a multi-task multi-view learning method to predict urban water quality, where they jointly consider six sources of data including water quality, hydraulic, road networks, pipe attributes, meteorology and POIs.

- **Similarity-based:** Similarity-based methods leverage the underlying similarity (or correlation) between different objects of interests to fuse data from different domains. Specifically, multiple similarities among objects can be discovered from multiple domains, and these similarities can complement each other, especially when some domains do not have sufficient data. For example,

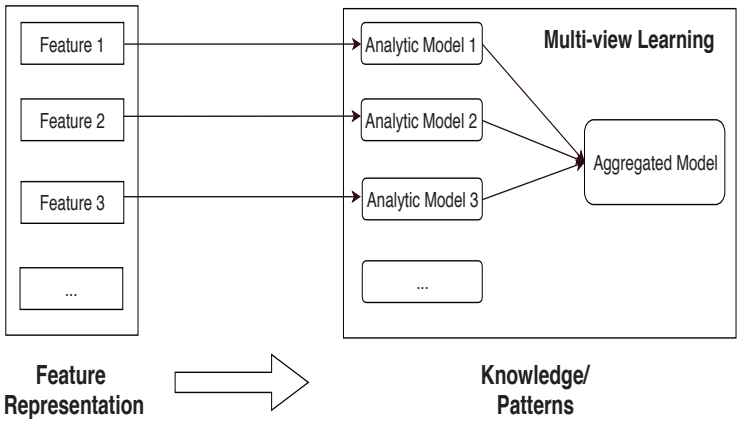


Figure 2.5: Multi-view learning-based methods

if there are new users without any data in a book recommender system, it is difficult to discover interesting books for them; however, if there are some data for all users in a movie recommender system, by measuring and utilizing the similarities between the new users and others in the movie recommender system, the interesting books can be discovered. As shown in Figure 2.6, features from different domains are fed into a unified similarity-based model, usually a coupled matrix/tensor factorization model [72, 111, 106]. In this way, the knowledge in different domains can reinforce each other based on similarities, and enhance the performance of analytic models. For example, a coupled matrix factorization method is proposed in [72] to infer the travel speed on each road by jointly considering the information from GPS trajectories of taxis, road network and POIs. With a coupled tensor factorization model, Zheng et al. [111] jointly analyze data from social media, road network, POIs and complaint data to estimate the noise situation in New York. GPS trajectories, POIs and social network data are also jointly analyzed for location and activity recommendation in [106].

- **Dependency-based:** Dependency-based methods model the dependency be-

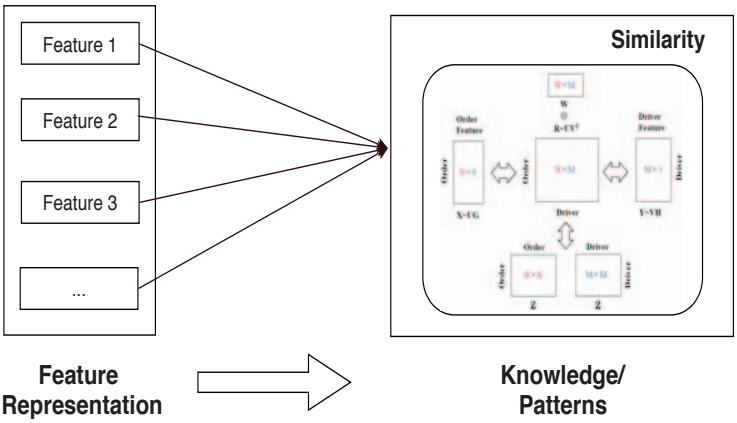


Figure 2.6: Similarity-based methods

tween different datasets using a graphic representation for analysis. As shown in Figure 2.6, features from different domains are fed into a unified dependency-based model, usually a probabilistic graphical model, where nodes represent random variables and edges represent the dependencies among them. The models can be divided into two groups, namely [59] Bayesian Networks and Markov Random Field. The model structure, namely the dependency among the variables, can be either pre-defined or automatically learned from data. The model learning process is to estimate the state of unobserved variables or probabilistic dependencies given the observed data. For example, Shang et al. [72] estimate the traffic volume of roads with a Bayesian Network jointly considering travel speed, road network, POIs and weather. Yuan et al. [100] infer the functions of each region in a city with a Latent Dirichlet Allocation-based model using GPS trajectories of taxis, road network and POIs. In [99], a non-parametric Bayesian method is proposed to fuse check-in and social account profile data, and model the relationships among different activities among a group of people. Fu et al. [30] model the dependency between geography data, taxi trajectories data and real estate data for the prediction of real estate value

in the future.

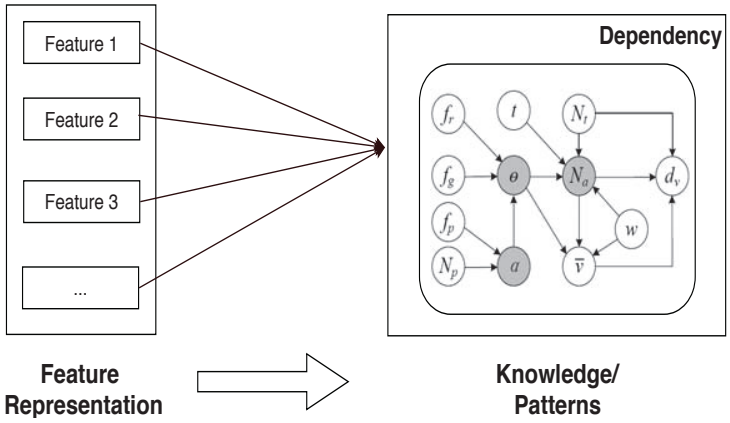


Figure 2.7: Dependency-based methods

2.3 Remarks

Table 2.1 shows a comparison among different methods for cross domain data analytics, where Knowl. represents the domain knowledge required, Gen. represents the ability to generalize to other applications, Interpr. represents the interpretability of methods for knowledge discovery, Perf. represents the application performance, and "L", "M", "H" denote Low, Medium, High, respectively. Generally, feature-based methods require less domain knowledge and have better ability to generalize, while model-based methods have better performance in the application domain, and better interpretability for knowledge discovery.

Table 2.1: Comparison among different methods for cross domain data analytics

Methods	**Knowl.**	**Gen.**	**Interpr.**	**Perf.**
Direct Concatenation	L	H	L	L
Knowledge-based	M	H	M	M
Learning-based	M	H	L	H
Multi-view Learning-based	H	L	M	H
Similarity-based	H	L	M	H
Dependency-based	H	L	H	H

Feature-based methods can also be combined with model-based methods. In [112], a city is partitioned into regions by major roads, then the GPS trajectories of taxicabs are mapped onto the regions to formulate a region graph, where a node is a region and an edge denotes the aggregation of commutes (by taxis in this case) between two regions. The region graph actually blends knowledge from the road network and taxi trajectories. By analyzing the region graph, research on traffic anomalies detection [18] is carried out. More details with a different categorization of cross domain data analytics methods can be found in [108].

Table 2.2 summarizes representative existing works on cross domain data analytics in different domains. Works in urban planning domain refer to those facilitating optimization of urban plan such as road network plan and function of regions; works in transportation domain refer to those offer insights for transportation system such as traffic analysis and anomaly detection; works in environment domain refer to those related to the environment such as air quality and noise situation; works in social application domain refers to those help improve social experience such as activity recommendation and social event prediction; works in economy and logistics refer to those related to the economy or supply demand analysis. Apparently, there have been quite a lot studies on urban planning, transportation, environment and social application, and very few studies on economy and logistics. Thus, in this thesis, we study cross domain data analytics for urban applications not only in domains that are very important, namely transportation, but also in domain that is less studied, namely logistics, to deal with both the existing challenges in well-studied domains and some new ones brought by requirements in other domains. Specifically, we propose a domain knowledge-based method and study from a new perspective on traffic congestion as an example application; we propose two similarity-based method and study the problems of response time and accepting probability prediction in an emerging logistic service as example applications.

Table 2.2: Representative existing works on cross domain data analytics in different domains

Domains	**Existing Works**
Urban Planning	[112, 100]
Transportation	[72, 112, 18, 63]
Environment	[19, 110, 113, 111, 52]
Social Application	[93, 106, 99, 92, 104]
Economy, Logistics	[30]

Chapter 3

Exploring Traffic Congestion Correlation from Multiple Data Sources

In this chapter, we propose a domain knowledge-based method to model and discover correlations among cross domain data. The reasons are that it generally has better interpretability compared to other feature-based method, and has little assumption on data compared to model-based method. As an example application, we study traffic congestion correlation, as it is both useful and a new perspective on congestion patterns.

3.1 Introduction

With the rapid process of urbanization, traffic congestion becomes an increasingly serious problem in more and more cities around the world. Understanding, alleviating, and further tackling traffic congestion have received urgent attentions from governments and their citizens. Much research work has been conducted to study congestion from different aspects, including traffic congestion prediction [5], traffic condition estimation [40], impact [39] and correlation [62] of traffic congestion and traffic flow propagation [53]. They provide many useful insights on traffic conges-

tions, which may facilitate the building of many practical applications.

However, the existing work typically assumes or ignores correlations [67], leaving the impact of correlated patterns on traffic congestion largely unknown. Analyzing and uncovering the correlated patterns in traffic congestion can reveal the insights of congestion such as what factors are correlated in congestion and how congestions propagate from one road to another. Furthermore, it can also facilitate building various applications including road planning, traffic condition prediction, impact analysis of congestion and etc. As such, both governments and individuals can benefit. For example, when a person is stuck in traffic congestion, the information about nearby congestion correlated road segments (i.e., these roads are likely to be congested as well) will be very useful since s/he can better estimate the travelling time, or possibly choose to bypass those roads to avoid congestion. Besides, with the information of congestion correlation between road segments acquired, governments are able to make better decisions on traffic light management, road planning, etc.

To fill the gap of existing work on congestion correlation analysis, we utilize multiple real world data to predict whether a road segment is correlated with another one in terms of congestion, where we can uncover some correlated congestion patterns from features on road segments. Thanks to the wide deployment of GPS devices and the widely available road and Point Of Interest (POI) information, we are able to obtain congestion information and features on road segments easily. To analyze the correlation between road segments, we apply a mining algorithm to find out all the existing correlations, and extract features on each road segment pair. We then build learning models based on classifiers to infer the correlated road segments from data. The models also help to identify some important features and correlated patterns.

To the best of our knowledge, we are the first to explore traffic congestion correlation from a classification perspective using real world datasets. In sum, our contributions are four folds:

- We propose a novel framework to explore traffic congestion correlation between road segments. The framework utilizes multiple sources of data to mine and analyze congestion correlation. In addition, the framework is general, and can be applied to other pairwise correlation analysis problems as well.

- We analyze the congestion correlation patterns based on our formulation, and found traffic congestion correlation has obvious directionality and transmissibility.

- We focus on congestion analysis of two peak periods during a day, train two corresponding models on several well-known classifiers to predict congestion correlation, and compare the results of different models on different feature sets.

- We predict congestion correlation and found some important patterns, such as congestions are very likely to propagate between trunk roads during the evening peak hours, which can facilitate the decision making for both individuals and governments.

3.2 Related Work

This section surveys the related work on traffic congestion prediction, traffic condition estimation, impact and correlation of congestion and traffic propagation. In [95], Yang formulated congestion prediction as a binary classification problem and applied feature selection techniques to reduce the dimensionality of data, yet still maintained the comparable accuracy. In [57], Min et al. proposed an approach based on the multivariate spatial-temporal autoregressive model to incorporate spatial and temporal characteristics for real-time traffic prediction, and found that congestion can change the traffic flow patterns. Gajewski et al. proposed a Bayesian-based approach in [31]

to estimate link travelling time correlation, and found that the heavier the congestion, the lower the correlation of travelling time between links. In [67], Rachtan et al. argued that correlation patterns among the traffic variables are largely unknown, while most of the work ignores congestion correlation or assumes correlation exits. Jenelius et al. estimated travelling time based on low frequency GPS data in [40], and demonstrated that there is significant correlation between segments and showed the feasibility of using low frequency GPS data for monitoring the performance of transport system. In [58, 53], the authors studied the traffic flow propagation by simulation. In [41, 24], the authors reviewed several approaches on traffic density estimation, detection and avoidance. In [62, 85, 39, 46], the authors studied the impact and correlation on weather, accident, employment, safety, respectively.

Different from the above work, we focus on congestion correlation between road segments, which can benefit various applications including traffic prediction, traffic light management, road planning and etc.

3.3 Overview

Figure 3.1 presents the framework of our work. In this framework, we utilize three sources of data i.e., GPS trajectory of taxis, road network and POI data to explore the congestion correlation between road segments. We divide the framework into following three phases.

[1] Congestion and correlation extraction: Extract congestion information on each road segment from GPS trajectories and road network data, define and mine congestion correlation between each road segment pair.

[2] Feature and sample generation: Extract various topological features and POI features from road network and POI data, respectively, and generate training samples on road segment pairs.

[3] Classification and analysis: Input the results of the first two phases into several classifiers to predict congestion correlation, and analyze the evaluation results for pattern discovery.

We design the framework in a way that it is general enough to be used for other pairwise correlation analysis problems by changing the specific data sources and implementing techniques such as feature extraction and correlation definition.

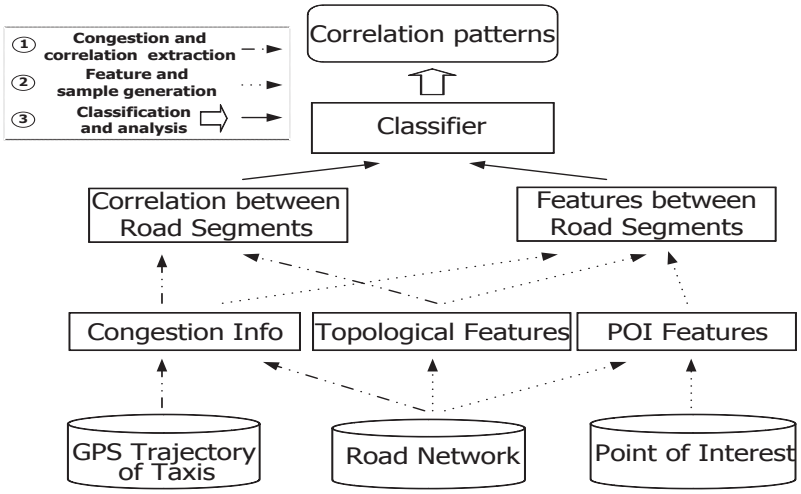


Figure 3.1: Framework of congestion correlation mining

3.4 Methodology

In this section, we describe the proposed framework in detail. Specifically, we first present the three domains of data sources we use, and then show how each phase of the framework works.

3.4.1 Data sources

Traffic congestion usually results from multiple factors. Intuitively, the underlying transportation infrastructure and human mobility are the major ones. Therefore, in this work, we exploit three data sources, i.e., road network, GPS trajectories of

taxis and POIs to cover these major factors. Concretely, road network describes the spatial topology of the transportation infrastructure; GPS trajectories of taxis contain the traffic information related to human mobility; and POIs implicitly convey some information about the mobility of people whose daily activities are relevant to them. We formalize these information as follows.

Definition 3.1 (Road network). *A road network is modelled as a directed graph* $G = (V, E)$*, where* $v_i \in V$ *represents an intersection of road segments, and* $e_{i,j} \in E$ *represents the direct road segment from* v_i *to* v_j*.*

Definition 3.2 (GPS point). *A GPS point, gp is denoted by a quadruple, i.e.,* $gp = (TaxiID, t, s, l)$*, where* $TaxiID$ *is the identifier of the taxi,* t *is the time at which this GPS point is sampled,* s *is the speed of the taxi, and* l *is the spatial location consisting of longitude and latitude.*

Definition 3.3 (GPS trajectory). *A GPS trajectory,* tr*, is consisted of a sequence of GPS points, i.e.,* $tr = (gp_1, gp_2, \ldots, gp_n)$*, where* n *is the length of* tr *and* $gp_i.t \leqslant gp_j.t$ *if* $i \leqslant j$*.*

Definition 3.4 (Point of Interest, POI). *A POI,* o_i*, is denoted by* $o_i = (ID, Cate, Lng, Lat)$*, where* ID *is the identifier of* o_i*,* $Cate$ *is the category of* o_i*, and* Lng *and* Lat *is the longitude and latitude, respectively, of the spatial location of* o_i*.*

3.4.2 Congestion and Correlation Extraction

In this phase, we first extract the congestion information from the GPS trajectories of taxis on each road segment. After that, with a definition of congestion correlation between road segments, we propose a mining algorithm to find out all the existing congestion correlation from data.

Congestion Extraction

To extract the congestion information, we need to first obtain the traffic information on each road segment. According to the definition of GPS trajectories, a GPS trajectory is a sequence of discrete spatial points. Thus, we need to map-match each GPS trajectory to the underlying road segments. In this work, we leverage the map-matching technique in [97]. Meanwhile, considering the time-consuming characteristic of map-matching operation, a spatial index R*-tree [9] is built on all road segments to accelerate the process of map-matching. After map-matching, each road segment is associated with a set of GPS points capturing the traffic information there.

To extract congestion information from traffic information, we divide a day into time slots, and obtain the traffic information T_r^t on road segment r in a specific time slot t, using the average speed of all GPS points on road segment r in time slot t as the proxy. Then we have the definition of congestion as follows.

Definition 3.5 (Congestion)**.** *A congestion on road segment r in a specific time slot t is denoted by C_r^t, and*

$$C_r^t = \begin{cases} 1 & \text{if } T_r^t \leqslant T_r * Ratio; \\ 0 & \text{otherwise.} \end{cases}$$

where T_r is the average speed of all GPS points in road segment r in all time, and Ratio is a parameter to set the speed threshold of congestion, and its settings will be discussed in Section 3.5.

We store the congestion information of a day in a congestion matrix as illustrated in Figure 3.2, where each row represents a road segment and each column represents a time slot.

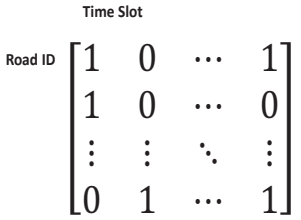


Figure 3.2: Congestion matrix

Correlation Extraction

To study how congestion occurs sequentially in terms of time, and consider the propagation rate of congestions in terms of space, as shown in figure 3.3, we define congestion correlation between two road segments as follows.

Definition 3.6 (Congestion correlation between two road segments)**.** *A congestion correlation from road segment a to road segment b, i.e., $Cor(a, b)$, occurs if the following requirements are satisfied:*

(1) a congestion occurs on road a at time t_0

(2) from time t_0 to $t_0 + t$, a congestion occurs on road b

(3) a and b are within a certain distance d

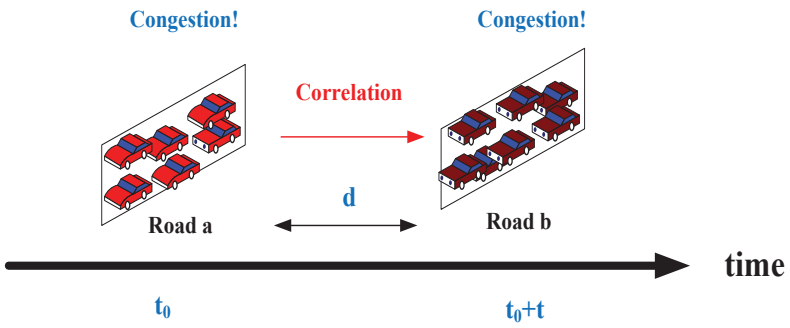


Figure 3.3: Congestion correlation

We propose Algorithm 3.1 to mine all congestion correlations in a designated time period, i.e., from t_{start} to t_{end}. The correlations are stored in a square matrix

R, where R_{ik} stores the occurrence count of congestion correlation between road segments i and k from t_{start} to t_{end}.

Algorithm 3.1 Congestion Correlation Mining

Input: the congestion matrix C, time threshold t, distance threshold d, start time slot t_{start} and end time slot t_{end};
Output: the correlation matrix R;

```
1:  R = 0; Create a vector cv of size C.rowNumber;
2:  for j = t_start to t_end do
3:      cv = 0;
4:      isFound = false;
5:      for i = 1 to C.rowNumber do
6:          if C[i][j] == 1 then
7:              if isFound == false then
8:                  for k = 1 to C.rowNumber do
9:                      for t = j+1 to j+t do
10:                         if C[k][t] == 1 then
11:                             cv[k] = 1;
12:                             break;
13:                 isFound = true;
14:             for k = 1 to C.rowNumber do
15:                 R[i][k] = R[i][k] + cv[k];
16: for i = 1 to C.rowNumber do
17:     for k = 1 to C.colNumber do
18:         if Dist(i, k) > d then
19:             R[i][k] = 0;
    return R;
```

In Algorithm 3.1, at each time slot j, for each congested road segment i, we retrieve all the congested road segments in next t time slots, and increase the occurrence count of correlation stored in $R_{i.}$. We use a vector cv to store the retrieved congested road segments, so that the retrieving process only executes once in each time slot, thus improving the efficiency of the algorithm. Then, we also check the distances of all pairs of road segments to make sure that the distance requirement is also satisfied. The time complexity of the proposed algorithm is $O(n^2m)$, where n is number of road segments and m is the number of time slots from t_{start} to t_{end}.

To further refine congestion correlation, we have the following definition about confidence of correlation.

Definition 3.7 (Correlation confidence)**.** *Correlation confidence from road segment a to road segment b, i.e., CC_{ab} indicates the confidence level of the congestion corre-*

lation and is computed as below:

$$CC_{ab} = \frac{occurrence\ count\ of\ Cor(a,b)}{No.\ of\ congestions\ at\ a}$$

With the correlation confidence, an analogy to the confidence in Association Analysis [79], we are able to identify some false positive and true positive correlations, and use them to conduct analysis more accurately in later phases.

We also perform some analysis on the found congestion correlations, which will be further discussed in Section 3.5.4 later.

3.4.3 Feature and Sample Generation

In this phase, we first extract various features on each road segment from road network and POI data, and then fuse the features of each road segment pair to generate training samples.

Feature Extraction

To extract features on each road segment from road network data, we consider not only their traditional features, including length, type, and degree, but also some advanced features, including betweenness and closeness. It is straightforward to extract those traditional features. Therefore, we will only detail how to extract the advanced features as follows.

In graph theory, betweenness is used to measure the importance of nodes in terms of the number of shortest paths passing them. The intuition is that a node is more important if more shortest paths go through it. The betweenness of a node v_i is computed with the following formula [21].

$$B(v_i) = \frac{1}{(N-1)(N-2)} \sum_{v_j, v_k \in V \wedge i \neq j \neq k} \frac{n_{jk}(v_i)}{n_{jk}} \tag{3.1}$$

where n_{jk} is the total number of shortest paths between nodes v_j and v_k, $n_{jk}(v_i)$ is the number of shortest paths between nodes v_j and v_k that pass node v_i.

Similarly, we compute the betweenness of a road segment, e_{i_1,i_2} as below (cf. Definition 3.1).

$$B(e_{i_1,i_2}) = \frac{1}{(N-1)(N-2)} \sum_{v_j,v_k \in V} \frac{n_{jk}(e_{i_1,i_2})}{n_{jk}} \tag{3.2}$$

where n_{jk} is the total number of shortest paths between nodes v_j and v_k, $n_{jk}(e_{i_1,i_2})$ is the number of shortest paths between nodes v_j and v_k that pass edge e_{i_1,i_2}.

According to [21], closeness centrality is used to measure the centrality of a node, v_i, in the network and is computed as below.

$$C(v_i) = \frac{N-1}{\sum_{j \in V \wedge j \neq i} netDis(v_i, v_j)} \tag{3.3}$$

where $netDis(v_i, v_j)$ is the network distance between nodes v_i and v_j.

To compute the closeness of a road segment, e_{i_1,i_2}, we change the formula above to the following form.

$$C(e_{i_1,i_2}) = \frac{N-1}{\sum_{e \in E \wedge e \neq e_{i_1,i_2}} netDis(e, e_{i_1,i_2})} \tag{3.4}$$

where $netDis(e, e_{i_1,i_2})$ is the network distance between edges e and e_{i_1,i_2} (cf. Eq.(3.6)).

To extract features from POI data on each road segment, we consider the total number of POIs, the number of POIs in each category, and the Term Frequency-Inverse Document Frequency(TF-IDF) value of each POI category. Specifically, we treat road segments as documents and POI categories as terms, and TF-IDF value indicates the importance of POI categories on road segments. Similar to [96], to compute TF-IDF value of the i-th POI category of a given road segment, we have

the following formula:

$$\text{TF-IDF}_i = \frac{n_i}{N} \times log \frac{R}{||\{r|the\ i\text{-}th\ POI\ category \in r\}||} \tag{3.5}$$

where n_i is the number of POIs in the i-th category and N is the total number of POIs on the given road segment. The first term calculates POI frequency in the given road segments, and the second term calculates the inverse segment frequency by taking the logarithm of a quotient, resulting from the number of road segments R divided by the number of segments which have POIs in i-th category.

The extracted features are summarized in Table 3.1.

Table 3.1: Extracted features on a road segment

Features	**Description**
length	the length of each road segment
degree	the degree of each road segment
type	type of road segments, e.g., motorway and trunk
$B(e_{i,j})$	the betweenness of the road segment $e_{i,j}$
$C(e_{i,j})$	the closeness of the road segment $e_{i,j}$
#POIs	the total number of POIs
#CatPOIs	the number of POIs in each category
POI-TF-IDF	the tf-idf value of each POI category

Sample Generation

To generate training samples, considering all features extracted on a road segment, we need to fuse the features of each road segment pair, and generate features for each pair.

For length, degree, betweenness, closeness and total number of POI, we calculate their differences between two road segments, and then add them to the features for each pair of road segments. We also add network distance and Pearson similarity of POI TF-IDF value distributions between two road segments into features for each pair.

Network distance between road segments e_{i_1,i_2} and e_{j_1,j_2} is computed based on the underlying road network (cf. Definition 3.1), i.e.,

$$netDis(e_{i_1,i_2}, e_{j_1,j_2}) = \min_{i \in \{i_1,i_2\}, j \in \{j_1,j_2\}} \{netDis(v_i, v_j)\} \tag{3.6}$$

where $netDis(v_i, v_j)$ is the length of the shortest path between nodes v_i and v_j. To accelerate the computation of network distance, we index road network G with CH (Contraction Hierarchy) [32] which organizes G in a hierarchy structure.

For each distinct ordered combination of two road types in a pair of road segments, we create a binary indicator variable to represent the existence of it between road segments. For example, a road segment type is 'trunk' and that of the other is 'primary', then the corresponding indicator variable that represents the existence of the ordered combination 'trunk→primary' is set to 1, and all other indicator variables of this ordered pair are set to 0. The idea of this design is to see how congestion correlation varies from one road type to another. Slightly different, for each distinct ordered combination of two POI categories, we create a variable to represent its importance level by calculating the product of TF-IDF values of the two categories on each pair of road segments. The idea of this design is to see how congestion correlation varies from one POI category to another.

Finally, we apply Min-Max scaling [3] to scale all the features for each pair of road segments into the range of [0, 1], which not only enhances the performance of the trained models, but also facilitates the process of analysis on feature importance later, since the trained models are not biased towards the features simply due to their large numeric range.

The features for each road segment pair are summarized in Table 3.2.

Table 3.2: Features for each road segment pair

Features	Description
Diff-Len	the difference of length
Diff-Degree	the difference of degree
Diff-B	the difference of betweenness
Diff-C	the difference of closeness
Diff-POI	the difference of the total number of POIs
$netDis$	the network distance
SimPOIs	Pearson similarity of POI TF-IDF value distributions
OrderedComb-types	the binary indicator variable for ordered combination of road types
OrderedComb-POI	the variable for ordered combination of POI categories

3.4.4 Classification and Analysis

In this phase, we input the results of the first two phases into several classifiers to predict congestion correlation, and analyze the evaluation results for pattern discovery.

Classification After the first two phases, we have all congestion correlations and extracted features for each pair of road segments. We now combine these two parts to generate training samples and build models for binary classification.

For any given pair of road segments, the models will predict whether there exists high congestion correlation between them. To refine and enhance the knowledge models learn from data, we set a threshold of Correlation confidence (cf. Definition 3.7) for positive class and negative class, respectively. Thus, we only keep those pairs of road segments, whose correlation confidence is higher than the threshold for positive class, and treat them as positive training samples; or lower than the threshold for negative class and higher than 0, and treat them as negative training samples.

Usually the classes of training samples are highly imbalanced, i.e., the samples in uncorrelated class are much more than those in the correlated class, which will impair

the performance of classifiers. To deal with this issue, we apply random majority undersampling (RUS) [84] to generate a balanced training samples, as it is effective and makes the model learning process more efficient.

Finally, we input the balanced training samples into well-known classifiers including Decision Tree (DT), Random Forest (RF), Logistic Regression (LR) and Support Vector Machine (SVM), and then evaluate the performance of the built models using classic metrics.

Analysis After the evaluation of the models, we analyze the built models for pattern discovery.

Feature importance indicates how important a feature is for the prediction of classifiers, which can help to identify important features and patterns during the analysis process. We employ different feature importance measures for different classifiers based on how those classifiers are built. For Decision Tree and Random Forest, we use Gini importances [13]. For Logistic Regression and Support Vector Machine, we consider the absolute values of feature coefficients as the measure of feature importance. Besides, we also generate some decision rules from Decision tree for better understanding of the analysis results.

With different training samples, we can build different models on different classifiers. The comparison of evaluation results, identified features and patterns among different models on different classifiers can also provide useful insights on congestion correlation between road segments.

3.5 Experiments

In this section, we present the details of datasets, experiment settings and results.

3.5.1 Datasets

In the experiments, we use three datasets, i.e., road network, POIs, and the GPS trajectories of taxis. All these three datasets are for Beijing, China and their details are elaborated as below.

The road network data is extracted from OpenStreetMap (OSM),* an open source online map. In Beijing road network, we have 109, 029 edges and 105, 030 nodes, with 13 categories of road types.

POI data set contains all kinds of physical objects in spatial space such as shops, schools, banks, and restaurants. Though we can also download POIs from OSM, the number of POIs there is quite small. To collect enough POIs, we obtain the POI data from a data sharing web site called DataTang.† This POI data set is comprised of 220, 137 POIs, which is divided into 21 categories.

We collect a large set of GPS trajectories of over 10, 000 taxis in Beijing for 30 consecutive days in 2012.

3.5.2 Data Filtering

After map-matching the GPS trajectories to the underlying road network, each road segment has a set of GPS records with time stamps. Considering that the goal of this work is to explore the congestion correlation between road segments, it is very important to obtain the traffic information on road segments as accurate as possible.

According to [96], more than 12 percent of traffic flow in Beijing is occupied by taxi trips. Therefore, it is reasonable for us to use the speeds of GPS records of taxis to approximate the real traffic congestion information. However, though we have over 10, 000 taxis, the number of speed samples of some road segments are still very small, which makes it difficult to capture the real traffic on these road segments. In

*https://www.openstreetmap.org

†http://www.datatang.com/

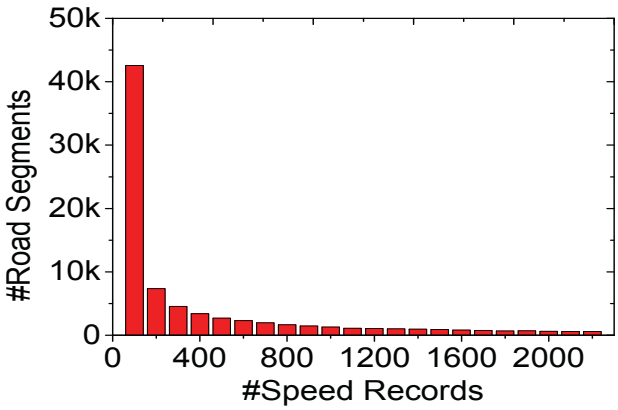
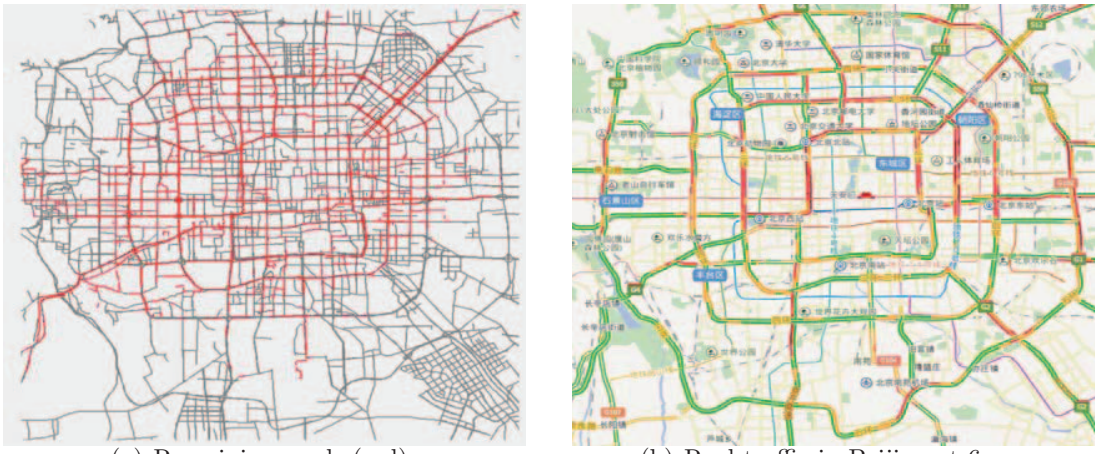


Figure 3.4: The distribution of the number of speed records on each road segment per day



(a) Remaining roads (red) (b) Real traffic in Beijing at 6pm

Figure 3.5: The remaining road segments after filtering and the real traffic in Beijing at 6pm.

the experiments, we divide a day into 10 minutes time slots, resulting in 144 time slots one day.

As shown in Figure 3.4, many road segments have speed records less than 100 per day, meaning that there is no traffic information in some time slots for many road segments. To alleviate the impact of data sparsity, we remove those segments that have less than 500 speed samples in a whole day. Finally, we get 3,004 road segments which have enough traffic information to support our further analysis. The

remaining roads are plotted with red color in Figure 3.5(a). Figure 3.5(b) illustrates the real traffic[‡]in Beijing at 6 pm, where red color represents busy traffic. Obviously, the remaining roads in Figure 3.5(a) cover most of the roads that have busy traffic in Figure 3.5(b). Therefore, it is reasonable for us to conduct analysis on remaining roads since our goal is to explore the congestion correlation between road segments.

3.5.3 Settings

In the experiments, we set the ratio in Definition 3.5 to 0.5, which is similar to [95], and compute the average speed on a road segment by all GPS records on the segment over 30 days.

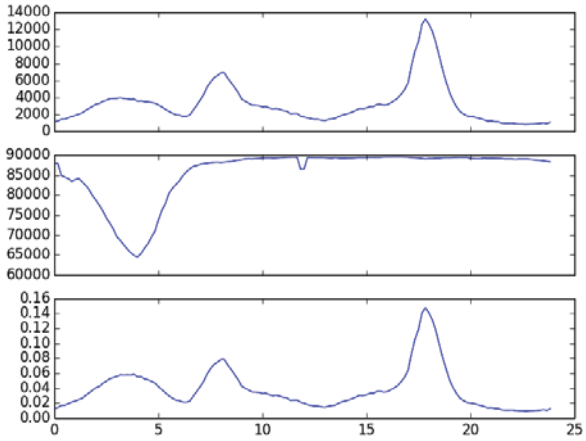


Figure 3.6: The number of congested roads, the number of roads with GPS records, and the proportion of congested roads

As illustrated in Figure 3.6, there are three sub-figures representing respectively the number of congested roads, the number of roads with GPS records and the proportion of congested roads from 0:00 to 23:59 over 30 days. We can see two peaks of the number of congested roads and the proportion, which corresponds to morning peak and evening peak in a day. Besides, during late night, the number of roads with GPS records dramatically falls, which is probably because there are

[‡]http://map.baidu.com

much fewer taxis travelling during this period. Since our goal is to explore congestion correlation between road segments, to ensure accurate traffic information extraction and enough congested roads for analysis, we focus on morning peak and evening peak. Specifically, we generate two sets of training samples from these two peaks in 30 days, respectively. The morning peak is from 7:30-9:00, and the evening peak is from 17:30 - 19:00.

Recall Definition 3.6, in the experiments, we set the time threshold $t = 2$, which is 20 minutes; $d = 5$ km, since the average speed of all GPS records in congested roads is about 16 km/h, and in 20 minutes the congestion can propagate at most around 5 km, thus reducing the false congestion correlation to some extent.

For the two sets of training samples, we set the threshold of correlation confidence for positive sample to 0.6, and the threshold for negative sample to 0.4. In the morning peak samples, 33,909 positive samples are collected, and 38,6875 negative samples are collected. After RUS, a balanced morning peak samples are generated with a total of 67,915 samples. In the evening peak samples, 53,968 positive samples are collected, and 495,808 negative samples are collected. After RUS, a balanced evening peak samples are generated with a total of 108,435 samples. For each sample, we initially generate 618 features as described in Table 3.2. Then we discard Diff-Len and Diff-Degree, since they hardly contribute to the performance of models during the experiments, and end up with 616 features for each sample.

We input the two sets of training samples with selected features, and train the two peak models on four well-known classifiers: Decision Tree (DT), Random Forest (RF), Logistic Regression (LR) and Support Vector Machine (SVM) [66] to predict congestion correlation. Then the average precision and recall computed by 10-fold cross validation are applied to evaluate the performance of the trained models. The precision and recall are defined as follows.

$$Precision = \frac{No.\ of\ predicted\ true\ correlations}{No.\ of\ all\ predicited\ correlations} \tag{3.7}$$

$$Recall = \frac{No.\ of\ predicted\ true\ correlations}{No.\ of\ all\ true\ correlations} \tag{3.8}$$

We summarize the experiment settings as shown in Table 3.3.

Table 3.3: Experimental settings

Ratio	0.5
t	20 minutes
d	5 km
Positive correlation confidence threshold	0.6
Negative correlation confidence threshold	0.4
Number of features	616
Morning Peak	679,151 samples
Evening Peak	108,435 samples
Classifiers	DT,RF,LR,SVM

3.5.4 Results and Analysis

Congestion Correlation and Transmissibility

After the data filtering, we show and analyze the congestion correlation heatmap of road segments here. As shown in Figure 3.7, the axes represent the numbers of road segments. Congestion correlation ranges from 0 to 1, and the denser the color, the higher congestion correlation are between two road segments. We can see that points with dense color are sparse overall, but we can also notice a clear diagonal line in the heatmap, as well as some vertical and horizontal bars. The sparse points indicate that congestion correlations are normally not high, which may be due to two reasons. First, the congestions themselves only occur on some road segments, which make it hard to find correlations among most road segments. Second, the number of road segment pairs that can be considered as having high congestion correlations is quite

small. Another observation of the diagonal line indicates that congestions on a road segment usually lasts longer than one time slot i.e., 10 minutes. Besides, the bars on the heatmap indicate that congestions on some road segments are more likely to propagate to multiple other road segments, which may be because they are hubs in the road network.

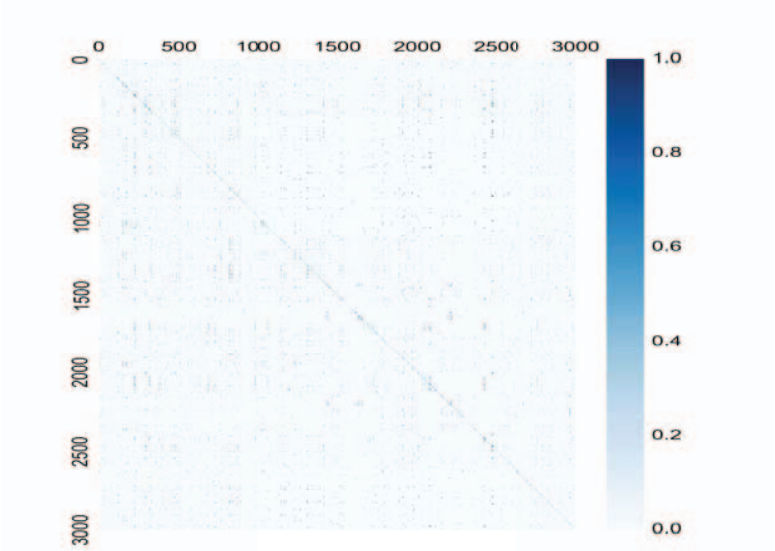
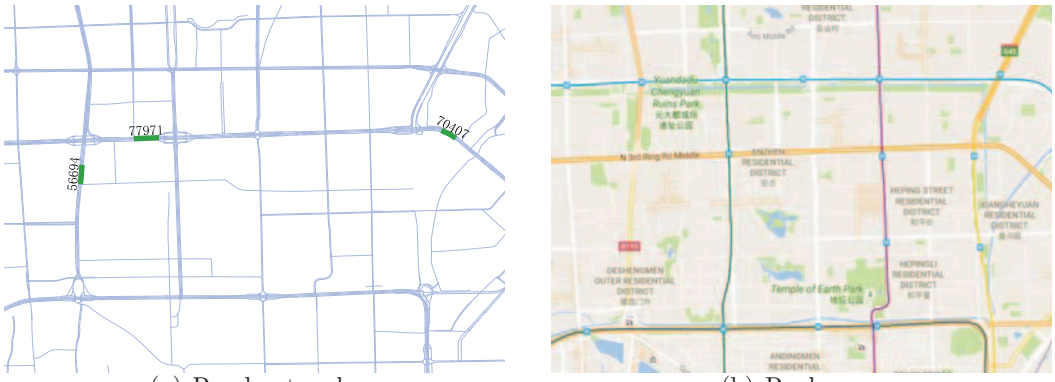


Figure 3.7: Congestion correlation heatmap of roads

As discussed above, some traffic congestions are highly correlated. Moreover, we found that these correlations have some kind of transmissibility. In other words, one traffic congestion may leads to a series of congestions at different locations. For example, the congestion of road segment A results in the congestion of road segment B, and road segment B further leads to the congestion of road segment C. In some cases, such congestion transmission can cover a quite wide area.

As illustrated in Figure 3.8, traffic congestions propagate among road segments 56694, 77971 and 70407 in the morning, where the numbers are road segment identifiers. According to Figure 3.8, the traffic congestion at road segment 56694 (an important motor way between the urban area and the suburbs of Beijing) will affect the traffic at road segment 77971 (a trunk road on the third ring road next to a

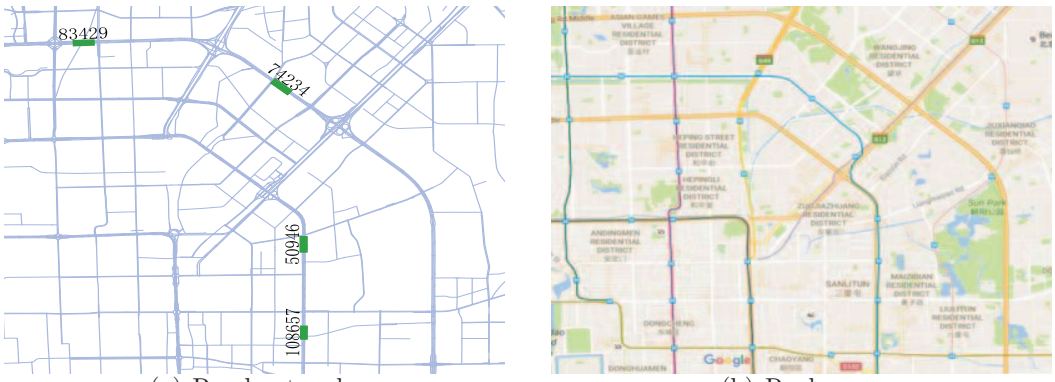
residential district). The congestion at road segment 77971 will then results in the congestion at road segment 70407 (a trunk road at the traffic hub of Northeastern Beijing).



(a) Road network (b) Real map

Figure 3.8: Case study for the traffic congestion transmission in the morning

Similarly, Figure 3.9 shows the traffic congestion transmission among road segments 83429, 74234, 50946 and 108657 in the evening. All these road segments are trunk roads and the congestion transmits sequentially from road segment 83429 to road segment 108657. Particulary, road segments 83429 and 74234 are next to the Asian games village residential district and Wang Jing residential district, respectively, while both road segments 50946 and 108657 are within the Sanlitun commercial area.



(a) Road network (b) Real map

Figure 3.9: Case study for the traffic congestion transmission in the evening

Model Performance and Feature Selection

We evaluate the trained models using average precision and recall. The 10-fold cross validation results are shown in Table 3.4, where the number in the bracket is the standard deviation. Generally, the results are stable with satisfactory precision and recall, considering that we have not conducted a very fine parameter tuning for the best performance.

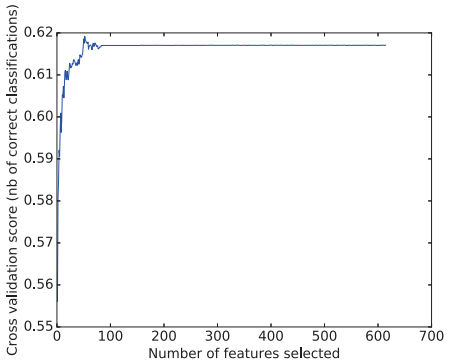
Table 3.4: 10-fold CV results on different classifiers of two models

	Morning Peak		**Evening Peak**	
Metrics / **Classifiers**	Precision	Recall	Precision	Recall
Decision Tree	0.615(0.012)	**0.598(0.033)**	0.661(0.014)	0.642(0.053)
Random Forest	**0.693(0.020)**	0.550(0.029)	**0.742(0.013)**	0.627(0.031)
Logistic Regression	0.626(0.017)	0.559(0.048)	0.682(0.012)	**0.665(0.030)**
SVM	0.639(0.027)	0.446(0.055)	0.692(0.011)	0.633(0.032)

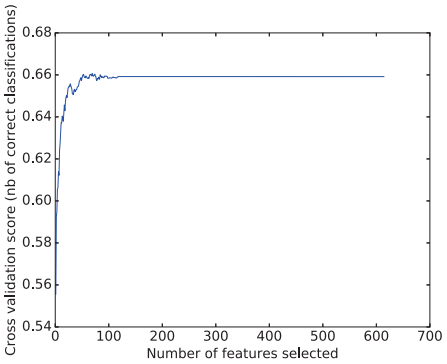
In terms of the two peak models, the evening peak models achieve better performance in both precision and recall than the morning peak models. In terms of precision, models trained on Random Forest achieve the best performance in both morning and evening peaks. This is probably because Random Forest achieves better tradeoff performance between the bias and variance, as it is the only ensemble model among the four classifiers. In terms of recall, models trained on Decision Tree and Logistic Regression achieve the best performance, respectively in morning peak and evening peak.

Now we further perform Recursive Feature Elimination on the features. The basic idea is to iterate over all combinations of features on a designated model to find the optimal number of features with best performance. Here, we performed 3-fold cross validation to measure the overall precision of Decision Tree and Random Forest with different number of features selected. As shown in Figure 3.10, with the increasing number of selected feature, the overall precisions of Decision Tree keep increasing,

though oscillating from time to time, until reaching the optimum, and then go down a little bit and become stable; while in Figure 3.11, the overall precisions of Random Forest, though show similar trends, keep increasing and oscillating a little slower, and reach the optimal on a larger number of features with higher value. The optimal number of features for Decision Tree is much smaller than that of Random Forest, and the precision is also lower. This may be because that Random Forest is capable of extracting more useful information from large feature sets for prediction, and making better prediction.

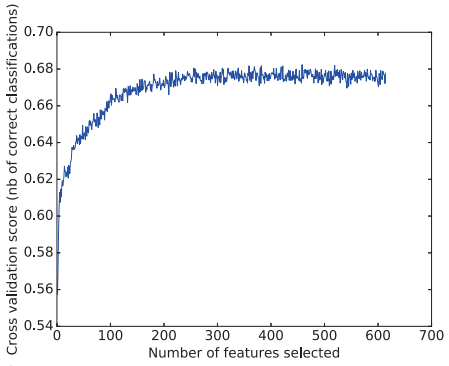


(a) Feature selection of Decision Tree in the morning peak

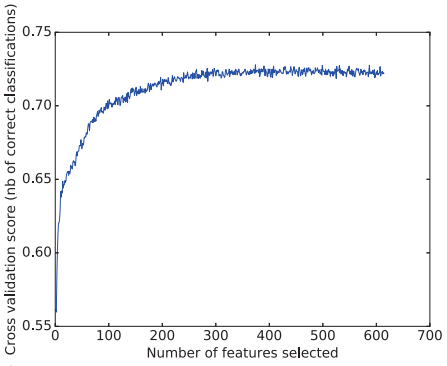


(b) Feature selection of Decision Tree in the evening peak

Figure 3.10: Feature selection of Decision Tree



(a) Feature selection of Random Forest in the morning peak



(b) Feature selection of Random Forest in the evening peak

Figure 3.11: Feature selection of Random Forest

We also divide the feature sets into two categories. One includes features ex-

tracted from road network, the other includes features extracted from POI. We are trying to see how features from these two different categories contribute to the final prediction. As shown in Table 3.5, classifiers using Road network features achieve higher precision than using POI features, and again Random Forest achieve best precision overall. Besides, for some classifiers e.g., Decision Tree, solely using road network features achieve higher precision than using both two categories of features, while for others e.g., Random Forest, using both achieve best precision. The reason may be some classifiers are not able to extract useful features from a large number of features, when introducing more features, more noise are also introduced, resulting to a lower precision. While for others, they are able to extract useful information for prediction from a large number of features even with more noise, resulting to a higher precision. This is somehow in line with our analysis on optimal number of features previously.

Table 3.5: 10-fold CV results on different classifiers of two models based on two categories of features

	Morning Peak		**Evening Peak**	
Features / **Classifiers**	RoadNetwork	POI	RoadNetwork	POI
Decision Tree	0.622(0.019)	0.586(0.010)	0.676(0.016)	0.591(0.010)
Random Forest	0.646(0.018)	0.613(0.017)	0.700(0.012)	0.623(0.016)
Logistic Regression	0.631(0.023)	0.556(0.020)	0.680(0.012)	0.550(0.011)
SVM	0.635(0.022)	0.592(0.016)	0.687(0.013)	0.602(0.015)

Importance Features and Generated Rules

We also compare the top 10 important features identified by two models on different classifiers, and list the commonly identified important features on two models in Table 3.6. In addition, Table 3.7 shows some rules generated by Decision Tree on the two models (note that all the features have been scaled into the range of [0, 1] as described in Section 3.4).

Table 3.6: Commonly identified important features

	Features	**Description**
Morning Peak	Diff-B	the difference of betweenness
	Diff-C	the difference of closeness
	Diff-POI	the difference of the total number of POIs
	SimPOIs	Pearson similarity of POI TF-IDF value distributions
	$netDis$	the network distance
	'trunk→trunk'	binary indicator variable for the ordered combination 'trunk→trunk'of road types
	'motorway→motorway'	binary indicator variable for the ordered combination 'motorway→motorway'of road types
	'catering→catering'	variable for the ordered combination 'catering→catering'of POI categories
Evening Peak	Diff-B	the difference of betweenness
	Diff-C	the difference of closeness
	Diff-POI	the difference of the total number of POIs
	$netDis$	the network distance
	'trunk→trunk'	binary indicator variable for the ordered combination 'trunk→trunk' of road types
	'trunk→secondary'	binary indicator variable for the ordered combination 'trunk→secondary' of road types
	'tertiary→secondary'	binary indicator variable for the ordered combination 'tertiary→secondary' of road types

Table 3.7: Generated rules

	Rules
Morning Peak	If 0.4184 < Diff-POI $\leqslant$ 0.4454 and Diff-B > 0.4755 and Diff-C $\leqslant$ 0.4906, then uncorrelated
	If Diff-POI > 0.4947 and $netDis \leqslant$ 0.294 and 'motorway→motorway' = 1, then correlated
Evening Peak	If Diff-POI $\leqslant$ 0.49 and Diff-B > 0.4938 and 'tertiary→secondary' = 1, then uncorrelated
	If 0.0038 < $netDis \leqslant$ 0.0877 and 'trunk→trunk' = 1, then correlated

As we can see, Diff-B, Diff-C, Diff-POI, $netDis$, and 'trunk→trunk' are both commonly identified important features in the two models, meaning that they are important to predict whether a road segment is correlated with another one in terms of congestion in both morning and evening peaks. On the other hand, 'motorway→motorway' and 'catering→catering' are more important in the morning peak, and 'trunk→secondary' and 'tertiary→secondary' are more important in the evening peak. The results reveal the common and different patterns between morning and evening peaks.

From the generated rules, we can observe more different patterns in the morning and evening peaks. For example, in the morning peak, there exits high congestion correlation from one motorway to another if the POI numbers of them are quite different, meaning that congestions are more likely to propagate from a motorway with more POIs to another one with less POIs in the morning peak. On the other hand, there exits high congestion correlation from one trunk road to another in the evening peak, meaning that congestions are more likely to propagate between trunk roads in the evening peak.

3.6 Conclusion

In this work, we outline a three-phase framework to explore the congestion correlation between road segments from multiple data sources. We first obtain congestion information on road segments from GPS data, give the definition of congestion correlation and design the mining algorithm. Then we extract topological and POI features on each road segment, and fuse them to generate the features of training samples for each pair of road segments. Finally, the congestion correlation and features on each pair of road segments are input to well-known classifiers including Decision Tree, Random Forest, Logistic Regression and Support Vector Machine. We train two models on different classifiers to predict congestion correlation, com-

pare and analyze the performance and important features. The experiment results show stable and satisfactory performance as well as some important patterns of congestion correlation. In addition, we discuss the patterns of congestion correlations, and found obvious directionality and transmissibility. Meanwhile, we also analyze the impact of feature selection on the performance of models. Notably, the proposed framework is general and can be applied to other pairwise correlation analysis.

Chapter 4

Coupled Sparse Matrix Factorization for Response Time Prediction in Logistics Services

In this chapter, we propose a similarity-based method for cross domain data analytics to enable intelligent decision making. The reasons are that it generally has good application performance compared to feature-based method, and it has a weak assumption on data and a medium computation cost compared to other model-based method. As an example application, we study response time prediction in an emerging logistics service, as it is a brand new application and important to both users and service providers.

4.1 Introduction

There is an emerging and convenient way to transport goods in logistics services: Users can make goods delivery orders via a mobile application as illustrated in Figure 4.1, and registered van drivers would respond to take these orders in less than couple of minutes, which is much faster than that of traditional way through van calling centers. The platform behind the service connects large numbers of registered van drivers with on-demand logistics needs from users. Response time, i.e., the time

it takes for a newly created order to be responded by drivers, is an important key performance indicator of such a logistics service. Long response time would lead to low user satisfaction and low system efficiency, which would eventually impair the benefits of the service provider. If the order response time can be predicted accurately, this prediction can be used to reduce the long response time, e.g., service providers can add extra bonus on those orders with long response time to attract drivers. Furthermore, the prediction can also be utilized for supply-demand analysis and driver scheduling, so that both the user experience and system efficiency would be improved. Thus, predicting response time in such logistics service is important and beneficial.

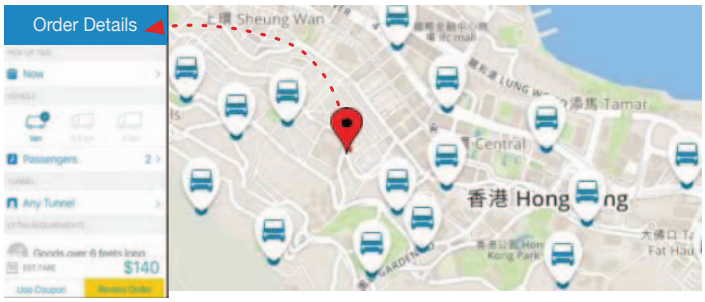


Figure 4.1: Emerging way of transporting goods in logistics services

However, order response time prediction is a challenging task due to the following reasons. First, the heterogeneous fusion challenge: as different driver would have different response times towards different orders in different situations, such a task is affected by many factors including order characteristics, driver preference, driver routine behavior, time of day, *etc.* In addition, the information of these factors usually can only be extracted from heterogeneous data sources. To effectively consider all these factors simultaneously is quite difficult. Simply extracting a combined feature vector from all these data sources, and feeding the feature vector into traditional regression models can hardly capture the complex relation between these factors, which would lead to poor performance. Second, the sparsity challenge: interaction

between orders and drivers in these data sources are quite sparse, as one order can only be responded by one driver. Thus how to enrich these interactions and extract useful information for better response time prediction is a challenging task.

To address the aforementioned issues, we predict the response time using two domains of data: order histories, which describe characteristics of orders, and driver historical GPS locations, which indicate driver routine behaviors and preferences. Specifically, we formulate the response time prediction problem as a matrix factorization problem, and propose a coupled sparse matrix factorization (CSMF) model to fuse these heterogeneous and sparse data from different domains to capture both order and driver information for accurate prediction.

We summarize the contributions as follows:

- We propose a novel framework to fuse heterogeneous and sparse data for response time prediction. In addition, the framework is general, and can be applied to other response time prediction problems.

- We propose a coupled sparse matrix factorization (CSMF) model to deal with the heterogeneous and sparse data. Specifically, coupled matrix factorization with Laplacian and $\ell_{2,1}$ regularizations are utilized for heterogeneous fusion, and a weight setting mechanism is utilized to handle sparsity. In the end, a corresponding efficient optimization algorithm is also devised.

- We compare the proposed CSMF method with many baseline methods, and demonstrate its superiority. We also show the performances of many variants of our method to show the effectiveness of each component.

4.2 Related Work

Response Time Prediction: In terms of response time prediction, two groups of work are most related. One considers response time in Question and Answers (Q&A) communities, while the other considers response time in web services.

For research on Q&A communities, response time is considered as the time it takes for a newly post question to be answered. Predictions on response time in different Q&A communities have been made. Mahmud et al. [54] proposed several statistical models based on Poisson process to predict the response time for questions asked on Twitter. Rechavi et al. [68] analyzed Q&A data from Yahoo! Answers, and conclude that askers tend to choose the best answer based on response time, while the communities pay more attention to the quality of answers. Avrahami et al. [8] generated various features from different settings in instant messaging, and applied decision tree classifiers to predict and analyze whether a message will receive a response within a certain period. Arunapuram et al. [6] made a preliminary effort on analysis of distribution, correlated features as well as prediction of response time on StackOverflow. Burlutskiy et al. [15] proposed a framework which can vary features and machine learning algorithms to predict response time, and validated its effectiveness on datasets from Stack Exchange. Bhat et al. [10] analyzed the factors related to response time in StackOverflow, and found that tag-related factors have a major on response time. They further demonstrated the findings by applying machine learning algorithm on these factors in response time prediction tasks. Goderie et al. [34] predicted response time StackOverflow based on similarity of tags, and achieved an acceptable performance.

For research on response time in web services, response time is considered as the time it takes for a use request to receive a response from web services. Zheng et al. [107] analyzed the historical data of web services, and identified four modes of

response time. Then, they applied different statistical method for different mode to predict the response time of web services. Liang et al. [49] leveraged the content and structure of the social discussion on public discussion boards to predict the response time of web services. Cheung et al. [20] proposed a queueing model based framework along with regression analysis techniques for response time prediction of web services. The response time data of a university website is collected in [103] for 84 days. Then, support vector machine with information granulation is applied for prediction. Atluri et al. [7] decomposed response time as a summation of three constituent times, and predicted them individually using hidden Markov model and Bayesian networks. Zheng et al. [114] proposed a neighborhood-integrated matrix factorization approach for response time prediction, and validated it in large-scale experiments with 5,825 real-world web services.

In both the above related work and ours, response time is crucial to user satisfaction, and accurate prediction would benefit both users and system. Differently, the response time prediction problem in our work contains either richer information i.e., spatial and temporal relation, or more complicated settings, e.g., the response time of a new order, unlike the response time of web services, does not have historical data from exactly the same order. Furthermore, more complicated multiple driver-order dependencies are involved. Thus, the response time prediction problem here is more challenging than previous studies.

Matrix Factorization: Matrix factorization has been widely used in many applications including computer vision [23], natural language processing [11], traffic congestion analysis [88] and especially successful in recommendation systems [47]. It can not only allow for a flexible and generic integration of data, but also discover the hidden patterns among rows and columns of matrices through the factorized latent factors. As large amount of data from multiple sources are available nowadays, each source of data can be represented as a matrix, and jointly analyzing these correlated

data can usually enhance knowledge discovery. Different kinds of coupled matrix factorization methods have been proposed for different applications. Tang et al. [80] proposed a coupled matrix factorization method with graph Laplacian regularization for action recognition. Sung et al. [43] jointly factorize user-item matrix with user-tag matrix to improve the recommendation results. Acar et al. [2] identify potential biomarkers using coupled matrix factorization with ℓ_1 penalties on factors. To deal with missing values, they set the weights for known values as 1, and missing values as 0. Zheng et al. [105] utilize location data, activity data as well as the correlation among them with the proposed coupled matrix factorization method for location and activity recommendations. Shang et al. [73] estimate travel speed of vehicles by jointly factorizing three matrices extracted from map data and historical trajectories.

In both the above related work and ours, matrices from different sources of data are jointly factorized, so that the correlations among them can be discovered and utilized for prediction. Differently, the matrix is much more sparse in our problem, as one order can only be responded by one driver. Through filling the missing entries with their physical meaning along with the corresponding weight setting mechanism, we make the proposed method effective for order response time prediction. Furthermore, we also impose $\ell_{2,1}$ pelnalties on factorized matrices to enable automatic group feature selection within the model for accurate prediction.

4.3 Framework Overview

Figure 4.2 presents the framework of this work. In this framework, we utilize order history and driver historical GPS locations for order response time prediction. We first collect these two sources of datasets. Then, we build an order-driver response time matrix, where each entry stores the response time of a driver to an order. Note that, this matrix is very sparse, i.e., each row only has one non-zero value, since

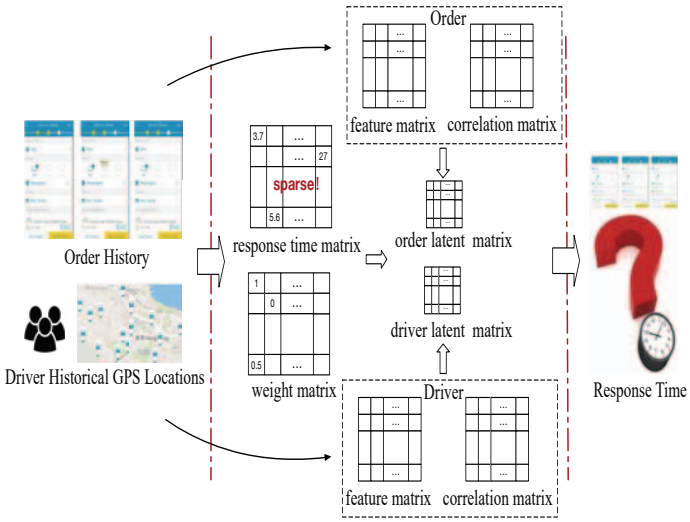


Figure 4.2: CSMF framework

one order can only be responded by one driver. We fill some entries with physical meanings, and use a weight matrix to indicate the confidence of the filled values. Meanwhile, we extract a feature matrix and a correlation matrix from each of these two sources. Lastly, we fuse all the information with the response time matrix to alleviate its sparsity for accurate prediction.

4.4 Data Sources and Feature Extraction

4.4.1 Order History

Each order o_i is associated with an order *id* and driver *id*. In addition, for each o_i, we extract a large set of features which are relevant to the corresponding response time. Concretely, these extracted features are detailed below.

- **Time relevant features**: The day type (i.e., day of week), the hour in which the order is created.

- **User relevant features**: The rating of the user who made the order, and whether he/she has social welfare.

- **Price relevant features**: The price of the order and the tips provided by the user.
- **Personalized requirements**: the tunnel/bridge/route required to pass, required language, whether the user has pets, whether it needs cart, whether the cargo is over 6 feet, whether it needs to help carry, rent hours, cart count, the number of passengers, and whether the order will be paid by the receiver.
- **Location relevant features**: The start location, the number of total orders at the start location historically, end location, the number of order at the end location historically, and the intermediate location and its name, if any.

We then use one-hot encoding to transform all categorical features to numerical values. Also, each order is associated with a real response time.

4.4.2 Driver Historical Location

In addition, we also have the corresponding GPS trajectory data for all drivers corresponding to the orders above. For each driver dr_i, his GPS trajectory tr_i is a sequence of GPS points, i.e.,

$$tr_i = (p_1, p_2, \ldots, p_n)$$

where each GPS point $p_j = (loc, t, status)$ has its location $p_j.loc$, time stamp $p_j.t$, and status flag $p_j.status$ (i.e., if being busy with orders, $status = 1$, otherwise $status = 0$).

In this work, we compute four kinds of features for each driver based on his/her trajectories, i.e., the average and variance of his travel distances in the same hour of all days, and average and variance of the idle ratio in the same hour of all days.

A. Computing average and variance of travel distance

Assuming that the trajectory of driver dr_i during hour j on the k-th day is $tr_{i,j}^k =$

$(p_1, p_2, \cdots, p_m)$, the corresponding travel distance $d^k_{i,j}$ is computed by

$$d^k_{i,j} = \sum_{e=1}^{m-1} dist(p_e, p_{e+1})$$

where $dist(p_e, p_{e+1})$ is the network distance between points p_e and p_{e+1}.

After computing the travel distance $d^k_{i,j}$ for all days in the history, we get a set of travel distance values $D_{i,j}$ for each hour j of day. Then, the average $\bar{d}_{i,j}$ and variance $\sigma^d_{i,j}$ of $D_{i,j}$ are computed as below.

$$\bar{d}_{i,j} = \frac{1}{|D_{i,j}|} \sum_{d \in D_{i,j}} d$$

and

$$\sigma^d_{i,j} = \frac{1}{|D_{i,j}|} \sum_{d \in D_{i,j}} (d - \bar{d}_{i,j})^2$$

B. Computing average and variance of idle ratio

Given the trajectory $tr^k_{i,j} = (p_1, p_2, \cdots, p_m)$ of driver dr_i in hour j on the k-th day, the corresponding idle ratio $c^k_{i,j}$ is

$$c^k_{i,j} = \frac{1}{p_m.t - p_1.t} \cdot \sum_{(p_e, p_{e+1}) \in tr^k_{i,j} \wedge p_e.status = p_{e+1}.status = 0} p_{e+1}.t - p_e.t$$

Similar to computing the travel distance, we then have a set of idle ratio values $C_{i,j}$ for each hour j of day. The average $\bar{c}_{i,j}$ and variance $\sigma^c_{i,j}$ of $C_{i,j}$ are computed as below.

$$\bar{c}_{i,j} = \frac{1}{|C_{i,j}|} \sum_{c \in C_{i,j}} c$$

and

$$\sigma^c_{i,j} = \frac{1}{|C_{i,j}|} \sum_{c \in C_{i,j}} (c - \bar{c}_{i,j})^2$$

4.5 Problem Formulation

Notation. To facilitate the distinction, scalars are denoted by lowercase letters $(x, y, \cdots)$, vectors by boldfaced lowercase letters $(\mathbf{x}, \mathbf{y}, \cdots)$, and matrices by boldface uppercase letters $(\mathbf{X}, \mathbf{Y}, \cdots)$. All vectors are row vectors unless otherwise specified. An element of a vector $\mathbf{x}$ and a matrix $\mathbf{X}$ is denoted by x_i and $x_{i,j}$, respectively.

Suppose that the problem includes N orders and M drivers, where each of them is respectively represented by a feature vector denoted by $\mathbf{x}_i \in \mathbb{R}^S$ and $\mathbf{y}_j \in \mathbb{R}^T$, $i = 1, 2, \cdots, N$, $j = 1, 2, \cdots, M$, where S and T are the dimension of order feature vector and that of driver feature vector, respectively. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N] \in \mathbb{R}^{N \times S}$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_M] \in \mathbb{R}^{M \times T}$ denote the feature matrix of orders and drivers, respectively. We also assume that each order can only be responded by one driver with the minimal time, so all such responses can be arranged into a response time matrix $\mathbf{R} \in \mathbb{R}^{N \times M}$, whose entry $r_{i,j}$ is the value of response time between order i and driver j. If order i is not responded by driver j, then the corresponding entry $r_{i,j}$ in $\mathbf{R}$ is missing. Note that, for each observed order, only the one entry with real response time is observed; for each unobserved order, all entries are missing. In particular, assuming the problem is associated with N_1 observed orders $\mathcal{L} = \left\{(\mathbf{x}_i,\ r_{i,k})\right\}_{i=1}^{N_1}$ and N_2 unobserved orders $\mathcal{U} = \left\{(\mathbf{x}_j)\right\}_{j=1}^{N_2}$, where $N_1 + N_2 = N$, $r_{i,k}$ is the response time of order i, and k indicates which driver has responded to order i. The goal of our response time prediction is to estimate the response time of unobserved orders $\mathbf{x}_j$. Specifically, we aim at leveraging the observed order information and driver information in $\mathbf{X}, \mathbf{Y}$ and $\mathbf{R}$ to help accurately predict the response time of unobserved orders.

4.6 The CSMF Model

4.6.1 Weighted Matrix Factorization

The matrix factorization [78] has been widely used in many applications including computer vision, recommendation systems and natural language processing. In our problem, the basic idea is to factorize the response time matrix $\mathbf{R} \in \mathbb{R}^{N \times M}$ into two low-rank latent matrices $\mathbf{U} \in \mathbb{R}^{N \times L}$ and $\mathbf{V} \in \mathbb{R}^{M \times L}$, representing order and driver distributions on latent semantics, respectively. Then, the unobserved entry in the response time matrix can be predicted through these two specific matrices. However, in our problem, the response time matrix $\mathbf{R}$ is very sparse, only one observed entry in each row, since one order can only be responded by one driver. We fill some of the missing entries to facilitate accurate factorization and use an additional weight matrix to indicate the confidence we have on each filled entry. This approach minimizes the following objective function:

$$\min_{\mathbf{U},\mathbf{V}} \mathcal{L} = \frac{1}{2} \|\mathbf{W} \odot (\mathbf{R} - \mathbf{U}\mathbf{V}^{\mathbf{T}})\|_F^2$$

where $\odot$ represents the Hadamard product (i.e., elementwise product), and $\mathbf{W} = [w_{i,j}]$ is a corresponding weight matrix.

We fill a small portion of missing entries in matrix $\mathbf{R}$ with physical meaning as follows, where $r_{i,j}$ is the missing value in i-th row of $\mathbf{R}$ and $r_{i,k}$ is the real response time in i-th row of $\mathbf{R}$. Since in reality if the entry is missing, then the corresponding driver would probably have a longer response time.

$$r_{i,j} = r_{i,k} + 1, \ if \quad j \neq k \tag{4.1}$$

After filling a small portion of missing entries, we need to set the weight matrix for each entry, which is vital for accurate factorization. Notice that we assume order i is responded by driver k, we set the weight matrix as follows, where corr_{dr_j,dr_k} is the

correlation between driver j and driver k. We will discuss about how to compute the correlation in Sec 4.8.

$$w_{i,j} = \begin{cases} 1 & \text{if } j = k; \\ \text{corr}_{dr_j, dr_k} & \text{if } r_{i,j} \text{ is filled and } j \neq k; \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

The intuition behind this setting is that if the entry $r_{i,j}$ represents real response time, then the weight is as high as 1. If the entry represents the entry we filled previously, then the weight is set to the correlation between driver j whose response time is filled and the driver k who actually respond to the order, with the assumption that the more two drivers are correlated, the more likely they would both consider to respond to same orders, and that missing of entry is simply because the driver j has a longer response time than driver k, so that $r_{i,j}$ is filled in the way close to the real situation. If the entry is missing and not filled, then the weight is as low as 0, which would not affect the model learning process.

4.6.2 Coupled Matrix Factorization

Since the response time matrix is originally very sparse, in order to facilitate a more accurate factorization, we incorporate more information from order and driver data. Specifically, we use order feature matrix $\mathbf{X}$ to help determine the order latent matrix $\mathbf{U}$. This is achieved by factorizing order feature matrix $\mathbf{X} \in \mathbb{R}^{N \times S}$ into low-rank latent matrices $\mathbf{U} \in \mathbb{R}^{N \times L}$ and $\mathbf{G} \in \mathbb{R}^{L \times S}$. Similarly, we use driver feature matrix $\mathbf{Y}$ to help determine the driver latent matrix $\mathbf{V}$, which is achieved by factorizing driver feature matrix $\mathbf{Y} \in \mathbb{R}^{M \times T}$ into $\mathbf{V} \in \mathbb{R}^{M \times L}$ and $\mathbf{H} \in \mathbb{R}^{L \times T}$. That is to say, we collaboratively factorize $\mathbf{R}$ to $\mathbf{U}$ and $\mathbf{V}^{\mathbf{T}}$, with order feature matrix $\mathbf{X}$ to low-rank latent matrices $\mathbf{U}$ and $\mathbf{G}$, and driver feature matrix $\mathbf{Y}$ to $\mathbf{V}$ and $\mathbf{H}$, where they share the latent matrices $\mathbf{U}$ and $\mathbf{V}$. In this way, order feature matrix $\mathbf{X}$ and driver feature matrix $\mathbf{Y}$ would contribute to a more accurate factorization of $\mathbf{R}$ through the

shared matrices $\mathbf{U}$ and $\mathbf{V}$. The objective function can be formed as follows, where λ_1 and λ_2 are the parameters to control the contribution of each component during the copuled factorization.

$$\min_{\mathbf{U},\mathbf{V},\mathbf{G},\mathbf{H}} \mathcal{L} = \frac{1}{2}\|\mathbf{W} \odot (\mathbf{R} - \mathbf{U}\mathbf{V}^{\mathbf{T}})\|_F^2 + \frac{\lambda_1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{G}\|_F^2 + \frac{\lambda_2}{2}\|\mathbf{Y} - \mathbf{V}\mathbf{H}\|_F^2$$

4.6.3 Correlations and Group Feature Selection

To further alleviate the data sparsity problem, we also utilize correlations between orders as well as drivers to facilitate the process of coupled factorization, by incorporating information in Laplacian matrices of their correlation graphs using $\mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{L}_u\mathbf{U})$ and $\mathrm{tr}(\mathbf{V}^{\mathrm{T}}\mathbf{L}_v\mathbf{V})$, respectively. For order correlation, $\mathbf{L}_u = (\mathbf{D} - \mathbf{Z}) \in \mathbb{R}^{N\times N}$ is the Laplacian matrix of the order correlation graph, in which $\mathbf{Z}$ is a correlation matrix of orders, with each entry $z_{i,j}$ represent the correlation between order o_i and o_j, and $\mathbf{D}$ is a diagonal matrix with diagonal entries $d_{i,i} = \sum_i z_{i,j}$. Then, $\mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{L}_u\mathbf{U})$ is obtained through the following deduction, which guarantees two orders o_i and o_j with a higher correlation (i.e., $z_{i,j}$ is big) should also have a closer distance between the vector $\mathbf{u}_i$ and $\mathbf{u}_j$ in the matrix $\mathbf{U}$.

$$\begin{aligned}
&\frac{1}{2}\sum_{i,j} z_{i,j}\|\mathbf{u}_i - \mathbf{u}_j\|_2^2 = \sum_{i,j} \mathbf{u}_i z_{i,j}\mathbf{u}_i^{\mathrm{T}} - \sum_{i,j} \mathbf{u}_i z_{i,j}\mathbf{u}_j^{\mathrm{T}} \\
&= \sum_i \mathbf{u}_i d_{i,i}\mathbf{u}_i^{\mathrm{T}} - \sum_{i,j} \mathbf{u}_i z_{i,j}\mathbf{u}_j^{\mathrm{T}} \\
&= \mathrm{tr}(\mathbf{U}^T(\mathbf{D} - \mathbf{Z})\mathbf{U}) \\
&= \mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{L}_u\mathbf{U})
\end{aligned}$$

where $\mathrm{tr}(\cdot)$ denotes the matrix trace. Similarly, we define $\mathbf{L}_v = (\tilde{\mathbf{D}} - \tilde{\mathbf{Z}}) \in \mathbb{R}^{M\times M}$ as the Laplacian matrix of the driver correlation graph, and consider the

driver correlation as $\text{tr}(\mathbf{V}^{\text{T}}\mathbf{L}_v\mathbf{V})$. In this way, we utilize order correlations and driver correlations for accurate factorization of $\mathbf{R}$ to $\mathbf{U}$ and $\mathbf{V}$.

Since the collected features contain various kinds of information, we assume only a small part of features are predictive. To this end, we add an $\ell_{2,1}$-norm regularization on each latent matrix. The $\ell_{2,1}$-norm promotes row-wise sparsity of the target matrix, such property makes it suitable for the task of feature selection. In particular, as ℓ_2 norm is combined through an ℓ_1 norm, it encourages all feature matrices to select a common set of features and thus play the role of group feature selection [98].

4.6.4 Coupled Sparse Matrix Factorization

All components mentioned in previous subsections are combined, and the final model is illustrated in Figure 4.3. We first fill some of the missing entries in response time matrix $\mathbf{R}$, and set the corresponding weight matrix $\mathbf{W}$. Then, we factorize $\mathbf{R}$ with order feature matrix $\mathbf{X}$ and driver feature matrix $\mathbf{Y}$, so that the factorized share latent matrices $\mathbf{U}$ and $\mathbf{V}$ are more accurate. Furthermore, we utilize order correlation matrix $\mathbf{Z}$, so that if two orders i, j are correlated, they would have a closer distance between the vector $\mathbf{u}_i$ and $\mathbf{u}_j$ in the matrix $\mathbf{U}$. We utilize driver correlation matrix $\tilde{\mathbf{Z}}$ in the same way to make $\mathbf{V}$ more accurate. Finally, the factorized matrices $\mathbf{U}$ and $\mathbf{V}$ are used to reconstruct $\mathbf{R}$ to $\tilde{\mathbf{R}}$, namely $\tilde{\mathbf{R}} = \mathbf{U}\mathbf{V}^{\text{T}}$. The reconstructed $\tilde{\mathbf{R}}$ is used to predict the response time of unobserved orders in the following way. Each unobserved order in $\tilde{\mathbf{R}}$ would have values, which are missing in the original response time matrix $\mathbf{R}$. The minimal value of each unobserved order is consider as the predicted response time of that order.

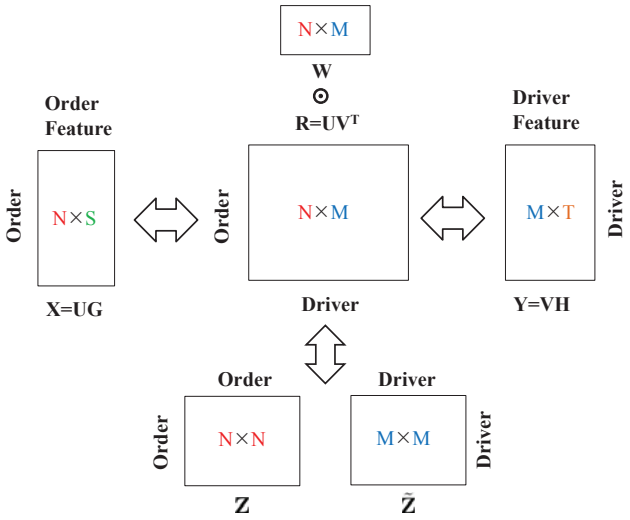


Figure 4.3: Coupled sparse matrix factorization

The objective function of the proposed CSMF model is formally given as follows.

$$\begin{aligned}\min_{\mathbf{U},\mathbf{V},\mathbf{G},\mathbf{H}} \mathcal{L} = \frac{1}{2}\underbrace{\|\mathbf{W}\odot(\mathbf{R}-\mathbf{U}\mathbf{V}^{\mathbf{T}})\|_F^2}_{\text{response time factorization}} + \frac{\lambda_1}{2}\underbrace{\|\mathbf{X}-\mathbf{U}\mathbf{G}\|_F^2}_{\text{order factorization}} \\ + \frac{\lambda_2}{2}\underbrace{\|\mathbf{Y}-\mathbf{V}\mathbf{H}\|_F^2}_{\text{driver factorization}} + \frac{\lambda_3}{2}\underbrace{\text{tr}(\mathbf{U}^{\text{T}}\mathbf{L}_u\mathbf{U})}_{\text{order correlation}} + \frac{\lambda_4}{2}\underbrace{\text{tr}(\mathbf{V}^{\text{T}}\mathbf{L}_v\mathbf{V})}_{\text{driver correlation}} \\ + \frac{\lambda_5}{2}\underbrace{(\|\mathbf{U}\|_{2,1}+\|\mathbf{V}\|_{2,1}+\|\mathbf{G}\|_{2,1}+\|\mathbf{H}\|_{2,1})}_{\text{row-wise sparsity regularization}} \end{aligned} \tag{4.3}$$

where $\|\mathbf{W}\odot(\mathbf{R}-\mathbf{U}\mathbf{V}^{\mathbf{T}})\|_F^2$ is to control the error of factorizing response time matrix $\mathbf{R}$, $\|\mathbf{X}-\mathbf{U}\mathbf{G}\|_F^2$ is to control the error of factorization of order feature matrix $\mathbf{X}$, $\|\mathbf{Y}-\mathbf{V}\mathbf{H}\|_F^2$ is to control the error of factorization of driver feature matrix $\mathbf{Y}$, $\text{tr}(\mathbf{U}^{\text{T}}\mathbf{L}_u\mathbf{U})$ and $\text{tr}(\mathbf{V}^{\text{T}}\mathbf{L}_v\mathbf{V})$ is to utilize information in order and driver correlations, λ_1, λ_2, λ_3 and λ_4 are used to control the contribution of each component during the coupled factorization, and λ_5 is a regularization parameter preventing overfitting and controlling group feature selection.

4.7 Optimization Algorithm

Since the objective function in Eq. (4.3) is non-convex with all variables $\mathbf{U}, \mathbf{V}, \mathbf{G}$, and $\mathbf{H}$ together, we employ an efficient iterative algorithm to solve this problem, by alternatively updating one variable while fixing others until convergence. In addition, it is difficult to directly minimize the compound $\ell_{2,1}$ objective function, because $\ell_{2,1}$-norm is not continuous on the origin. Inspired by the half-quadratic minimization method [61], we transform the $\ell_{2,1}$-norm term into an approximate form by introducing an auxiliary variable. Taking $\mathbf{U}$ as an example, we transform:

$$\|\mathbf{U}\|_{2,1} \approx \mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{Q}_u\mathbf{U}) \tag{4.4}$$

where $\mathbf{Q}_u$ is a diagonal matrix with the i-th diagonal element as $q^u_{i,i} = \frac{1}{2\|\mathbf{u}_i\|_2}$. Similarly, we will introduce auxiliary variables $\mathbf{Q}_v, \mathbf{Q}_g$, and $\mathbf{Q}_h$ for variables $\mathbf{V}, \mathbf{G}, \mathbf{H}$, respectively.

Based on the above transformation, we calculate the derivative of Eq. (4.3) with respect to $\mathbf{U}, \mathbf{V}, \mathbf{G}$ and $\mathbf{H}$, respectively, and set them to zero, we have:

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \mathbf{U}} = \mathbf{W} \odot \mathbf{W} \odot (\mathbf{U}\mathbf{V}^{\mathbf{T}} - \mathbf{R})\mathbf{V} + \lambda_1(\mathbf{U}\mathbf{G} - \mathbf{X})\mathbf{G}^{\mathbf{T}} \\ \qquad\qquad + \lambda_3\mathbf{L}_u\mathbf{U} + \lambda_5\mathbf{Q}_u\mathbf{U} = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \mathbf{V}} = (\mathbf{U}^{\mathbf{T}}(\mathbf{W} \odot \mathbf{W} \odot (\mathbf{U}\mathbf{V}^{\mathbf{T}} - \mathbf{R})))^{\mathbf{T}} \\ \qquad\qquad + \lambda_2(\mathbf{V}\mathbf{H} - \mathbf{Y})\mathbf{H}^{\mathbf{T}} + \lambda_4\mathbf{L}_v\mathbf{V} + \lambda_5\mathbf{Q}_v\mathbf{V} = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \mathbf{G}} = \lambda_1\mathbf{U}^{\mathbf{T}}(\mathbf{U}\mathbf{G} - \mathbf{X}) + \lambda_5\mathbf{Q}_g = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \mathbf{H}} = \lambda_2\mathbf{V}^{\mathbf{T}}(\mathbf{V}\mathbf{H} - \mathbf{Y}) + \lambda_5\mathbf{Q}_h = 0 \end{cases} \tag{4.5}$$

Eq. (4.5) involves solving four linear systems, where we use conjugate gradient (CG) method to solve it, which is an effective and widely used technique [42], and only needs to perform matrix multiplications of Eq. (4.5).

Algorithm 4.1 Algorithm to solve the problem in Eq. (4.3)

Input: Response time matrix $\mathbf{R}$, weight matrix $\mathbf{W}$, order feature matrix $\mathbf{X}$, driver feature matrix $\mathbf{Y}$, Laplacian matrices $\mathbf{L}_u, \mathbf{L}_v$ and regularization parameters
Output: $\mathbf{U}$, $\mathbf{V}$
1: Initialize $\mathbf{U}, \mathbf{V}, \mathbf{G}, \mathbf{H}$ randomly
2: Calculate the diagonal matrices $\mathbf{Q}_u$, $\mathbf{Q}_v$, $\mathbf{Q}_g$, $\mathbf{Q}_h$
3: **repeat**
4: Update $\mathbf{U}$, $\mathbf{V}$, $\mathbf{G}$, $\mathbf{H}$ by solving Eq. (4.5) using CG method
5: Update the diagonal matrices $\mathbf{Q}_u$, $\mathbf{Q}_v$, $\mathbf{Q}_g$, $\mathbf{Q}_h$ by

$$q^u_{i,i} = \frac{1}{2\|\mathbf{u}_i\|_2}, q^v_{i,i} = \frac{1}{2\|\mathbf{v}_i\|_2}, q^g_{i,i} = \frac{1}{2\|\mathbf{g}_i\|_2}, q^h_{i,i} = \frac{1}{2\|\mathbf{h}_i\|_2}$$

6: **until** convergence

The overall algorithm is summarized in Algorithm 4.1.

Convergence Analysis. Although we have to solve Eq. (4.3) in an iterative process, due to non-convexity and non-smoothness, the objective monotonically decreases in each iteration and it has a lower bound (proof can be derived in a similar manner as in [38]). Therefore, it guarantees that we can find the optimal solution of each iteration and finally, Algorithm 4.1 can converge to a local minimum of the objective function in Eq. (4.3).

4.8 Experiments

4.8.1 Datasets

The order data used in this study is from a leading logistics company in Hong Kong, which offers an on-demand goods transport service. The data covers a time span of three months and has around 570,000 orders (the real number is about 900,000 among which we remove those orders whose drivers have no enough location data) in total. In addition, we also have the corresponding GPS trajectory data for all drivers during the period above. The total number of drivers is around 7,000.

Figure 4.4 shows the distribution of orders over different day types, i.e., day of the week, where 0 represents Sunday, 1 represents Monday, and so on. According

to Figure 4.4, Sunday has the smallest number of orders during a week, and except Sunday the distribution of orders does not change much.

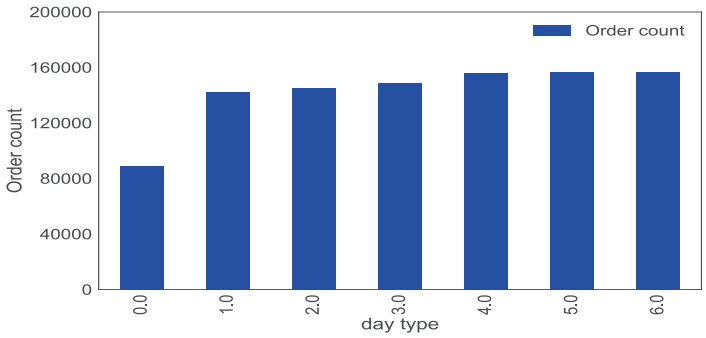


Figure 4.4: The distribution of orders in different day types during all three months.

We use absolute value of Pearson Correlation [77] to calculate order-order and driver-driver correlations, and calculate the corresponding Laplacian Matrix. In order to avoid introducing too much noise, we also only fill some of the missing entries so that 15% of entries in response time matrix $\mathbf{R}$ have values.

Ground Truth and Metric

We predict the order response time of current day based on order history of the previous day, order features of current day and driver historical locations. The ground truth is obtained from current day. All methods are evaluated by the accuracy of their predicted response time averaged over three months. The metric we use in the experiment is Mean Absolute Value (MAE):

$$MAE = \frac{\Sigma_i |y_i - \hat{y}_i|}{N} \tag{4.6}$$

where N is the number of predicted orders, y_i is the real response time of order i in testing data, and $\hat{y}_i$ is calculated by getting the minimal predicted value of order i.

4.8.2 Baselines and Metrics

In order to demonstrate the effectiveness of our CSMF method, we compared the proposed framework with the following existing baselines. For all these baselines methods, as they are all vector-based methods, we join the order feature with the driver feature into a combined feature vector via driver *id*, and use the combined feature vector as input.

- **Ridge**: Ridge regression [37] is a commonly used method for regression problem by imposing a penalty on the size of coefficients of ordinary least squares.

- **Lasso**: The Lasso [81] is another widely used method for regression problem by imposing a penalty on the number of nonzero coefficients of ordinary least squares

- **EN**: The elastic net [115] is a regularized regression method that linearly combines the ℓ_1 and ℓ_2 penalties of the lasso and ridge models, providing a trade-off option between sparsity of coefficients and sizes of coefficients.

- **KNN**: The K Nearest Neighbours method [4] predicts the label, i.e., response time using a weighted sum of its k-nearest neighbours. The neighbour is calculated by Minkowski distance between feature vectors.

- **SVR**: Support Vector Regression [76] is implemented with Gaussian-RBF kernel, which is the most widely used vector-based method.

- **GBR**: Gradient Boosting Regression [29] produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

4.8.3 Model Comparison

Figure 4.5 shows the performance comparison among various methods. The lower the MAE value is, the better the method performs. We can see that ridge, Lasso, EN and KNN perform similarly, among which EN performs slightly better. SVR and GBR perform much better than previous methods, and SVR performs the best among baseline methods, which may be due to it can capture nonlinear relationships in the data. The proposed CSMF method achieves the best MAE value compared to all baseline methods.

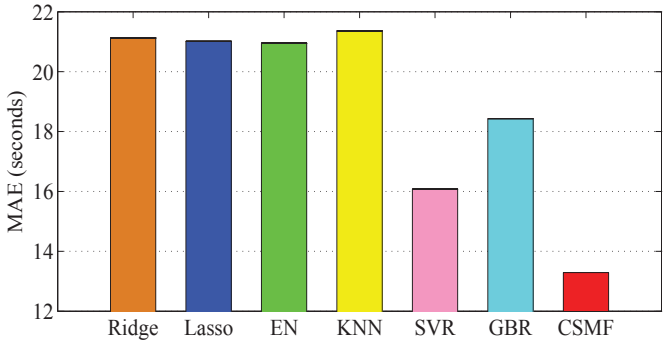


Figure 4.5: Performance comparison among various methods

4.8.4 Evaluation on Model Components

To evaluate each component of the CSMF model, we compared it with its three different variants.

- CSMF-Y: In this variant, the driver feature matrix $\mathbf{Y}$ is not included in the model. We can derive it by setting $\lambda_2 = 0$.

- CSMF-Lp: In this variant, the Laplacian matrices of order and driver correlation graphs are not included in the model. We can derive it by setting λ_3, and $\lambda_4 = 0$.

- CSMF-W: In this variant, we do not fill any of the missing entries in response time matrix **R**. The weights of all missing entries in weight matrix **W** are set to 0, while the weights of entries with real response time remain 1.

Figure 4.6 shows the performance comparison among these variants. Without driver feature, CSMF-Y performs worst, which indicates fusing driver information is quite crucial in prediction. Without Laplacian matrices of order and driver correlation graphs, CSMF-Lp also cannot achieve a good MAE value, which shows utilizing correlations between orders and drivers is important. Without filling missing entries and setting corresponding weights, CSMF-W also performs worse than the original model, which indicates the filling and weight setting effectively contribute to the prediction.

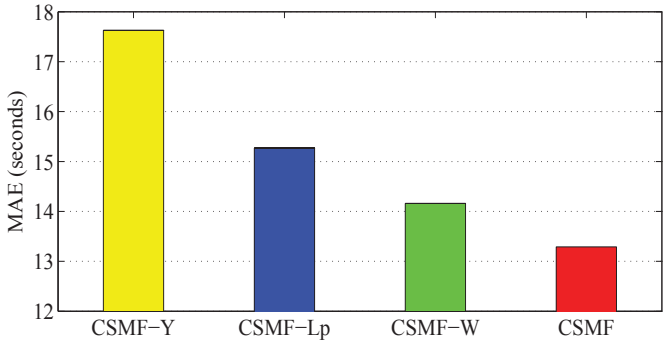


Figure 4.6: Performance comparison among variants

4.8.5 Evaluation on Filling Missing Entries and Weight Settings

To further investigate how filling missing entries and weight setting affect the performance of the proposed CSMF model, we evaluate three parts of the proposed weight settings and filling method.

The first part is to investigate how the percentage of filled entries would affect the performance of the model. We vary the percentage of filled entries in response time

matrix $\mathbf{R}$ from 0% to 30%, and show the results in Figure 4.7. As the percentage of filled entries increases from 0% to 15%, the MAE value decreases, which indicates the filling process introduces useful information for prediction. As the percentage increases from 15% to 30%, the MAE value goes up and down. This is probably because there is a trade-off between the useful information and noise introduced in the filling process.

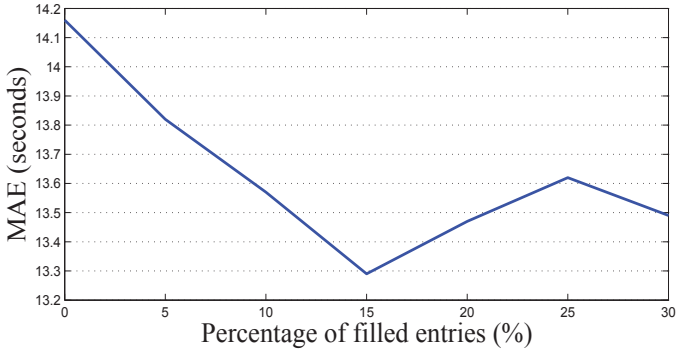


Figure 4.7: Evaluation on percentage of filled missing entries

The second part is to investigate how the weight of filled entries would affect the performance of the model. Previously in Section 4.6.1, we set the weight of missing entries using Pearson correlation. Now we also compare Pearson correlation to another widely used correlation metric: Spearman's rank correlation [77]. While Pearson correlation measures linear relationships, Spearman's rank correlation measures monotonic relationships including both linear and nonlinear relationships. The results are shown in Figure 4.8. The performance is very similar, and CSMF using Pearson correlation is slightly better than CSMF using Spearman's rank correlation, which may be because the correlation in the data is mainly linear.

The third part is to investigate how the value of filled entries would affect the performance of the model. Previously in Section 4.6.1, we set the value of missing entries to real response time plus 1, now we vary the value from real response time plus 1 to 5, and show the results in Figure 4.9. As the value increases, we can see the

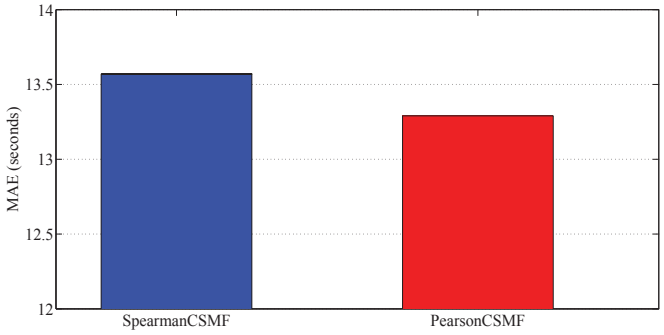


Figure 4.8: Evaluation on weights of filled entries

MAE value also increases, which indicate that a larger value would introduce more noise to the model learning process.

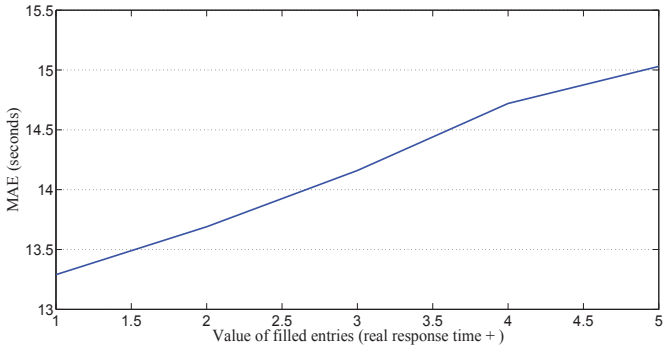


Figure 4.9: Evaluation on values of filled entries

4.8.6 Evaluation on Parameters of CSMF

We also investigate how parameters affect the performance of the CSMF model. As shown in Section 4.6.4, there are 5 parameters in the proposed model, namely λ_1, λ_2, λ_3, λ_4, and λ_5. We range all the parameters from 10^{-2} to 10^1, and the parameters that achieve optimal performance are $\lambda_1 = 1$, $\lambda_2 = 10^{-1}$, $\lambda_3 = 10^{-1}$, $\lambda_4 = 10^{-1}$, and $\lambda_5 = 1$. To analyze how each parameter affects the performance of the model, below we show the performance of varying one parameter while fixing other parameters as the optimal values.

Figure 4.10 shows the performance comparison of λ_1 and λ_2. We can see as the value of λ_1 increases, the MAE value of the model first decreases, achieves the optimal value when $\lambda_1 = 1$, and then increases. Similar trend can be observed on λ_2, while the MAE achieves the optimal value when $\lambda_2 = 10^{-1}$. Recall that λ_1 and λ_2 are respectively to control the contributions of the order feature matrix and the driver feature matrix. Figure 4.10 indicates that when first incorporating the order and driver feature information into the model, the performance of the model gets improved. However, when this information is given too much attention, as the corresponding parameters increase, much noise is also introduced into the model, which impairs the performance of the model. Note that, although the two parameters show similar trends, order feature information with the corresponding parameter λ_1 contributes more and introduces less noise to the model, since the MAE values are smaller compared to that of driver feature information with λ_2.

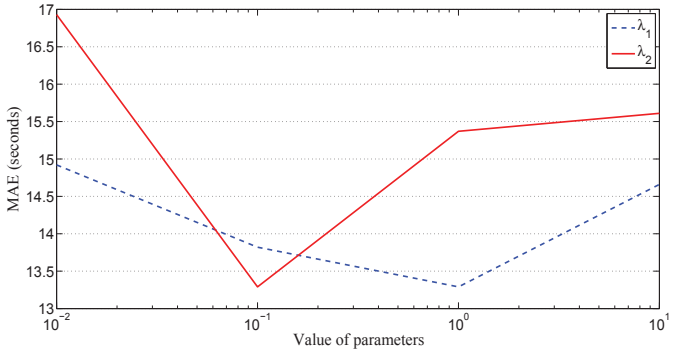


Figure 4.10: Evaluation on λ_1 and λ_2

Figure 4.11 shows the performance comparison of λ_3 and λ_4. We can see as the value of λ_3 increases, the MAE value of the model first decreases, achieves the optimal value when $\lambda_3 = 10^{-1}$, and then increases. Similar trend can be observed on λ_4. Recall that λ_3 and λ_4 are respectively to control the contributions of the order correlations and the driver correlations. Figure 4.11 indicates that when first incorporating the order and driver correlations into the model, the performance of the

model gets improved. However, when this information is given too much attention, as the values of corresponding parameters increase, much noise is also introduced into the model, impairing the performance of the model. Note that, although the two parameters show similar trends, driver correlations with the corresponding parameter λ_4 contribute more and introduce more noise to the model, since the MAE values are smaller at the beginning and larger at the end compared to that of order feature information with λ_3.

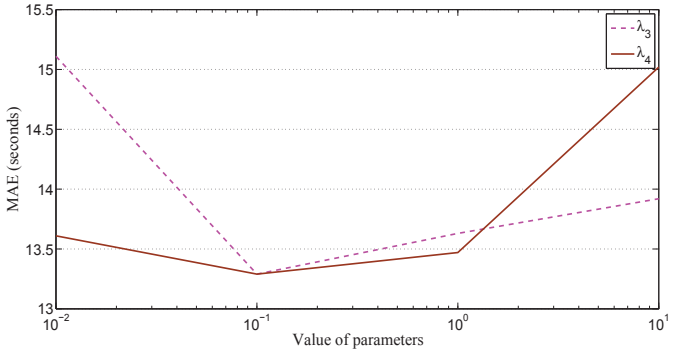


Figure 4.11: Evaluation on λ_3 and λ_4

Figure 4.12 shows the performance comparison of λ_5. We can see as the value of λ_5 increases, the MAE value of the model first decreases, achieves the optimal value when $\lambda_5 = 1$, and then dramatically increases. Recall that λ_5 is to control overfitting and group feature selection. Figure 4.12 indicates that the group feature selection would help improve the performance of the model when the parameter is relatively small (smaller than 1). While the parameter is too large (say larger than 1), the model could be under-fitting or not many useful group features are selected, constraining the model performance from its best.

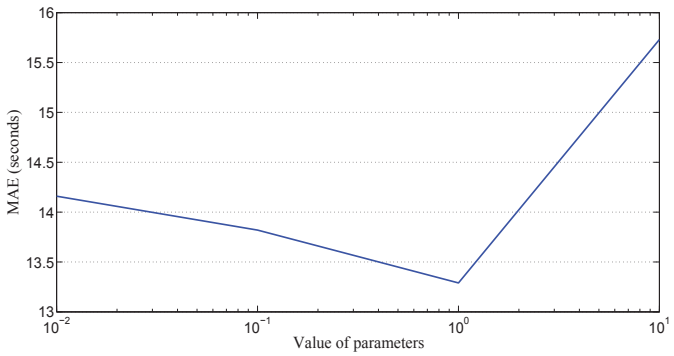


Figure 4.12: Evaluation on λ_5

4.9 Conclusion

In this work, we propose a coupled sparse matrix factorization (CSMF) model to predict order response time on current day by fusing data from order history and driver historical locations. Specifically, we address the heterogeneous fusion and data sparsity challenges raised in this problem using our proposed model, which jointly learns from multiple heterogeneous sparse data through the proposed weight setting mechanism therein. Experiments on real-world datasets show improvement of our approach as compared to various baseline models as well as its different variants. The performances of many variants of the proposed method are also presented to show the effectiveness of each component.

Chapter 5

Coupled Weighted Tensor-matrix Factorization for Order Accepting Probability Prediction in Logistics Services

In this chapter, we extend the previous similarity-based method to higher dimension to consider more factors: not only the similarity among orders and drivers, but also among time. Moreover, we improve the efficiency of the model. As an example application, we study accepting time prediction in logistics, as it is a brand new application and facilitate informed order dispatching, benefiting both users and service providers.

5.1 Introduction

Using a mobile application to make on-demand orders for goods transportation is becoming prevalent in logistics services. In this case, users can specify requirements about the orders in real time, and autonomous van drivers can choose to accept the interesting orders. Figure 5.1 illustrates the distributions of orders and drivers in Kowloon, where orders are dispatched to available drivers through a dispatching center. By connecting on-demand logistics needs with large numbers of autonomous

van drivers, the emerging logistics service outperforms the traditional counterparts (e.g., call centers) with faster response and higher efficiency.

However, most existing dispatching algorithms in on-demand logistics are implemented just based on the spatial distances between drivers and orders, while ignore the personalized preferences of drivers. This not only leads to inferior experience for drivers, but also results into a high order cancel rate due to the large number of mismatched orders and drivers. Therefore, to make more informed dispatching, it is of high importance to estimate the probability of a driver accepting a particular order. This order accepting probability, measuring how likely a driver will take an order, is a key indicator for both of the attractiveness of the order and the preferences of the driver. Precisely predicting the order accepting probability of drivers could guide order dispatching when particular orders have special requirements, offer suggestions on order requirement adjustment when current requirements are hard to fulfill, facilitate supply and demand analysis, etc. Thus, effective prediction on order accepting probability would improve the user satisfaction and service efficiency, and is desired in practice.

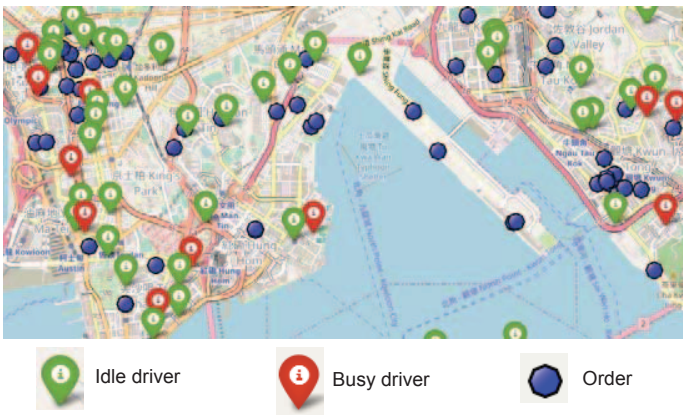


Figure 5.1: The distributions of orders and drivers, where the green balloon markers represent idle drivers, the red balloon makers represent busy drivers, and the circles represent orders.

Nevertheless, effective order accepting probability prediction involves three major challenges that cannot be easily addressed. The first challenge is heterogeneous data fusion. Order accepting probability is affected by many factors such as order characteristics, driver preferences and routine behaviors, and spatio-temporal contexts. The information of these factors is usually extracted from different data sources and heterogeneous in terms of formats and domains. Directly concatenating the features extracted from different sources would lead to undesirable performance [108]. Thus, to effectively consider all these factors is difficult, and requires advanced models. The second challenge is sparsity. As each order is unique and can only be accepted by one driver, the interactions between orders and drivers are very sparse, especially for new drivers with very limited number of accepted orders. The third challenge is efficiency. Computationally expensive models cannot be directly applied, since the prediction has to be made in an efficient manner, so that the predicted order accepting probability can be utilized for order dispatching and suggestions on requirement adjustment in real time.

To address the aforementioned challenges, we propose a three-stage framework with a Coupled Weighted Tensor-matrix Factorization (CWTF) model to predict the order accepting probability accurately and efficiently. Specifically, we first group the orders into clusters based on historical order data to solve the sparsity challenge; then to fuse order characteristics and driver behaviors, we design an efficient tensor-matrix factorization method to model and infer the accepting probability of each driver with respect to each order cluster in each time slot; finally, given a new order at a specific time, we find the most similar cluster, and use the cluster accepting probability of the driver at that time as the prediction, which could be done in real time.

We summarize the contributions of this work as follows:

- To our knowledge, we are the first to study the important problem of order accepting probability prediction in logistics services, and propose a three-stage framework to effectively and efficiently address it.
- We propose the CWTF, which introduces weight setting and slice-based reformulation on the tensor, for effective and efficient heterogeneous fusion. A corresponding optimization algorithm is also devised.
- We compare the proposed method with various baseline methods, and demonstrate its superiority. We also evaluate the performance of several variants to show the effectiveness of model components.

5.2 Related Work

In this section, we will review recent literatures that are closely related to our work from the aspects of *taxi order dispatching*, *logistics order forecasting & planning*, *regression-based prediction* and *coupled tensor-matrix factorization*.

Taxi Order Dispatching. To our knowledge, there are few works studying the problem of order accepting probability prediction for goods in logistics services. Most existing works focus on studying taxi order dispatch modeling [71, 22, 101, 56, 48, 87, 55, 33], which is similar to our task. Seow et al. [71] presented a novel multi-agent approach to automating taxi dispatch in a distributed fashion. Ravina et al. [33] proposed a taxi order supply forecasting methodology which incorporated information from the affecting factors on taxi supply, such as driver incentive schemes, holidays, hour of the day and the day of the week. Neema et al. [22] proposed a multi-level clustering approach for forecasting taxi travel demand in the context of a mobile application-based taxi hailing service. Miao et al. [55] studied the taxi dispatch model with real-time sensing data, and proposed a Receding Horizon Control (RHC) framework to dispatch taxis in metropolitan areas. Fei Miao et al. [56]

further studied the demand uncertainty issue in taxi order dispatch. To address this issue, they developed a data-driven robust taxi dispatch framework to consider the spatial-temporally correlated demand uncertainties. Zhang et al. [101] proposed a combinatorial optimization model to maximize the global success rate in the dispatch system of a taxi-booking app. Tong et al. [82] proposed a unified regression model to accurately predict the Unit Original Taxi Demand (UOTD) for online taxicab platforms. Although many models are proposed for taxi order dispatch modeling, whether a taxi driver will accept the order of a customer is not fully explored, let alone our studied problem of order accepting probability prediction for goods in logistics services. The differences in both the studied problem and the application scenarios make the above mentioned models not applicable to our problem.

Logistics Order Forecasting & Planning. The studied problem is also related to logistics services, and thus we also review the related work in order forecasting & planning in logistics services. Previous researches on order planning mostly focused on travel time prediction of the orders. Lin et al. [50] provided an early survey on this problem. Simroth and Zahle studied the problem of travel time prediction in transportation and logistics [75]. A nonparametric distribution-free regression model is proposed to predict the remaining travel times of long-range trips. Ehmike et al. [26] proposed to utilize the source of Floating Car Data (FCD) to develop and provide the travel times in logistics services. Cattaruzza et al. [16] reviewed the vehicle routing problems for the logistics orders in a city. They gave a comprehensive overview of the literature devoted to vehicle route optimization in cities. Ehmke and Campbell [25] developed and compared strategies that maximize the profits of a logistics service provider by accepting as many delivery requests as possible, while assessing the potential impact of a request on the service quality of a delivery tour. Groß et al. [35] proposed the usage of interval travel times (ITT) to enable cost-efficient and reliable routing for logistics services in urban areas. Laan et al

[83] analyzed the logistics order forecasting and planning process, and found several internal and external factors that can influence the performance of order forecasting and planning. Brent and Mattew [91] studied the the problem of creating order forecasting problem, and they discovered that the order historical data is essential to forecast the future orders placed by retailers. To summarize, most previous work on logistics order planning & forecasting focused on travel route planning, order arriving time prediction, and order number prediction. Predicting the probability of a driver accepting an on-demand order is less touched.

Regression-based Prediction. The focus of regression analysis is on modeling the relationship between a dependent variable and one or more independent variables [28], and has been widely used for prediction and forecasting in various applications [44, 86, 102, 94]. Wang et al. [86] proposed a sparse logistic regression model that can effectively handle the multi-dimensional data, and applied the model to predict the onset risk of patients with Alzheimer's disease and heart failure. Zhang et al. [102] proposed a generalized linear mixed regression model for response prediction of the LinkedIn users. A significant advantage of their model is that it scales to very large datasets. Yan et al. [94] proposed a coupled group lasso regression model CGL to predict the click through rate in display advertising. Compared to traditional linear regression models, CGL can capture the conjunction information from user features and ad features. In our studied problem, directly applying regression-based models may not achieve promising results due to the heterogeneity and sparsity of the data.

Coupled Tensor-Matrix Factorization. As in this paper we utilize the coupled tensor-matrix factorization model, we also briefly review recent advances in this topic. Tensor matrix or coupled tensor-matrix factorization is widely used for data fusion [1, 65] and missing data completion [27, 51, 90]. This technique has attracted rising research interests recently and has been explored in various areas and tasks,

including graph mining [27], urban computing [89, 88, 111], natural language processing [17], and image processing [51]. Beyza et al. [27] applied coupled tensor factorization technique to link prediction with heterogeneous data. They modelled this problem as filling the missing entries in a relational dataset, and proposed to address this issue by coupled tensor factorization. Wang et al. [88] proposed to model urban traffic congestions in different regions and time slots as a tensor, and utilized a tensor-matrix co-factorization method to estimate city-level traffic congestions. Zheng et al. [111] for the first time studied the urban noise diagnose problem with a coupled tensor-matrix factorization model. Their model can effectively discover the major types of noise in different areas of New York. As coupled tensor-matrix factorization is usually time consuming, some recent works also proposed some efficient solutions. Rendle and Thieme [69] proposed the factorization model Pairwise Interaction Tensor Factorization to explicitly model the pairwise interactions between users, items and tags for item recommendation. Evangelos et al. [64] proposed Turbo-SMT algorithm which can accelerate coupled tensor-matrix factorization by 200x. Tensor decompositions have been very popular and successful in achieving state-of-the-art performance in many applications. However, our proposed weighted coupled tensor-matrix factorization model is different from existing models in both the weight setting mechanism and the reformulation of the CP model into the coupled factorization, which bring more improvement in accuracy and efficiency.

5.3 Framework Overview

Figure 5.2 presents the framework of our work, which consists of three stages, i.e., grouping, inferring and retrieving. In the grouping stage, orders are grouped into different clusters $C_1, C_2, \ldots, C_m$ based on their characteristics so as to enrich the interactions between orders and drivers. In the inferring stage, we build a three-

mode tensor $\mathcal{P}$, and each entry $p_{i,j,t}$ stores the accepting probability of driver dr_j to an order cluster C_i in a specific time slot t. Some of the entries in the tensor can be directly derived from historical data, while others are missing and can be inferred with the proposed CWTF method. In the retrieving stage, given a new order o_i and time t, we find the most similar cluster C_k to it, and then use cluster accepting probability $p_{k,j,t}$ as the prediction result of driver dr_j to the order o_i in time slot t. The framework is general that different clustering and tensor factorization methods can be fitted into it for order accepting probability prediction.

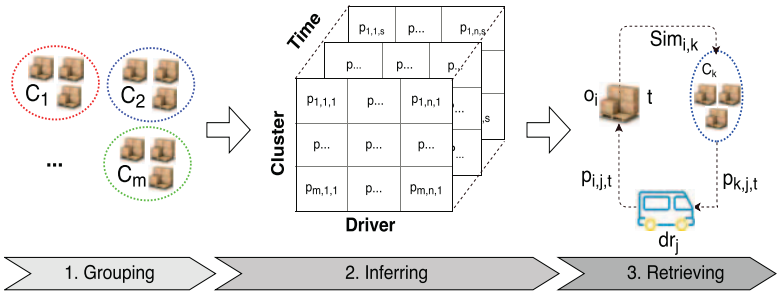


Figure 5.2: Framework for order accepting probability prediction

5.4 Feature Extraction

The essential objective of this work is to explore the preferences of drivers to different orders. Therefore, we extract features from two domains of data: features of orders to capture their characteristics and features of drivers to capture their behaviors, respectively. The extracted features are then utilized to analyze the preferences of drivers to different orders using the proposed model.

5.4.1 Order Feature Extraction

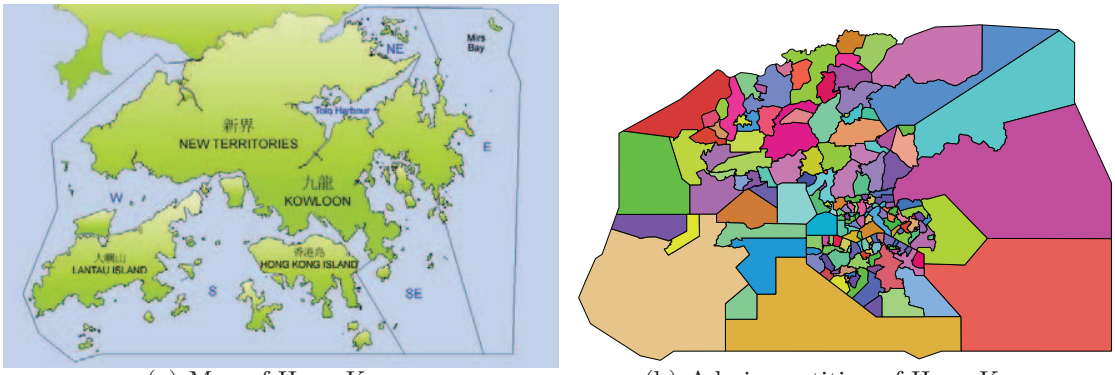
For each order in on-demand logistics services, we extract the following four categories of features that may affect the preferences of drivers.

- *Temporal features*: the hour of the day, the day of the week, and whether the day is a public holiday when the order is made.

- *Spatial features*: the origin and destination regions of the order, where origin region is the region in which the order is made, and the destination region is the region to which the goods will be delivered. Specifically, the data used for evaluation in this work is from Hong Kong and we partition the whole city into 140 regions according to its administration partition, which is illustrated in Figure 5.3. Note that, the solution proposed in this work can also be applicable for other partition methods, e.g., grid partition.

- *Price relevant features*: the price computed based on the start and the end locations of the order and the travelling distance between the two locations, and the bonus provided by the user to attract drivers.

- *Personalized requirement features*: the additional requirements specified by users, e.g., whether the user has pets and whether the user needs help to carry. Specifically, there are 27 types of additional requirements in total for users to mark in the data used by this work.



(a) Map of Hong Kong (b) Admin partition of Hong Kong

Figure 5.3: The map and the administration partition of Hong Kong.

We then use one-hot encoding to transform all categorical features to numerical values before further processing. In addition, each order is associated with an accepting time (i.e., the time costed for the order to be accepted by the driver who completed it) that serves as the indicator of order accepting probability (more details in Section 5.6.2).

5.4.2 Driver Feature Extraction

In addition to order features, we also extract the following features for each driver to obtain the behavior information.

- *Order accepting features*: the count, the average response time, the average price, and the average bonus of all the orders accepted by the driver.
- *Spatial features*: the top-5 regions in which the driver accepts the most orders, capturing the locality preference of the driver.
- *Temporal features*: working time and idle ratio. The working time is quite flexible for drivers in on-demand logistics services because they have the autonomy to decide when to work. Therefore, with the GPS trajectory data with geographic location and status (busy or idle, as shown in Figure 5.4), we compute the working time as well as the idle ratio, i.e., the ratio of time that the driver is idle, for each driver.
- *Order requirement features*: the number of orders with each particular type of additional requirement. As stated previously, many orders are associated with some additional requirements, which may greatly affect drivers' willing to accept these orders. For example, some drivers may directly reject the orders with additional requirements which they cannot satisfy, e.g., speaking English. Therefore, for each driver, we count the number of orders accepted in history with respect to each particular type of additional requirement.

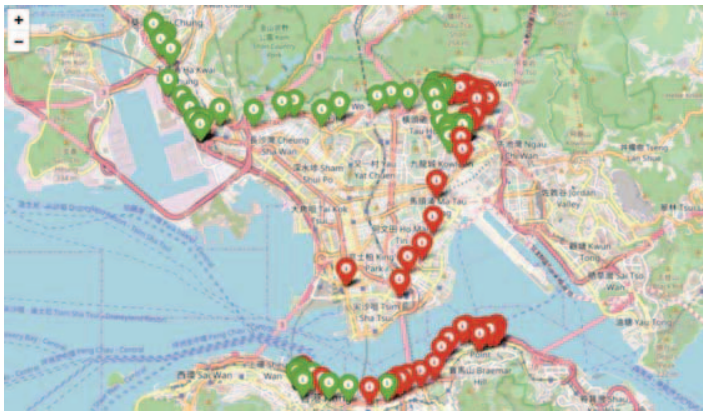


Figure 5.4: The GPS trajectory of a driver, where green balloon markers indicate that the driver is idle and red balloon makers mean that the driver is busy with an order.

5.5 Methodology

5.5.1 Notation and Basic Operations

The mode of a tensor is the number of dimensions, also known as ways. An N-mode tensor is represented as $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, where I_n is the cardinality of its n-th mode, $n \in \{1, 2, \cdots, N\}$. An element of a vector $\mathbf{a}$, a matrix $\mathbf{A}$, or a tensor $\mathcal{A}$ is denoted by a_i, $a_{i,j}$, $a_{i,j,k}$, etc., depending on the number of modes. The *Hadamard product* and the *outer product* [45] are denoted by $*$ and $\otimes$, respectively. The diagonal matrix and the trace of matrix $\mathbf{A}$ are denoted by $diag(\mathbf{A})$ and $\text{tr}(\mathbf{A})$, respectively. A slice of a three-mode tensor is a matrix, extracted by fixing all modes but two. Specifically, the frontal slice of a three-mode tensor $\mathcal{A}$ is $a_{:,:,k}$, denoted as $\mathbf{A}_k$.

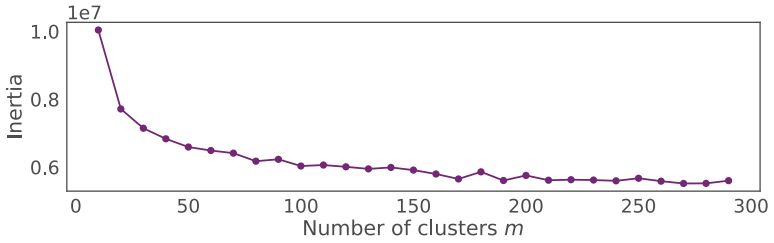
5.5.2 Order Clustering

With the extracted features for orders, we cluster all orders into different groups using Mini Batch K-Means [70], which is much more efficient than other clustering algorithms when applied for a large number of orders. To determine the cluster

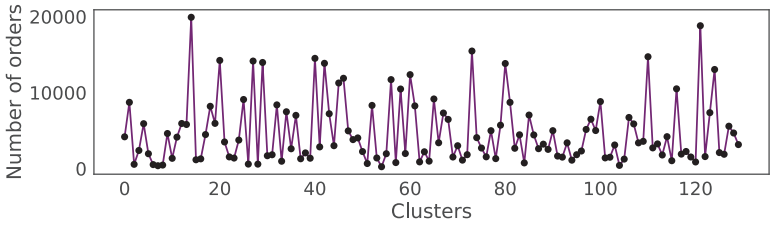
number m, we evaluate the inertia I (i.e., the sum of the square distances of samples to the nearest cluster center) of each m by the following formula.

$$I = \sum_{o} \|\mathbf{f}_o - \mathbf{f}_c\|_2^2 \tag{5.1}$$

where $\mathbf{f}_o$ is the feature vector of order o, and $\mathbf{f}_c$ is the features of the nearest cluster center of order o. Essentially, inertia is used to quantify the cohesion of generated clusters and the smaller the better. Figure 5.5(a) illustrates the inertia with different values of m. As suggested by Figure 5.5(a), we set m to 130 because a smaller m has higher processing efficiency and the inertias of other m values larger than 130 keep almost the same.



(a) The inertia for different values of m in Mini Batch K-Means



(b) The distribution of orders over different clusters

Figure 5.5: The clustering of orders.

Figure 5.5(b) illustrates the clustering results when setting m to 130. Without loss of generality, we assume that all orders are clustered into clusters $C_1, C_2, \ldots, C_m$. For each cluster C_i, we compute the average values of the features of all the orders in C_i as the corresponding cluster features. For example, the feature vector $\mathbf{f}_{C_i}$ of

cluster C_i is computed by

$$\mathbf{f}_{C_i} = \frac{\sum_{o \in C_i} \mathbf{f}_o}{|C_i|} \tag{5.2}$$

where $|C_i|$ is the number of orders in cluster C_i. We organize the features of all clusters into a cluster feature matrix $\mathbf{X}$.

As suggested by Figure 5.5(b), the number of orders varies greatly across clusters. With one cluster of few orders, we cannot get enough information about it. To deal with this issue, as discussed in Section 5.5.3, we will introduce a weight for each cluster with respect to the size to indicate its confidence for analysis.

5.5.3 The CWTF Model

Weighted Slice-based CP Reformulation

CANDECOMP/PARAFAC (CP) factorization [36] is a widely used technique for exploring and extracting the underlying structure of multi-way data, which is critical to the development of our proposed method. The CP factorization of a three-mode tensor $\mathcal{P} \in \mathbb{R}^{M \times N \times K}$ is approximated by 3 *latent* matrices $\mathbf{U}, \mathbf{V}$ and $\mathbf{T}$, such that

$$\mathcal{P} \approx \sum_{r=1}^{R} \mathbf{u}_r \otimes \mathbf{v}_r \otimes \mathbf{t}_r = [\![\mathbf{U}, \mathbf{V}, \mathbf{T}]\!] \tag{5.3}$$

where $[\![\cdot]\!]$ is defined as the CP factorization operator for shorthand and the latent matrices $\mathbf{U} \in \mathbb{R}^{M \times R} = [\mathbf{u}_1, \cdots, \mathbf{u}_R]$, $\mathbf{V} \in \mathbb{R}^{N \times R} = [\mathbf{v}_1, \cdots, \mathbf{v}_R]$, $\mathbf{T} \in \mathbb{R}^{K \times R} = [\mathbf{t}_1, \cdots, \mathbf{t}_R]$, and R is referred to as the *rank* of the tensor $\mathcal{P}$, indicating the number of factors. To increase memory efficiency and facilitate parallel computation, Eq. (5.3) can be reformulated with respect to the frontal slice $\mathbf{P}_k$ of the tensor $\mathcal{P}$ [14]:

$$\mathbf{P}_k \approx \mathbf{U}\mathbf{S}_k\mathbf{V}^T \tag{5.4}$$

where $\mathbf{S}_k = diag(\mathbf{T}(\mathbf{k}:))$ and $k = 1, 2, ...K$.

In this work, we build a three-mode tensor $\mathcal{P}$, and each entry $p_{i,j,t}$ stores the accepting probability of a driver dr_j to an order cluster C_i in a specific time slot

t. Some of the entries in the tensor can be directly derived from historical data, while others are inferred with the above slice-based CP reformulation efficiently. For the entries that can be derived from historical data, each entry $p_{i,j,t}$ is computed by averaging the accepting probabilities of orders in $O_{i,j,t}$, which denotes the orders in cluster C_i accepted by driver dr_j in time slot t. In addition, we build a corresponding weight tensor $\mathcal{W}$, where each entry $w_{i,j,t}$ indicates the confidence of the accepting probability $p_{i,j,t}$. A smaller $w_{i,j,t}$ means lower confidence on $p_{i,j,t}$, and the weight $w_{i,j,t}$ is computed by

$$w_{i,j,t} = \frac{min(|O_{i,j,t}|, 5)}{5} \cdot (0.5 + \frac{0.05}{max(\sigma_{i,j,t}, 0.1)}) \tag{5.5}$$

where $|O_{i,j,t}|$ computes the number of orders in $O_{i,j,t}$, and $\sigma_{i,j,t}$ is the standard deviation of the accepting probabilities of orders in $O_{i,j,t}$. Specifically, term $\frac{min(|O_{i,j,t}|,5)}{5}$ considers the number of orders, so that if there are less orders in $O_{i,j,t}$, then the weight corresponds to $p_{i,j,t}$ is smaller. We set the upper bound to 5 based on the statistical analysis of the data used in this work. Generally, if the number of orders is larger, we can set the upper bound to a larger value, otherwise a smaller value is assigned. The second term $(0.5 + \frac{0.05}{max(\sigma_{i,j,t},0.1)})$ considers the distribution of order accept probabilities, so that if the order accepting probabilities in $O_{i,j,t}$ vary a lot, then the corresponding weight is smaller. The three numbers 0.05, 0.5 and 0.1 are introduced to control the contributions of this term, and make the value of $(0.5 + \frac{0.05}{max(\sigma_{i,j,t},0.1)})$ between 0 and 1.

As a result, the objective function for the weighted slice-based reformulation of CP is given as below.

$$\min_{\mathbf{U},\mathbf{V},\mathbf{T}} \mathcal{L} = \frac{1}{2} \sum_{k=1}^{K} \|\mathbf{W}_k * (\mathbf{P}_k - \mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})\|_F^2 \tag{5.6}$$

Coupled Tensor-matrix Factorization

The probability tensor is originally very sparse, i.e., many entries are missing due to either some drivers have never accepted orders in some clusters or time slots, or few orders were made in some time slots. Thus, we incorporate more information based on order and driver data to achieve a more accurate factorization. Specifically, we use cluster feature matrix $\mathbf{X}$ to help determine the cluster latent matrix $\mathbf{U}$, which is achieved by factorizing $\mathbf{X} \in \mathbb{R}^{M \times S}$ into low-rank latent matrices $\mathbf{U} \in \mathbb{R}^{M \times R}$ and $\mathbf{G} \in \mathbb{R}^{R \times S}$. Similarly, we factorize the driver feature matrix $\mathbf{Y} \in \mathbb{R}^{N \times T}$ into $\mathbf{V} \in \mathbb{R}^{N \times R}$ and $\mathbf{H} \in \mathbb{R}^{R \times T}$. That is to say, we collaboratively factorize $\mathcal{P}$ to $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{T}$, with cluster feature matrix $\mathbf{X}$ to low-rank latent matrices $\mathbf{U}$ and $\mathbf{G}$, and driver feature matrix $\mathbf{Y}$ to $\mathbf{V}$ and $\mathbf{H}$, where they share the latent matrices $\mathbf{U}$ and $\mathbf{V}$. In this way, cluster feature matrix $\mathbf{X}$ and driver feature matrix $\mathbf{Y}$ would contribute to a more accurate factorization of $\mathcal{P}$ through the shared matrices $\mathbf{U}$ and $\mathbf{V}$. The objective function is formulated as below, where λ_1 and λ_2 are the parameters to control the contribution of each component during the coupled factorization.

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V},\mathbf{T},\mathbf{G},\mathbf{H}} \mathcal{L} = & \frac{1}{2} \sum_{k=1}^{K} \|\mathbf{W}_k * (\mathbf{P}_k - \mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})\|_F^2 + \\ & \frac{\lambda_1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{G}\|_F^2 + \frac{\lambda_2}{2}\|\mathbf{Y} - \mathbf{V}\mathbf{H}\|_F^2 \end{aligned} \tag{5.7}$$

Correlation Utilization and Group Feature Selection

To further alleviate the data sparsity problem, we also utilize correlations between clusters as well as drivers to help determine the cluster latent matrix $\mathbf{U}$ and the driver latent matrix $\mathbf{V}$. This is achieved by incorporating information in Laplacian matrices of their correlation graphs using $\mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{L}_u\mathbf{U})$ and $\mathrm{tr}(\mathbf{V}^{\mathrm{T}}\mathbf{L}_v\mathbf{V})$, respectively.

For cluster correlation, we have $\mathbf{L}_u = (\mathbf{D} - \mathbf{Z}) \in \mathbb{R}^{M \times M}$, where $\mathbf{Z}$ is a correlation

matrix of order clusters, with each entry $z_{i,j}$ representing the value of absolute Pearson correlation between order cluster C_i and C_j, and $\mathbf{D}$ is a diagonal matrix with the diagonal entry $d_{i,i} = \sum_i z_{i,j}$. Then, $\mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{L}_u\mathbf{U})$ is obtained through the following deduction, which guarantees that two clusters C_i and C_j with a higher correlation (*i.e.*, $z_{i,j}$ is larger) should also have a closer distance between the vectors $\mathbf{u}_i$ and $\mathbf{u}_j$ in the matrix $\mathbf{U}$ [111].

$$\begin{aligned}
&\frac{1}{2}\sum_{i,j} z_{i,j}\|\mathbf{u}_i - \mathbf{u}_j\|_2^2 = \sum_{i,j}\mathbf{u}_i z_{i,j}\mathbf{u}_i^{\mathrm{T}} - \sum_{i,j}\mathbf{u}_i z_{i,j}\mathbf{u}_j^{\mathrm{T}} \\
&= \sum_i \mathbf{u}_i d_{i,i}\mathbf{u}_i^{\mathrm{T}} - \sum_{i,j}\mathbf{u}_i z_{i,j}\mathbf{u}_j^{\mathrm{T}} \\
&= \mathrm{tr}(\mathbf{U}^T(\mathbf{D}-\mathbf{Z})\mathbf{U}) \\
&= \mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{L}_u\mathbf{U})
\end{aligned}$$

Similarly, we define $\mathbf{L}_v = (\tilde{\mathbf{D}} - \tilde{\mathbf{Z}}) \in \mathbb{R}^{N\times N}$ as the Laplacian matrix of the driver correlation graph, and consider the driver correlation as $\mathrm{tr}(\mathbf{V}^{\mathrm{T}}\mathbf{L}_v\mathbf{V})$. In this way, we utilize cluster correlations and driver correlations for accurate factorization of $\mathcal{P}$ to $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{T}$.

We also add an $\ell_{2,1}$-norm regularization on each latent matrix to encourage all feature matrices to select a common set of features and thus play the role of group feature selection [98]. Since the data contain various kinds of information, we assume only a small part of features are predictive.

Coupled Weighted Tensor-matrix Factorization

All the components mentioned in previous subsections are combined, and the final model is illustrated in Figure 5.6. We first set the corresponding weight tensor $\mathcal{W}$, and factorize $\mathcal{P}$ with cluster feature matrix $\mathbf{X}$ and driver feature matrix $\mathbf{Y}$, so that the factorized shared latent matrices $\mathbf{U}$ and $\mathbf{V}$ are more accurate. Additionally,

we utilize the cluster correlation matrix $\mathbf{Z}$, so that if two clusters C_i and C_j are correlated, they would have a closer distance between the vectors $\mathbf{u}_i$ and $\mathbf{u}_j$ in the matrix $\mathbf{U}$. We utilize the driver correlation matrix $\tilde{\mathbf{Z}}$ in the same way to make $\mathbf{V}$ more accurate.

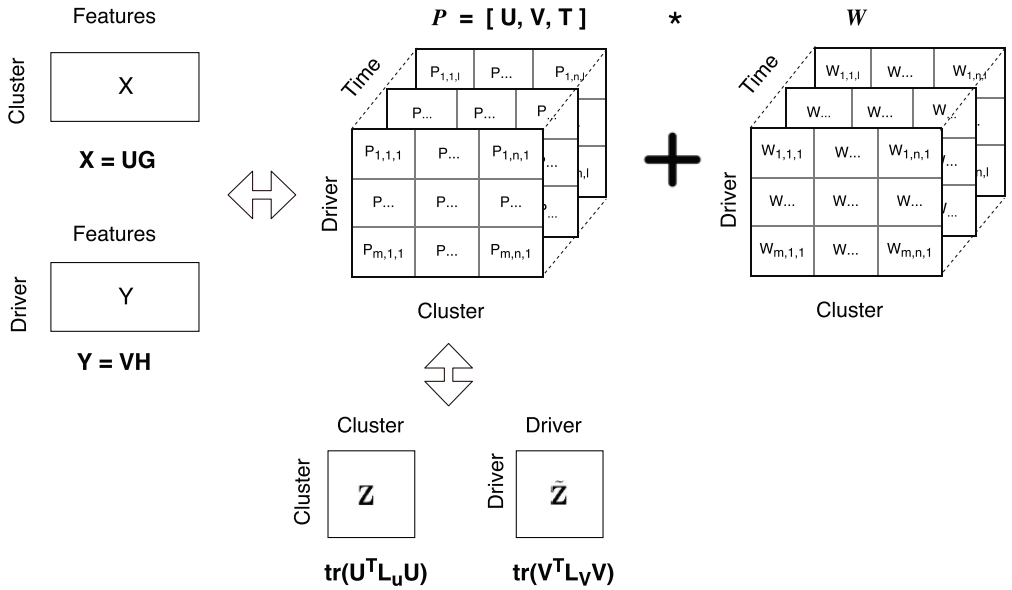


Figure 5.6: Coupled Weighted Tensor-matrix Factorization

The objective function of the proposed CWTF model is formally given as below.

$$\min_{\mathbf{U},\mathbf{V},\mathbf{T},\mathbf{G},\mathbf{H}} \mathcal{L} = \frac{1}{2} \underbrace{\sum_{k=1}^{K} \|\mathbf{W}_k * (\mathbf{P}_k - \mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})\|_F^2}_{\text{accepting probability factorization}} +$$

$$\frac{\lambda_1}{2} \underbrace{\|\mathbf{X} - \mathbf{U}\mathbf{G}\|_F^2}_{\text{cluster factorization}} + \frac{\lambda_2}{2} \underbrace{\|\mathbf{Y} - \mathbf{V}\mathbf{H}\|_F^2}_{\text{driver factorization}} +$$

$$\frac{\lambda_3}{2} \underbrace{\mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{L}_u\mathbf{U})}_{\text{cluster correlation}} + \frac{\lambda_4}{2} \underbrace{\mathrm{tr}(\mathbf{V}^{\mathrm{T}}\mathbf{L}_v\mathbf{V})}_{\text{driver correlation}} +$$

$$\frac{\lambda_5}{2} \underbrace{(\|\mathbf{U}\|_{2,1} + \|\mathbf{V}\|_{2,1} + \|\mathbf{T}\|_{2,1} + \|\mathbf{G}\|_{2,1} + \|\mathbf{H}\|_{2,1})}_{\text{row-wise sparsity regularization}} \tag{5.8}$$

where $\sum_{k=1}^{K} \|\mathbf{W}_k * (\mathbf{P}_k - \mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})\|_F^2$ is to control the error of factorizing accepting probability tensor $\mathcal{P}$; $\|\mathbf{X} - \mathbf{U}\mathbf{G}\|_F^2$ is to control the error of factorization of cluster feature matrix $\mathbf{X}$; $\|\mathbf{Y} - \mathbf{V}\mathbf{H}\|_F^2$ is to control the error of factorization of driver feature matrix $\mathbf{Y}$; $\mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{L}_u\mathbf{U})$ and $\mathrm{tr}(\mathbf{V}^{\mathrm{T}}\mathbf{L}_v\mathbf{V})$ is to utilize the cluster and driver correlations; λ_1, λ_2, λ_3 and λ_4 are used to control the contribution of each component during the coupled factorization, and λ_5 is a regularization parameter preventing overfitting and controlling group feature selection.

Optimization Algorithm

We devise an efficient iterative algorithm to solve the non-convex objective function in Eq. (5.8), by alternatively updating one variable while fixing others until convergence. Besides, as the $\ell_{2,1}$-norm is not continuous on the origin, directly minimizing the compound $\ell_{2,1}$ objective function is difficult. We employ the half-quadratic minimization method [61] to approximate the $\ell_{2,1}$-norm. Taking $\mathbf{U}$ as an example, we transform:

$$\|\mathbf{U}\|_{2,1} \approx \mathrm{tr}(\mathbf{U}^{\mathrm{T}}\mathbf{Q}_u\mathbf{U}) \tag{5.9}$$

where $\mathbf{Q}_u$ is a diagonal matrix with the i-th diagonal element as $q_{i,i}^u = \frac{1}{2\|\mathbf{u}_i\|_2}$. Similarly, we introduce the auxiliary variables $\mathbf{Q}_v$, $\mathbf{Q}_t$, $\mathbf{Q}_g$, and $\mathbf{Q}_h$ for $\mathbf{V}$, $\mathbf{T}$, $\mathbf{G}$, and $\mathbf{H}$, respectively.

Based on the above transformation, we calculate the derivative of Eq. (5.8) with respect to $\mathbf{U}, \mathbf{V}, \mathbf{T}_k, \mathbf{G}$ and $\mathbf{H}$, respectively, and set them to zero as shown in Eq. (5.10). Particularly, we make the following deduction to calculate the derivative of Eq. (5.8) with respect to $\mathbf{T}_k$.

$$
\begin{aligned}
&\frac{\partial \frac{1}{2}\| \sum_{k=1}^{K} \mathbf{W}_k * (\mathbf{P}_k - \mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})\|_F^2}{\partial \mathbf{T}_k} \\
= &\frac{\partial(- < \mathbf{W}_k * \mathbf{P}_k, \mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}}) >)}{\partial \mathbf{T}_k} \\
&+ \frac{\partial \frac{1}{2}(\|\mathbf{W}_k * \mathbf{P}_k\|_F^2 + \|\mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})\|_F^2)}{\partial \mathbf{T}_k} \\
= &\frac{\partial(-\mathrm{tr}((\mathbf{W}_k * \mathbf{P}_k)^{\mathrm{T}}(\mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})))}{\partial \mathbf{T}_k} \\
&+ \frac{\partial(\mathrm{tr}((\mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}}))^{\mathrm{T}}\mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})))}{\partial \mathbf{T}_k} \\
= &\frac{\partial(-\mathrm{tr}((\mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}}))^{\mathrm{T}}(\mathbf{W}_k * \mathbf{P}_k)))}{\partial \mathbf{T}_k} \\
&+ \frac{\partial(\mathrm{tr}((\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})^{\mathrm{T}}(\mathbf{W}_k * \mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}}))))}{\partial \mathbf{T}_k} \\
= &\frac{\partial(-\mathrm{tr}((\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}})^{\mathrm{T}}(\mathbf{W}_k * \mathbf{W}_k * \mathbf{P}_k)))}{\partial \mathbf{T}_k} \\
&+ \frac{\partial(\mathrm{tr}(diag(\mathbf{T}_k)(\mathbf{U}^{\mathrm{T}}(\mathbf{W}_k * \mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}}))\mathbf{V}))}{\partial \mathbf{T}_k} \\
= &\frac{\partial(-\mathrm{tr}(diag(\mathbf{T}_k)(\mathbf{U}^{\mathrm{T}}(\mathbf{W}_k * \mathbf{W}_k * \mathbf{P}_k)\mathbf{V})))}{\partial \mathbf{T}_k} \\
&+ \frac{\partial(\mathbf{T}_k diag(\mathbf{U}^{\mathrm{T}}(\mathbf{W}_k * \mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}}))\mathbf{V})}{\partial \mathbf{T}_k} \\
= &\, diag(\mathbf{U}^{\mathrm{T}}(\mathbf{W}_k * \mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}} - \mathbf{P}_k))\mathbf{V})
\end{aligned}
$$

$$
\begin{cases}
\dfrac{\partial \mathcal{L}}{\partial \mathbf{U}} = \sum\limits_{k=1}^{K} \mathbf{W}_k * \mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}} - \mathbf{P}_k)\mathbf{V}\mathbf{S}_k \\
\qquad + \lambda_1(\mathbf{U}\mathbf{G} - \mathbf{X})\mathbf{G}^{\mathrm{T}} + \lambda_3\mathbf{L}_u\mathbf{U} + \lambda_5\mathbf{Q}_u\mathbf{U} = 0 \\
\dfrac{\partial \mathcal{L}}{\partial \mathbf{V}} = \sum\limits_{k=1}^{K} (\mathbf{W}_k * \mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}} - \mathbf{P}_k)\mathbf{U}\mathbf{S}_k)^{\mathrm{T}} \\
\qquad + \lambda_2(\mathbf{V}\mathbf{H} - \mathbf{Y})\mathbf{H}^{\mathrm{T}} + \lambda_4\mathbf{L}_v\mathbf{V} + \lambda_5\mathbf{Q}_v\mathbf{V} = 0 \\
\dfrac{\partial \mathcal{L}}{\partial \mathbf{T}_k} = diag(\mathbf{U}^{\mathrm{T}}(\mathbf{W}_k * \mathbf{W}_k * (\mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}} - \mathbf{P}_k))\mathbf{V}) \\
\qquad + \lambda_5\mathbf{Q}_t\mathbf{T}_k = 0 \\
\dfrac{\partial \mathcal{L}}{\partial \mathbf{G}} = \lambda_1\mathbf{U}^{\mathrm{T}}(\mathbf{U}\mathbf{G} - \mathbf{X}) + \lambda_5\mathbf{Q}_g\mathbf{G} = 0 \\
\dfrac{\partial \mathcal{L}}{\partial \mathbf{H}} = \lambda_2\mathbf{V}^{\mathrm{T}}(\mathbf{V}\mathbf{H} - \mathbf{Y}) + \lambda_5\mathbf{Q}_h\mathbf{H} = 0
\end{cases}
\tag{5.10}
$$

Eq. (5.10) involves solving five linear systems. We use conjugate gradient (CG) method to solve it, which is an effective and widely used technique [42], and only needs to perform matrix multiplications of Eq. (5.10). The overall algorithm is summarized in Algorithm 5.1.

Algorithm 5.1 Algorithm to solve the problem in Eq. (5.8)

Input: Accepting probability tensor $\mathcal{P}$, weight tensor $\mathcal{W}$, cluster feature matrix $\mathbf{X}$, driver feature matrix $\mathbf{Y}$, Laplacian matrices $\mathbf{L}_u, \mathbf{L}_v$ and regularization parameters

Output: $\mathbf{U}$, $\mathbf{V}$, $\mathbf{T}$

1: Initialize $\mathbf{U}, \mathbf{V}, \mathbf{T}, \mathbf{G}, \mathbf{H}$ randomly
2: Calculate the diagonal matrices $\mathbf{Q}_u$, $\mathbf{Q}_v$, $\mathbf{Q}_t$, $\mathbf{Q}_g$, $\mathbf{Q}_h$
3: **repeat**
4: Update $\mathbf{U}$, $\mathbf{V}$, $\mathbf{T}$, $\mathbf{G}$, $\mathbf{H}$ by solving Eq. (5.10) using CG method
5: Update the diagonal matrices $\mathbf{Q}_u$, $\mathbf{Q}_v$, $\mathbf{Q}_t$, $\mathbf{Q}_g$, $\mathbf{Q}_h$ by

$$
q^u_{i,i} = \frac{1}{2\|\mathbf{u}_i\|_2}, q^v_{i,i} = \frac{1}{2\|\mathbf{v}_i\|_2}, q^t_{i,i} = \frac{1}{2\|\mathbf{t}_i\|_2},
$$
$$
q^g_{i,i} = \frac{1}{2\|\mathbf{g}_i\|_2}, q^h_{i,i} = \frac{1}{2\|\mathbf{h}_i\|_2}
$$

6: **until** convergence

Convergence Analysis. The objective monotonically decreases in each iteration and it has a lower bound (proof can be derived in a similar manner as in [38]).

Therefore, Algorithm 5.1 can converge to a local minimum of the objective function in Eq. (5.8).

5.5.4 Prediction

After inferring the missing entries in the probability tensor $\mathcal{P}$, we have the knowledge about each driver's preference to each cluster of orders in each time slot. Therefore, with the probability tensor $\mathcal{P}$, we can predict the accepting probability for new orders. Concretely, given a new order o_i in time slot t, we first calculate the most similar order cluster C_k for o_i by

$$C_k = \arg \min_{C \in \{C_1, C_2, \ldots, C_m\}} \|\mathbf{f}_{o_i} - \mathbf{f}_C\|_2^2 \tag{5.11}$$

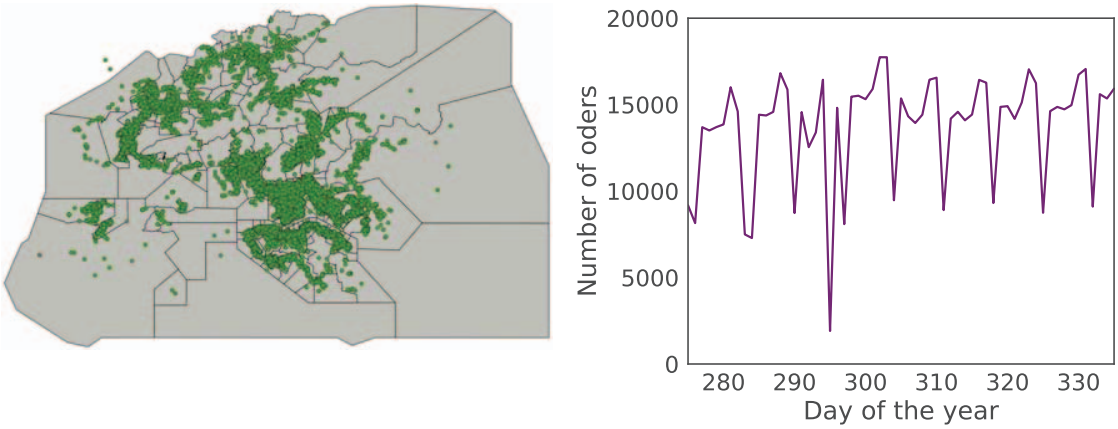
where $\mathbf{f}_{o_i}$ is the feature vector of order o_i, and $\mathbf{f}_C$ is the feature vector of cluster C. Intuitively, cluster C_k has the minimum Euclidean distance to order o_i in the feature space. Then, the accepting probability $p_{k,j,t}$ of driver j to order o_i in time slot t is retrieved from the probability tensor $\mathcal{P}$ and returned as the predicted result.

5.6 Experiments

5.6.1 Datasets

In this work, we take a real-world dataset from a leading logistics company in Hong Kong, as a use case to present the application of our solution to order accepting probability prediction. We collect around 800,000 orders within 61 consecutive days in 2016. Figure 5.7 illustrates the spatial and temporal distributions of the collected orders. The features of collected orders are extracted according to Section 5.4.1.

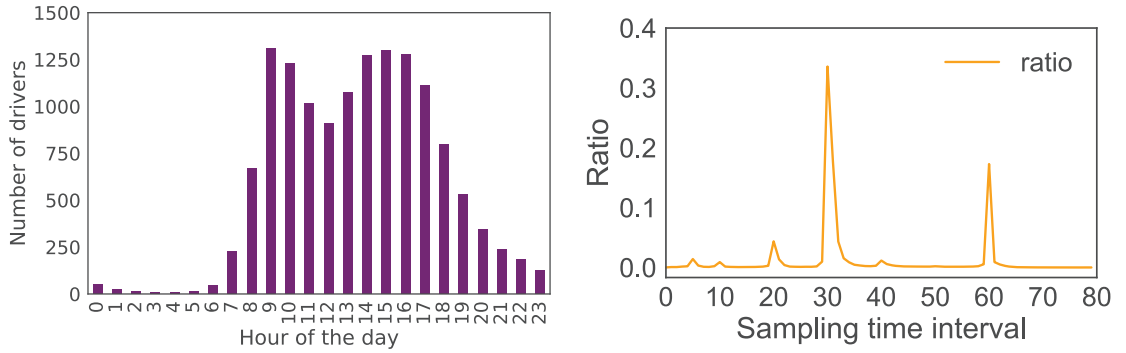
Among the collected orders, we have around 4,000 distinct regular drivers. Figure 5.8(a) presents the average number of active drivers in each hour of the day. For each particular hour, the number of active drivers is computed by counting the



(a) Spatial distribution (b) Temporal distribution

Figure 5.7: The spatial and temporal distributions of collected orders

number of distinct drivers who have picked at least one order within that hour. According to Figure 5.8(a), we have around 1,250 active drivers during the peak hours. In addition, the drivers are required to report their locations to the order dispatching center. As illustrated in Figure 5.8(b), most drivers report their GPS locations with a time interval less than one minute. The features of drivers are extracted according to Section 5.4.2.



(a) Number of drivers in each hour of the day (b) Distribution of sampling time intervals of GPS records

Figure 5.8: The average number of active drivers in different hours of a day and the distribution of sampling time intervals of GPS records

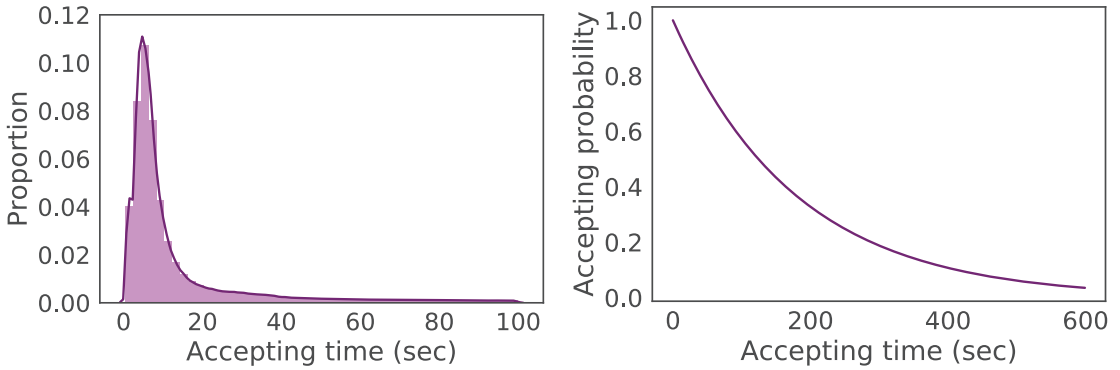
Note that, we conduct some data pre-processing to remove the outliers before extracting the features of orders and drivers. Concretely, we utilize both temporal

and spatial filters to remove those orders and drivers that have abnormal locations and time records. In addition, we also remove the time slots with abnormal numbers of orders, e.g., the time slots on day 295 in Figure 5.7(b).

5.6.2 Transform Accepting Time to Accepting Probability

In practice, it is difficult to get the order accepting probability directly. However, we can compute the order accepting probability indirectly from other indicators. In on-demand logistics services, users make orders while drivers provide services to satisfy the order demand. Particularly, once an order is made, the time costed for it to be accepted by a driver can capture its attraction to the driver and express the preference of the driver. Therefore, we compute the order accepting probability based on the order accepting time.

Figure 5.9(a) illustrates the distribution of order accepting time for all the collected orders. According to the figure, most orders are accepted within 40 seconds.



(a) Distribution of order accepting time (b) Order accepting probability vs. accepting time

Figure 5.9: The distribution of order accepting time, and order accepting probability vs. order accepting time

To compute the order accepting probability based on order accepting time, we introduce an exponential function to convert the accepting time to the accepting probability. Given order accepting time t, the corresponding order accepting proba-

bility p is computed by

$$p = e^{at} \tag{5.12}$$

where a is a parameter set by the logistics service providers according to their requirements for the service. For example, in this work, we assume that the accepting time of 40 seconds corresponds to an order accepting probability of 0.8. We can then get the value of a by solving the equation e^{40a}=0.8. The corresponding curve of the order accepting probability is plotted in Figure 5.9(b). Note that the longer the accepting time is, the lower the accepting probability is. The pivot probability value (i.e., 0.8) to determine the value of a could be set flexibly according to the application scenarios.

5.6.3 Settings and Baselines

Among the collected 840,538 orders within 61 consecutive days, the orders in the first 46 days are used for training while the others are used for testing. Without loss of generality, we set a time slot to 60 minutes in this work.

In order to demonstrate the effectiveness of our CWTF method, we compared it with the following baselines.

- **Ridge**: Ridge regression [37] is a commonly used method for regression problem by imposing a penalty on the size of coefficients of ordinary least squares.
- **Lasso**: The Lasso [81] is another widely used method for regression problem by imposing a penalty on the number of nonzero coefficients of ordinary least squares.
- **EN**: The Elastic Net [115] is a regularized regression method that linearly combines the ℓ_1 and ℓ_2 penalties of the lasso and ridge models, providing a trade-off option between the sparsity of coefficients and the sizes of coefficients.

- **RF**: Random Forest [12] regression produces a prediction model in the form of an ensemble of decision tree models through bagging.
- **GB**: Gradient Boosting [29] regression is an ensemble of tree models in a boosting manner. It always achieves optimal performance in various machine learning competitions.

For all these baselines methods, we join the order features with the driver features into a combined feature vector via driver *id*, and use the combined feature vector as input. We apply grid search on the important parameters to achieve the optimal performance of the baseline methods. Specifically, for Ridge, Lasso and Elastic Net, alpha is tuned; for Random Forest, multiple parameters including the number of trees in the forest, the minimum number of samples required to be at a leaf node, and the maximum depth of the tree, are tuned; for Gradient Boosting, multiple parameters including the number of boosting stages, the minimum number of samples required to be at a leaf node, the maximum depth of the individual regression, and the learning rate shrinking the contribution of each tree, are tuned.

Evaluation Metric. All methods are evaluated by the accuracy of their predicted accepting probability. The metric we use in the experiment is Mean Absolute Error (MAE):

$$MAE = \frac{\sum_{i=1}^{N} |y_{i,j_i,t_i} - \hat{y}_{i,j_i,t_i}|}{N} \tag{5.13}$$

where N is the number of predicted orders, y_{i,j_i,t_i} is the real accepting probability of driver j_i to order i in time t_i from testing data, and $\hat{y}_{i,j_i,t_i}$ is the predicted accepting probability.

5.6.4 Model Comparison

Figure 5.10 shows the performance comparison among various methods. The lower the MAE value is, the better the method performs. One can see that among the

three linear baseline methods, Ridge performs better than Lasso and EN. Among the two tree ensemble models, GB performs much better than RF. The proposed CWTF method achieves the best MAE value compared to all the baseline methods, which demonstrates a considerable performance improvement.

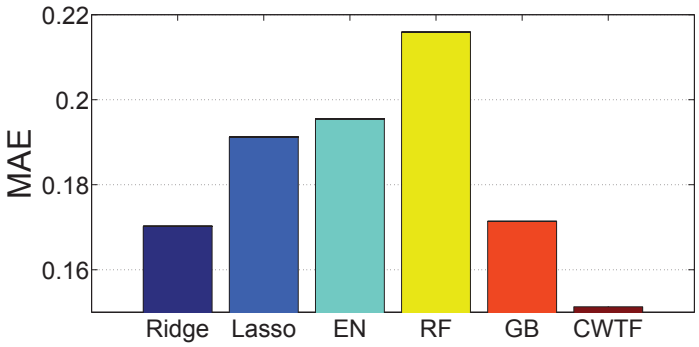


Figure 5.10: Performance comparison among various methods

5.6.5 Evaluation on Model Components

Recall in Section 5.5.3, there are several components in the proposed model. To evaluate each component of the CWTF model, we compare it with the following three variants.

- CWTF-Y: In this variant, the driver feature matrix $\mathbf{Y}$ is not included in the model. We can derive it by setting $\lambda_2 = 0$.
- CWTF-Lp: In this variant, the cluster and driver correlations are not included in the model. We can derive it by setting λ_3, $\lambda_4 = 0$.
- CWTF-W: In this variant, the weights of all the non-missing entries in weight tensor $\mathcal{W}$ are fixed to 1.

Figure 5.11 shows the performance comparison among these variants. Without the cluster and driver correlations, CWTF-Lp performs worst, which indicates fusing

correlation information is quite crucial in prediction. Without the proposed weight setting mechanism, CWTF-W cannot achieve a low MAE value, which shows that the weights contribute significantly in the prediction. CWTF-Y also performs slightly worse than CWTF, which indicates that the driver features have a positive effect on the prediction.

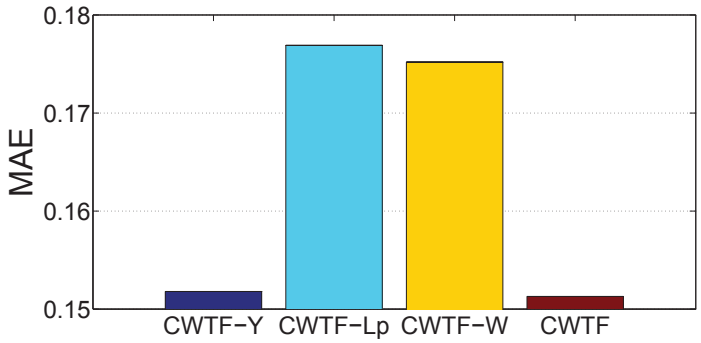


Figure 5.11: Performance comparison among variants

5.6.6 Evaluation on Parameters

We also investigate how the parameters affect the performance of the CWTF model. As shown in Section 5.5.3, there are 5 parameters in the proposed model, namely λ_1, λ_2, λ_3, λ_4, and λ_5. We range all the parameters from 10^{-2} to 10^{1}, and the parameters that achieve the optimal performance are $\lambda_1 = 0.5$, $\lambda_2 = 0.5$, $\lambda_3 = 10^{-1}$, $\lambda_4 = 10^{-1}$, and $\lambda_5 = 10^{-1}$. To analyze how each parameter affects the performance of the model, below we show the performance of varying one parameter while fixing other parameters as the optimal values.

Figure 5.12 shows the performance comparison of λ_1 and λ_2. We can see as the value of λ_1 increases, the MAE value of the model first decreases, achieves the optimal value when $\lambda_1 = 0.5$, and then increases. Similar trend can be observed on λ_2, while the MAE achieves the optimal value when $\lambda_2 = 0.5$. Recall that λ_1 and λ_2 are respectively to control the contributions of the cluster feature matrix and

the driver feature matrix. Figure 5.12 indicates that when first incorporating cluster and driver feature information into the model, the performance of the model gets improved. However, when this information is given too much attention, as the values of corresponding parameters increase, much noise is also introduced into the model, which impairs the performance of the model. Note that, although the two parameters show similar trends, cluster feature information with the corresponding parameter λ_1 contributes more and introduces less noise to the model, since the MAE values are smaller compared to that of driver feature information with λ_2.

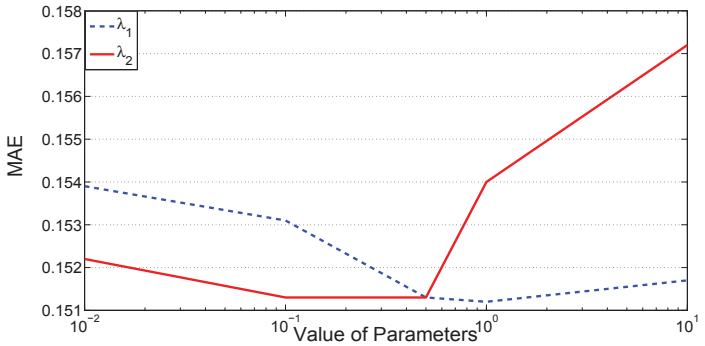


Figure 5.12: Evaluation on λ_1 and λ_2

Figure 5.13 shows the performance comparison of λ_3 and λ_4. We can see as the value of λ_3 increases, the MAE value of the model first decreases, achieves the optimal value when $\lambda_3 = 10^{-1}$, and then increases. Similar trend can be observed on λ_4. Recall that λ_3 and λ_4 are respectively to control the contributions of the cluster correlations and the driver correlations. Figure 5.13 indicates that when first incorporating the cluster and driver correlations into the model, the performance of the model gets improved. However, when this information is given too much attention, as the values of corresponding parameters increase, much noise is also introduced into the model. Note that, although the two parameters show similar trends, cluster correlations with the corresponding parameter λ_3 contribute more and introduce more noise to the model, since the MAE values are smaller at the

beginning and larger at the end compared to that of driver correlation information with λ_4.

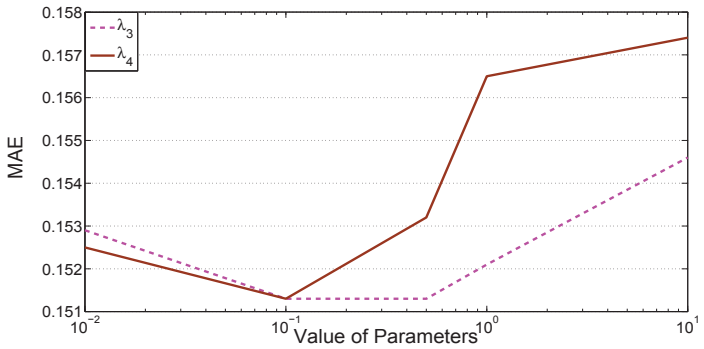


Figure 5.13: Evaluation on λ_3 and λ_4

Figure 5.14 shows the performance comparison of λ_5. We can see as the value of λ_5 increases, the MAE value of the model first decreases, achieves the optimal value when $\lambda_5 = 10^{-1}$, and then dramatically increases. Recall that λ_5 is to control overfitting and group feature selection. Figure 5.14 indicates that the group feature selection would help improve the performance of the model when the parameter is relatively small (e.g., 0.1). While the parameter is too large (e.g., 1), the model could be under-fitting or not many useful group features are selected, constraining the model performance from its best.

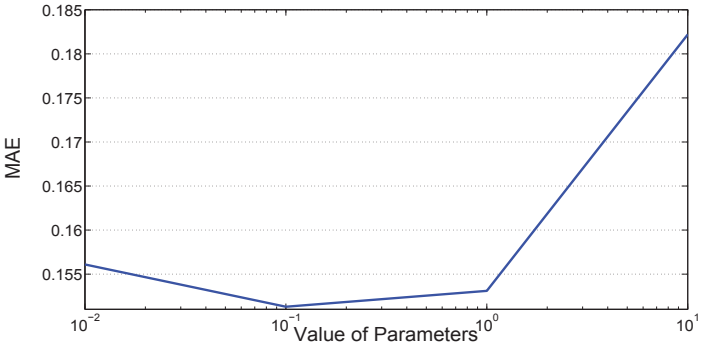


Figure 5.14: Evaluation on λ_5

5.6.7 Evaluation on Setting of Rank, Correlation and Convergence

To further investigate the performance of the proposed model, we evaluate another three important parts of CWTF.

The first part is to investigate how the setting of rank R would affect the performance of the model. We range the value of rank R from 10 to 180 to see its impact. The results are shown in Figure 5.15. As the value of rank increases, the MAE first decreases and achieves the optimal value when $R = 90$, and then increases. Recall in Section 5.5.3, rank R is the rank of tensor, and it also determines the rank of latent matrices. If the rank is set too low, the model may not be able to effectively learn from data; if the rank is set too high, the model may be overfitting.

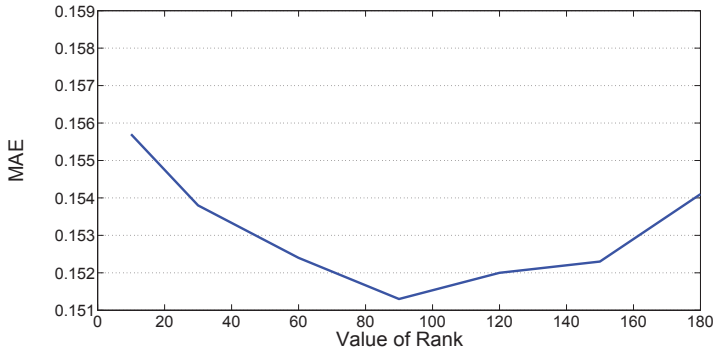


Figure 5.15: Evaluation on the setting of rank

The second part is to investigate how the setting of correlation affects the performance of the model. Previously in Section 5.5.3, we use Pearson correlation to measure the correlations among clusters and drivers. Now we also compare it to another widely used correlation metric: Spearman's rank correlation [77]. While Pearson correlation measures linear relationships, Spearman's rank correlation measures monotonic relationships including both linear and nonlinear relationships. The results are shown in Figure 5.16. The CWTF using Pearson correlation is slight-

ly better than CWTF using Spearman's rank correlation, which may indicate the correlations in the data is mainly linear.

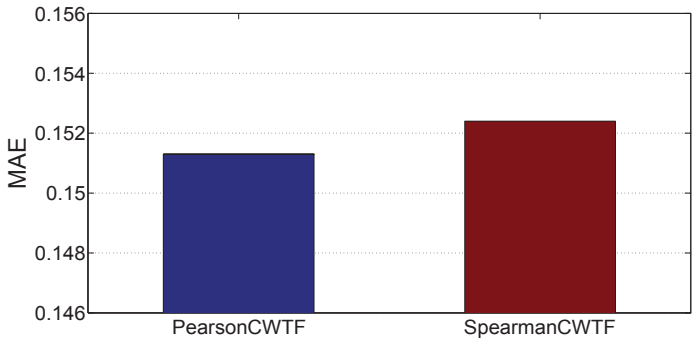


Figure 5.16: Evaluation on the setting of correlation

The last part is to investigate the convergence rate of CWTF. As shown in Figure 5.17, we can see that the value of the objective function decreases dramatically in the first 5 iterations. The decreasing trend becomes slower after 5 iterations, and finally the algorithm converges after 15 iterations.

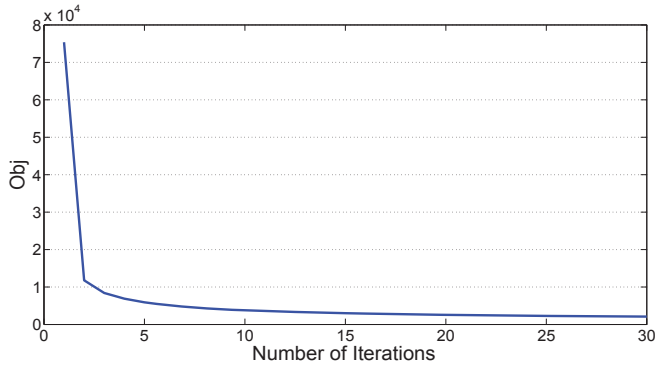


Figure 5.17: Evaluation on convergence

5.7 Conclusion

In this work, we study the important problem of order accepting probability prediction in logistics services. Specifically, we propose a three-stage framework with a

Coupled Weighted Tensor-matrix Factorization model to deal with the heterogeneous data fusion, sparsity and efficiency challenges. Experiments on real world datasets demonstrate the superiority of the proposed method. The performance of its several variants is also presented to show the effectiveness of each model component.

Chapter 6

Conclusions

Data analytics for urban computing is a challenging task in the complex and changeable environment. In addition, most of current research works focus on analyzing the data collected from a single domain, while there are actually multi-domain data existing in the urban spaces which make the analysis even harder to perform. The joint analysis of urban big data of different domains from multiple sources is very useful. It helps us to gain the hidden insights and enable intelligent decision-making. Specifically, cross domain data analytics has two advantages over traditional single domain data analytics. First, it offers a more comprehensive picture about the studied problems based on the information from different angles, where valuable insights can be discovered by analyzing the cross domain big datasets. Second, it improves decision making by complementing data sources for joint analysis, especially for the cases where data are insufficient in some domains.

Based on the above observations, I focus on ***Cross Domain Data Analytics for Urban Computing***, and study the problem of jointly analyzing data from different domains to ***generate hidden insights*** and ***enable intelligent decision-making*** with following three important applications in urban computing.

First, we study the problem of traffic congestion, and show how to jointly utilize data from three domains, namely GPS trajectories, road network and POI to generate

insights. Specifically, we propose a three-phase framework to explore the congestion correlation between road segments from multiple real world data. In the first phase, we extract congestion information on each road segment from GPS trajectories of over 10,000 taxis, define congestion correlation and propose a corresponding mining algorithm to find out all the existing correlations. In the second phase, we extract various features on each pair of road segments from road network and POI data. In the last phase, the results of the first two phases are input into several classifiers to predict congestion correlation. We further analyze the important features and evaluate the results of the trained classifiers through experiments. We found some important patterns that lead to a high/low congestion correlation, and they can facilitate building various transportation applications. In addition, we found that traffic congestion correlation has obvious directionality and transmissibility.

Second, we study the problem of order response time prediction to enable intelligent decision-making in logistics services by jointly considering both order historical records and driver GPS trajectories from two different domains. Specifically, we forecast order response time on current day by fusing data from order history and driver historical locations. Specifically, we propose Coupled Sparse Matrix Factorization (CSMF) to deal with the heterogeneous fusion and data sparsity challenges raised in this problem. CSMF jointly learns from multiple heterogeneous sparse data through the proposed weight setting mechanism therein. Experiments on real-world datasets demonstrate the effectiveness of our approach, compared to various baseline methods. The performances of many variants of the proposed method are also presented to show the effectiveness of each component.

Third, we extend the previous method to incorporate more context information by proposing a Coupled Weighted Tensor-matrix Factorization (CWTF) for accurate prediction on order accepting probabilities of van drivers, which would facilitate efficient order dispatching and improve user experience. Concretely, we propose a

three-stage framework with a Coupled Weighted Tensor-matrix Factorization method for order accepting probability prediction in logistics services. Specifically, orders are first grouped into clusters to enrich the sparse interactions between orders and drivers; then an accepting probability tensor with the three dimensions of driver, order cluster, and time is generated by a tensor-matrix factorization method that fuses order characteristics and driver behaviors in an efficient way; finally given a new order, the accepting probability of each driver is efficiently predicted by directly retrieving from the learned tensor. The experiment results on a large dataset from a famous app-based logistics platform, demonstrate the superiority of the proposed method against various baseline methods.

In summary, we propose new cross domain data analytics methods for urban computing to generate hidden insights and enable intelligent decision-making. The proposed methods are applied in three important applications involving both important application domains namely transportation, and domains that are less studied in terms of cross-domain data analytics, namely logistics. We identify the requirements and address the challenges therein, providing effective frameworks and solutions for practitioners as well as offering useful insights for future research.

Bibliography

[1] Evrim Acar, Rasmus Bro, and Age K. Smilde. Data fusion in metabolomics using coupled matrix and tensor factorizations. *Proceedings of IEEE*, 103(9):1602–1620, 2015.

[2] Evrim Acar, Gozde Gurdeniz, Morten A Rasmussen, Daniela Rago, Lars O Dragsted, and Rasmus Bro. Coupled matrix factorization with sparse factors to identify potential biomarkers in metabolomics. In *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, pages 1–8. IEEE, 2012.

[3] Luai Al Shalabi, Zyad Shaaban, and Basel Kasasbeh. Data mining: A preprocessing engine. *Journal of Computer Science*, 2(9):735–739, 2006.

[4] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[5] Yasushi Ando, Osamu Masutani, Hiroshi Sasaki, Hirotoshi Iwasaki, Yoshiaki Fukazawa, and Shinichi Honiden. Pheromone model: Application to traffic congestion prediction. In *Engineering Self-Organising Systems*, pages 182–196. Springer, 2006.

[6] Preeti Arunapuram, Jacob W Bartel, and Prasun Dewan. Distribution, correlation and prediction of response times in stack overflow. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*, pages 378–387. IEEE, 2014.

[7] Vani Vathsala Atluri and Hrushikesha Mohanty. Web service response time prediction using hmm and bayesian network. In *Intelligent Computing, Communication and Devices*, pages 327–335. Springer, 2015.

[8] Daniel Avrahami and Scott E Hudson. Responsiveness in instant messaging: predictive models supporting inter-personal communication. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 731–740. ACM, 2006.

[9] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990.*, pages 322–331, 1990.

[10] Vasudev Bhat, Adheesh Gokhale, Ravi Jadhav, Jagat Pudipeddi, and Leman Akoglu. Effects of tag usage on question response time. *Social Network Analysis and Mining*, 5(1):1–13, 2015.

[11] Guillaume Bouchard, Jason Naradowsky, Sebastian Riedel, Tim Rocktäschel, and Andreas Vlachos. Matrix and tensor factorization methods for natural language processing. In *ACL (Tutorial Abstracts)*, pages 16–18, 2015.

[12] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[13] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[14] Rasmus Bro. *Multi-way analysis in the food industry: models, algorithms, and applications*. PhD thesis, Københavns UniversitetKøbenhavns Universitet, LUKKET: 2012 Det Biovidenskabelige Fakultet for Fødevarer, Veterinærmedicin og NaturressourcerFaculty of Life Sciences, LUKKET: 2012 Institut for FødevarevidenskabDepartment of Food Science, LUKKET: 2012 Kvalitet og TeknologiQuality & Technology, 1998.

[15] Nikolay Burlutskiy, Andrew Fish, Nour Ali, and Miltos Petridis. Prediction of users' response time in q&a communities. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pages 618–623. IEEE, 2015.

[16] Diego Cattaruzza, Nabil Absi, Dominique Feillet, and Jesús González-Feliu. Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 6(1):51–79, 2017.

[17] Kai-Wei Chang, Wen tau Yih, Bishan Yang, and Christopher Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1579, 2014.

[18] Sanjay Chawla, Yu Zheng, and Jiafeng Hu. Inferring the root cause in road traffic anomalies. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 141–150. IEEE, 2012.

[19] Xuxu Chen, Yu Zheng, Yubiao Chen, Qiwei Jin, Weiwei Sun, Eric Chang, and Wei-Ying Ma. Indoor air quality monitoring system for smart buildings. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 471–475. ACM, 2014.

[20] Leslie Cheung, Leana Golubchik, and Fei Sha. A study of web services performance prediction: A client's perspective. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, pages 75–84. IEEE, 2011.

[21] Paolo Crucitti, Vito Latora, and Sergio Porta. Centrality measures in spatial networks of urban streets. *Physical Review E*, 73(3):35C39, 2006.

[22] Neema Davis, Gaurav Raina, and Krishna Jagannathan. A multi-level clustering approach for forecasting taxi travel demand. pages 223–228, 2016.

[23] Guiguang Ding, Yuchen Guo, and Jile Zhou. Collective matrix factorization hashing for multimodal data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2075–2082, 2014.

[24] Pooja P Dubey and Prashant Borkar. Review on techniques for traffic jam detection and congestion avoidance. In *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*, pages 434–440. IEEE, 2015.

[25] Jan Fabian Ehmke and Ann Melissa Campbell. Customer acceptance mechanisms for home deliveries in metropolitan areas. *European Journal of Operational Research*, 233(1):193–207, 2014.

[26] Jan Fabian Ehmke, Stephan Meisel, and Dirk Christian Mattfeld. Floating car based travel times for city logistics. *Transportation research part C: emerging technologies*, 21(1):338–352, 2012.

[27] Beyza Ermis, Evrim Acar, and A. Taylan Cemgil. Link prediction in heterogeneous data via generalized coupled tensor factorization. *Data Mining Knowledge Discovery*, 29:203–236, 2015.

[28] David A Freedman. *Statistical models: theory and practice*. cambridge university press, 2009.

[29] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[30] Yanjie Fu, Hui Xiong, Yong Ge, Zijun Yao, Yu Zheng, and Zhi-Hua Zhou. Exploiting geographic dependencies for real estate appraisal: a mutual perspective of ranking and clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1047–1056. ACM, 2014.

[31] Byron J Gajewski and Laurence R Rilett. Estimating link travel time correlation: an application of bayesian smoothing splines. *Journal of Transportation and Statistics*, 7(2/3):53–70, 2004.

[32] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *WEA*, pages 319–333, 2008.

[33] Ravina Gelda, Krishna Jagannathan, and Gaurav Raina. Taxi dispatches using supply forecasting: A time-series based approach. In *IEEE 14th International Conference on Smart City*, 2016.

[34] Jeffrey Goderie, Brynjolfur Mar Georgsson, Bastiaan van Graafeiland, and Alberto Bacchelli. Eta: Estimated time of answer predicting response time in stack overflow. In *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*, pages 414–417. IEEE, 2015.

[35] Patrick-Oliver Groß, Michael Geisinger, Jan Fabian Ehmke, and Dirk Christian Mattfeld. Interval travel times for more reliable routing in city logistics. *Transportation Research Procedia*, 12:239–251, 2016.

[36] Richard A Harshman. Foundations of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis. 1970.

[37] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[38] Chenping Hou, Feiping Nie, Dongyun Yi, and Yi Wu. Feature selection via joint embedding learning and sparse regression. In *International Joint Conference on Artificial Intelligence*, volume 22, page 1324, 2011.

[39] Kent Hymel. Does traffic congestion reduce employment growth? *Journal of Urban Economics*, 65(2):127–135, 2009.

[40] Erik Jenelius and Haris N Koutsopoulos. Travel time estimation for urban road networks using low frequency probe vehicle data. *Transportation Research Part B: Methodological*, 53:64–81, 2013.

[41] Ms Avani Joshi and Dhirendra Mishra. Review of traffic density analysis techniques. *Image*, 4(7), 2015.

[42] Carl T Kelley. *Iterative methods for optimization*, volume 18. Siam, 1999.

[43] Bu Sung Kim, Heera Kim, Jaedong Lee, and Jee-Hyong Lee. Improving a recommender system by collective matrix factorization with tag information. In *Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on*, pages 980–984. IEEE, 2014.

[44] David G Kleinbaum, Lawrence L Kupper, Azhar Nizam, and Eli S Rosenberg. *Applied regression analysis and other multivariable methods*. Cengage Learning, 2013.

[45] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[46] Jake Kononov, Barbara Bailey, and Bryan Allery. Relationships between safety and both congestion and number of lanes on urban freeways. *Transportation Research Record: Journal of the Transportation Research Board*, (2083):26–39, 2008.

[47] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[48] John D. Lees-Miller. Minimising average passenger waiting time in personal rapid transit systems. *Annals of Operations Research*, 236(2):405–424, 2016.

[49] Chen Liang, Sharath Hiremagalore, Angelos Stavrou, and Huzefa Rangwala. Predicting network response times using social information. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 527–531. IEEE, 2011.

[50] Hong-En Lin, Rocco Zito, Michael Taylor, et al. A review of travel-time prediction in transport and logistics. In *Proceedings of the Eastern Asia Society for transportation studies*, volume 5, pages 1433–1448, 2005.

[51] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013.

[52] Ye Liu, Yu Zheng, Yuxuan Liang, Shuming Liu, and David S Rosenblum. Urban water quality prediction based on multi-task multi-view learning. 2016.

[53] JianCheng Long, ZiYou Gao, HuaLing Ren, and AiPing Lian. Urban traffic congestion propagation and bottleneck identification. *Science in China Series F: Information Sciences*, 51(7):948–964, 2008.

[54] Jalal Mahmud, Jilin Chen, and Jeffrey Nichols. When will you answer this? estimating response time in twitter. In *ICWSM*, 2013.

[55] Fei Miao, Shuo Han, Shan Lin, John A. Stankovic, Desheng Zhang, Sirajum Munir, Hua Huang, Tian He, and George J. Pappas. Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach. *IEEE Transactions on Automation Science and Engineering*, 13(2):463–478, 2016.

[56] Fei Miao, Shuo Han, Shan Lin, Qian Wang, John Stankovic, Abdeltawab Hendawi, Desheng Zhang, Tian He, and George J. Pappas. Data-driven robust taxi dispatch under demand uncertainties. *IEEE Transactions on Control Systems Technology*, pages 1–17, 2017.

[57] Wanli Min and Laura Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606–616, 2011.

[58] Takashi Nagatani. Propagation of jams in congested traffic flow. *Journal of the Physical Society of Japan*, 65(7):2333–2336, 1996.

[59] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[60] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[61] Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. Efficient and robust feature selection via joint ?, 1-norms minimization. In *Advances in neural information processing systems.*, pages 1813–1821, 2010.

[62] Lalit Sivanandan Nookala. *Weather impact on traffic conditions and travel time prediction*. PhD thesis, University of Minnesota Duluth, 2006.

[63] Bei Pan, Yu Zheng, David Wilkie, and Cyrus Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 344–353. ACM, 2013.

[64] Evangelos E. Papalexakis, Christos Faloutsos, Tom M. Mitchell, Partha Pratim Talukdar, Nicholas D. Sidiropoulos, and Brian Murphy. Turbo-smt: Accelerating coupled sparse matrix-tensor factorizations by 200x. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 118–126, 2014.

[65] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology*, 8(2), 2017.

[66] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[67] Piotr Rachtan, He Huang, and Song Gao. Spatio-temporal link speed correlations: An empirical study. *Transportation Research Record*, 2390:34–43, 2013.

[68] Amit Rechavi and Sheizaf Rafaeli. Not all is gold that glitters: Response time & satisfaction rates in yahoo! answers. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 904–909. IEEE, 2011.

[69] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM, 2010.

[70] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 1177–1178, 2010.

[71] Kiam Tian Seow, Nam Hai Dang, and Der-Horng Lee. A collaborative multiagent taxi-dispatch system. *IEEE Transactions on Automation Science and Engineering*, 7(3):607–616, 2010.

[72] Jingbo Shang, Yu Zheng, Wenzhu Tong, Eric Chang, and Yong Yu. Inferring gas consumption and pollution emission of vehicles throughout a city. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1027–1036. ACM, 2014.

[73] Jingbo Shang, Yu Zheng, Wenzhu Tong, Eric Chang, and Yong Yu. Inferring gas consumption and pollution emission of vehicles throughout a city. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1027–1036. ACM, 2014.

[74] Jiaxing Shen, Jiannong Cao, Xuefeng Liu, and Chisheng Zhang. Dmad: Data-driven measuring of wi-fi access point deployment in urban spaces. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(1):11, 2017.

[75] Axel Simroth and Henryk Zahle. Travel time prediction using floating car data applied to logistics planning. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):243–253, 2011.

[76] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[77] Peter Sprent and Nigel C Smeeton. *Applied nonparametric statistical methods*. CRC Press, 2016.

[78] Nathan Srebro, Tommi Jaakkola, et al. Weighted low-rank approximations. In *Icml*, volume 3, pages 720–727, 2003.

[79] Pang-Ning Tan and Vipin Kumar. Chapter 6. association analysis: Basic concepts and algorithms. *Introduction to Data Mining. Addison-Wesley. ISBN*, 321321367, 2005.

[80] Jun Tang, Haiqun Jin, Shoubiao Tan, and Dong Liang. Cross-domain action recognition via collective matrix factorization with graph laplacian regularization. *Image and Vision Computing*, 55:119–126, 2016.

[81] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[82] Yongxin Tong, Yuqiang Chen, Zimu Zhou, Lei Chen, Jie Wang, Qiang Yang, Jieping Ye, and Weifeng Lv. The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1653–1662. ACM, 2017.

[83] Erwin van der Laan, Jan van Dalen, Michael Rohrmoser, and Rob Simpson. Demand forecasting and order planning for humanitarian logistics: An empirical assessment. *Journal of Operations Management*, 45:114–122, 2016.

[84] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*, pages 935–942. ACM, 2007.

[85] Chao Wang, Mohammed A Quddus, and Stephen G Ison. Impact of traffic congestion on road accidents: a spatial analysis of the m25 motorway in england. *Accident Analysis & Prevention*, 41(4):798–808, 2009.

[86] Fei Wang, Ping Zhang, Buyue Qian, Xiang Wang, and Ian Davidson. Clinical risk prediction with multilinear sparse logistic regression. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 145–154. ACM, 2014.

[87] Hao Wang, Der-Horng Lee, and Ruey (Kelvin) Cheu. Pdptw based taxi dispatch modeling for booking service. In *In Proceedings of Fifth International Conference on Natural Computation*, pages 242–247, 2009.

[88] Senzhang Wang, Lifang He, Leon Stenneth, Philip S Yu, and Zhoujun Li. Citywide traffic congestion estimation with social media. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 34. ACM, 2015.

[89] Senzhang Wang, Xiaoming Zhang, Jianping Cao, Lifang He, Leon Stenneth, Philip S. Yu, Zhoujun Li, and Zhiqiu Huang. Computing urban traffic congestions by incorporating sparse gps probe data and social media data. *ACM Transactions on Information Systems*, 35(4):1–30, 2017.

[90] Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 25–34, 2014.

[91] Brent D Williams and Matthew A Waller. Creating order forecasts: point-of-sale or order history? *Journal of Business Logistics*, 31(2):231–251, 2010.

[92] Xiangye Xiao, Yu Zheng, Qiong Luo, and Xing Xie. Finding similar users using category-based location history. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 442–445. ACM, 2010.

[93] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S Yu. Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 741–749. ACM, 2017.

[94] Ling Yan, Wu-Jun Li, Gui-Rong Xue, and Dingyi Han. Coupled group lasso for web-scale ctr prediction in display advertising. In *International Conference on Machine Learning*, pages 802–810, 2014.

[95] Su Yang. On feature selection for traffic congestion prediction. *Transportation Research Part C: Emerging Technologies*, 26:160–169, 2013.

[96] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and pois. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 186–194. ACM, 2012.

[97] Jing Yuan, Yu Zheng, Chengyang Zhang, Xing Xie, and Guangzhong Sun. An interactive-voting based map matching algorithm. In *Mobile Data Management*, pages 43–52, 2010.

[98] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[99] Nicholas Jing Yuan, Fuzheng Zhang, Defu Lian, Kai Zheng, Siyu Yu, and Xing Xie. We know how you live: exploring the spectrum of urban lifestyles. In *Proceedings of the first ACM conference on Online social networks*, pages 3–14. ACM, 2013.

[100] Nicholas Jing Yuan, Yu Zheng, Xing Xie, Yingzi Wang, Kai Zheng, and Hui Xiong. Discovering urban functional zones using latent activity trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):712–725, 2015.

[101] Lingyu Zhang, Tao Hu, Yue Min, Guobin Wu, Junying Zhang, Pengcheng Feng, Pinghua Gong, and Jieping Ye. A taxi order dispatch model based on combinatorial optimization. In *In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2151–2159, 2017.

[102] XianXing Zhang, Yitong Zhou, Yiming Ma, Bee-Chung Chen, Liang Zhang, and Deepak Agarwal. Glmix: Generalized linear mixed models for large-scale response prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 363–372. ACM, 2016.

[103] Xiaoming Zhang, Cuixia Feng, and Guang Wang. Prediction of website response time based on support vector machine. In *Image and Signal Processing (CISP), 2014 7th International Congress on*, pages 912–917. IEEE, 2014.

[104] Liang Zhao, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. Hierarchical incomplete multi-source feature learning for spatiotemporal event forecasting. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2085–2094. ACM, 2016.

[105] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th international conference on World wide web*, pages 1029–1038. ACM, 2010.

[106] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Towards mobile intelligence: Learning from gps history data for collaborative recommendation. *Artificial Intelligence*, 184:17–37, 2012.

[107] Xiao-Xia Zheng, Jun-Feng Zhao, Zhi-Wen Cheng, and Bing Xie. Web service response time dynamic prediction approach. *Journal of Chinese Computer Systems*, 32(8):1570–1574, 2011.

[108] Yu Zheng. Methodologies for cross-domain data fusion: An overview. *IEEE transactions on big data*, 1(1):16–34, 2015.

[109] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.

[110] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444. ACM, 2013.

[111] Yu Zheng, Tong Liu, Yilun Wang, Yanmin Zhu, Yanchi Liu, and Eric Chang. Diagnosing new york city's noises with ubiquitous data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 715–725. ACM, 2014.

[112] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. Urban computing with taxicabs. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98. ACM, 2011.

[113] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2267–2276. ACM, 2015.

[114] Zibin Zheng, Hao Ma, Michael R Lyu, and Irwin King. Collaborative web service qos prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing*, 6(3):289–299, 2013.

[115] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.