11-2017

An integrated framework for modeling and predicting spatiotemporal phenomena in urban environments

Tuc Viet LE
Singapore Management University, trucviet.le.2012@phdis.smu.edu.sg

An Integrated Framework for Modeling and Predicting Spatiotemporal Phenomena in Urban Environments

TRUC VIET LE

SINGAPORE MANAGEMENT UNIVERSITY

2017

An Integrated Framework for Modeling and Predicting Spatiotemporal Phenomena in Urban Environments

by

Truc Viet Le

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Hoong Chuin Lau (Supervisor/Chair)
Professor of Information Systems
Singapore Management University

Robert J. Kauffman
Professor of Information Systems
Singapore Management University

Akshat Kumar
Assistant Professor of Information Systems
Singapore Management University

Siyuan Liu
Assistant Professor of Information Systems
Pennsylvania State University

Singapore Management University
2017

An Integrated Framework for Modeling and Predicting Spatiotemporal Phenomena in Urban Environments

Truc Viet Le

This thesis proposes a general solution framework that integrates methods in machine learning in creative ways to solve a diverse set of problems arising in urban environments. It particularly focuses on modeling spatiotemporal data for the purpose of predicting urban phenomena. Concretely, the framework is applied to solve three specific real-world problems: human mobility prediction, traffic speed prediction and incident prediction.

For human mobility prediction, I use visitor trajectories collected a large theme park in Singapore as a simplified microcosm of an urban area. A trajectory is an ordered sequence of attraction visits and corresponding timestamps produced by a visitor. This problem has two related subproblems: (spatial) bundle prediction and trajectory prediction. In the first problem, I apply the framework to predict a bundle (i.e., an unordered set) of attractions that a given visitor would visit given a time budget. In the second problem, the framework is applied to predict the visitor's actual trajectory given the current partial trajectory and time budget. In both problems, I apply the methods of trajectory clustering, hidden Markov model, revealed preference learning and (inverse) reinforcement learning in the integrated framework.

In traffic speed prediction, I wish to predict the spatiotemporal distribution of traffic speed over urban road networks. To this end, I propose local Gaussian processes which combine non-negative matrix (NMF) factorization with Gaussian process (GP) in order to enhance the efficiency of model training such that the solution could be deployed in real-time use cases. NMF is essentially a spatiotemporal clustering technique. The solution is extensively evaluated using real-world traffic data collected in two U.S. cities.

The incident prediction problem is about predicting the distribution of the number of crime incidents over urban areas in future time periods. Because of its similarity to the traffic prediction problem above, its solution greatly benefits from the GP model developed earlier. Particularly, the GP kernel function is inherited and extended to model the distribution of incidents in urban areas and their features. The proposed solution is evaluated using real-world incident data collected in a large Asian city.

Conceptually, this thesis uses machine learning techniques to solve three separate urban problems, whose contribution belongs to the large category of *urban computing*. At the core, its technical contribution lies in the unification of separate solutions tailored to those problems into an integrated framework that reasons with spatiotemporal data and, thus, is highly generalizable to other problems of similar nature.

Contents

List of Figures

List of Tables

Common Abbreviations

Acronym	**W**hat (it) **S**tands **F**or
AI	**A**rtificial **I**ntelligence
BIC	**B**ayesian **I**nformation **C**riterion
CV	**C**ross-**v**alidation
GBR	**G**radient **B**oosting **R**egression
GIS	**G**eographic **I**nformation **S**ystem
GP	**G**aussian **P**rocess
GPS	**G**lobal **P**ositioning **S**ystem
HMM	**H**idden **M**arkov **M**odel
IRL	**I**nverse **R**einforcement **L**earning
LM	**L**inear (regression) **M**odel
MAE	**M**ean **A**bsolute **E**rror
MAPE	**M**ean **A**bsolute **P**ercentage **E**rror
MDP	**M**arkov **D**ecision **P**rocess
NMF	**N**on-negative **M**atrix **F**actorization
POI	**P**oint **o**f **I**nterest
QoS	**Q**uality **o**f **S**ervice
RBF	**R**adial **B**asis **F**unction
RL	**R**einforcement **L**earning
RFID	**R**adio-**f**requency **I**dentification
RMSE	**R**oot-**m**ean-**s**quare **E**rror
RP	**R**evealed **P**reference
SVM	**S**upport **V**ector **M**achine
TMC	**T**raffic **M**essage **C**hannel
VR	**V**alue **R**atio

Acknowledgements

I would like to express my heartfelt gratitude to the following people, without whose help, guidance and support, this thesis wouldn't have been completed or come into being.

Prof. Hoong Chuin Lau, my thesis supervisor, for his dedicated guidance and incredible wisdom during my soul-searching Ph.D. journey. Special thanks for his graciousness and trust in me that had allowed me to carry out my work with independence and integrity. Under the Ph.D. program of the School of Information Systems, he placed me in a collaborative environment, both internally and internationally, and granted me the opportunities to pursue pragmatic research that makes use of real-world datasets.

Prof. Siyuan Liu of Penn State, my external committee member, for his early and lasting involvement in my Ph.D. journey, and whose role was essential to the formation of the core work of this thesis. Special thanks for being my advisor in CMU – the most memorable and productive period of my Ph.D. journey.

Dr. Richard Oentaryo, my co-author and friend, from whom I have learned and grown up so much professionally, and whose contribution was crucial to Chapter 6.

Dean Ramayya Krishnan of Heinz College (CMU) for his support and guidance during my stay at CMU while being generously hosted by Heinz College. Especially for the insights and advices that he gave amidst a very busy schedule.

Prof. Robert Kauffman, my co-supervisor, for his wisdom, intellect and valuable contributions to the writing of this thesis. Special thanks to his supervision of one of my early ERP projects using the Sentosa dataset that eventually transformed into Chapters 4–5.

All those involved in the law enforcement project (Chapter 7), including Na Fu, Jonathan, Jiali and Keng Hoe, for making the last-mile run of this thesis fun and thrilling.

Personal thanks to Julien Le Flohic, who inspired me to take up this Ph.D. challenge.

Last but definitely not least, I am forever grateful to my parents for bringing me into this world and being my constant source of support and love ever since. Thanks for being my strength throughout this incredible journey and beyond.

List of Publications

The following papers (listed in reverse chronological order) were published during the execution of this thesis.

Journal Papers

1. **Truc Viet Le**, Richard J. Oentaryo, Siyuan Liu & Hoong Chuin Lau (2017). Local Gaussian Processes for Efficient Fine-grained Traffic Speed Prediction. *IEEE Transactions on Big Data* (TBD), *3*(2), 194–207. [Chapter 6]

Conference Papers

1. Jonathan Chase, Jiali Du, Na Fu, **Truc Viet Le** & Hoong Chuin Lau. Law Enforcement Resource Optimization with Response Time Guarantees. *The 2017 IEEE Symposium Series on Computational Intelligence* (IEEE SSCI 2017). Honolulu, Hawaii, USA. [Chapter 7 & Appendix A]

2. **Truc Viet Le**, Siyuan Liu & Hoong Chuin Lau. A Reinforcement Learning Framework for Trajectory Prediction under Uncertainty and Budget Constraint. *The 22nd European Conference on Artificial Intelligence* (ECAI 2016). The Hague, Netherlands. [Chapter 5]

3. **Truc Viet Le**, Siyuan Liu, Hoong Chuin Lau & Ramayya Krishnan. Predicting Bundles of Spatial Locations from Learning Revealed Preference Data. *The 14th International Conference on Autonomous Agents and Multi-agent Systems* (AAMAS 2015). Istanbul, Turkey. [Chapter 4]

Papers Not Included in the Thesis

1. **Truc Viet Le**, Baoyang Song & Laura Wynter. Real-time Prediction of Length of Stay Using Passive Wi-Fi Sensing. *The 2017 IEEE International Conference on Communications* (ICC 2017). Paris, France.

2. **Truc Viet Le**, Siyuan Liu, Hoong Chuin Lau & Ramayya Krishnan. A Quantitative Analysis of Decision Process in Social Groups Using Human Trajectories: (Extended Abstract). *The 13th International Conference on Autonomous Agents and Multi-agent Systems* (AAMAS 2014). Paris, France.

3. **Truc Viet Le** & Minh Thap Nguyen. An Empirical Analysis of a Network of Expertise. *The 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (ASONAM 2013). Niagara Falls, Canada.

Chapter 1

Introduction

1.1 Big Data & the City

Spatiotemporal data have become ubiquitous today with the explosive growth of sensor networks, "smart" connectivity and mobility technologies, the Internet of Things (IoT) and autonomous vehicles [5, 22]. The pace of spatiotemporal data generation has been advancing at an ever increasing speed and finer-grained scale. That has enabled big data captured in densely populated urban environments to provide multi-scaled perspectives at the complex behaviors of urban systems. Indeed, the voluminous and feature-rich wealth of such spatiotemporal data can be turned into valuable insights that can be used to make cities more efficient, safer and improve the quality of life of urban dwellers. This is a significant utility of big data for social good as it has been forecast by the United Nations that 66% of the world's population will be living in cities by 2050 [41].

Rapid progress of urbanization around the world has led to the emergence of megacities and engendered significant challenges in urban environments, in which big data carries the promising solutions [22, 112]. Consider the example depicted in Fig. 1.1, which visualizes a massive amount of GPS trajectories (approximately 180,000 trips) produced by over 1,000 taxicabs in the city of Shenzhen, China in September, 2009. A trajectory is an ordered sequence of spatial locations (longitude and latitude) and timestamps sampled by a GPS tracker every few seconds. A taxicab reports its trajectory in real-time to a central server for fleet management purposes as long as the tracker is turned on. Taxi trajectory is a typical example of spatiotemporal data collected in an urban

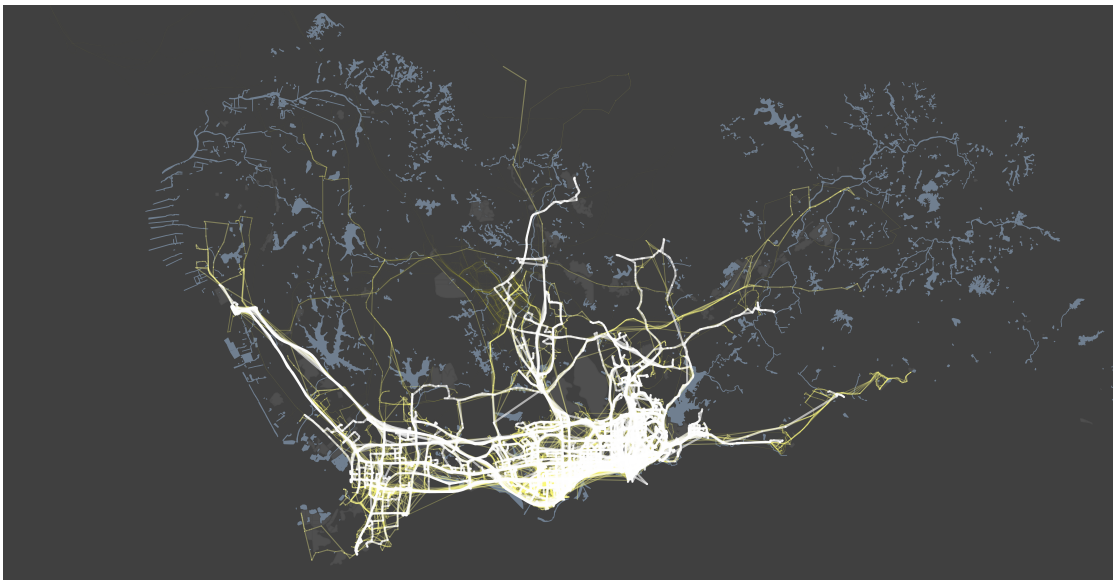

FIGURE 1.1: Visualization of almost 180,000 taxi trip trajectories in Shenzhen, China for the whole month of September, 2009. Each trajectory is a sequence of fine-grained GPS samples of locations and timestamps from the origin to the destination of the trip. Brighter areas correspond to higher and denser mobility demands in the road network.

environment. It tells a rich story about the urban residents' daily mobility patterns such as where the crowds are at what time. It informs the city planner how to design a better public transportation system. It gives the real-time traffic flow information that can be used for routing and avoiding congestions. It also records the real-world behaviors of taxi divers' cruising, picking up and delivering strategies that can be used to effectively train, e.g., an autonomous driverless car.

The overarching theme in this thesis is to **use spatiotemporal data to make cities smarter and safer**. In particular, I propose three specific prediction tasks based on the provided real-world data and show how to solve them: human mobility prediction, traffic speed prediction and crime incident prediction. These tasks are ultimately a means to an end, which could be urban crowd management, traffic congestion management, or effective deployment of law enforcement resources. However, this thesis does not have the ambition to provide "full-stack" solutions to those big problems due to its limited scope and their sheer complexities, where each would deserve a thesis of its own. On the other hand, by tackling those challenges at the core, a common pattern of problem solving emerges that can be synthesized into an integrated solution framework that could be useful for other problems of similar nature. This is the purpose of the thesis.

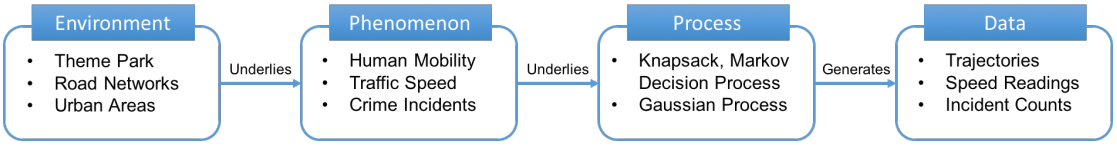

FIGURE 1.2: Relationship between the important concepts used in this thesis. Examples (used in the thesis) of each of the concepts are also illustrated.

As useful as they are, big spatiotemporal data pose two particular **technical challenges**. The first is the violation of the basic assumption of independent and identically distributed (i.i.d.) of observations due to the nature of space-time processes (i.e., observations that are "nearer" to each other should be more similar). The second is the scalability issue of big data in general. This thesis attempts to address those two challenges via the applications of established machine learning methods in creative ways. Briefly speaking, the first problem can be tackled using kernel methods, and the complexity of the second one can be significantly reduced via spatiotemporal clustering. Moreover, spatiotemporal data occurred in urban environments require an additional layer of modeling that captures the essence of the built environment underlying it. This is also adequately addressed in the thesis, e.g., via the design of kernel functions.

Concretely, this thesis develops a general machine learning framework to solve spatiotemporal problems (a.k.a "phenomena"[1]) occurred in urban environments. An **urban environment** is one typically characterized by high population density with complex human mobility patterns and advanced infrastructures (e.g., multimodal transportation networks) [112]. A spatiotemporal phenomenon is one that underlies a spacetime process, which in turn generates spatiotemporal data. A spacetime process is a stochastic process indexed by spatial locations and temporal labels. Thus, **spatiotemporal data** are multidimensional (and often multivariate) data that encode both the spatial and temporal dimension of the underlying phenomenon [50]. Examples of spacetime processes include a rational agent's decision-making process that generates a trajectory (Chapters 4 and 5 explore this), or Gaussian processes that generate observed traffic speeds and distribution of crime incidents in a big city (Chapters 6 and 7 have more on this). No matter what process it is, the goal of this thesis is unify solutions tailored to those problems into a common integrated framework. Fig. 1.2 illustrates the relationship between important concepts used in this thesis.

[1]To be used interchangeably throughout the thesis.

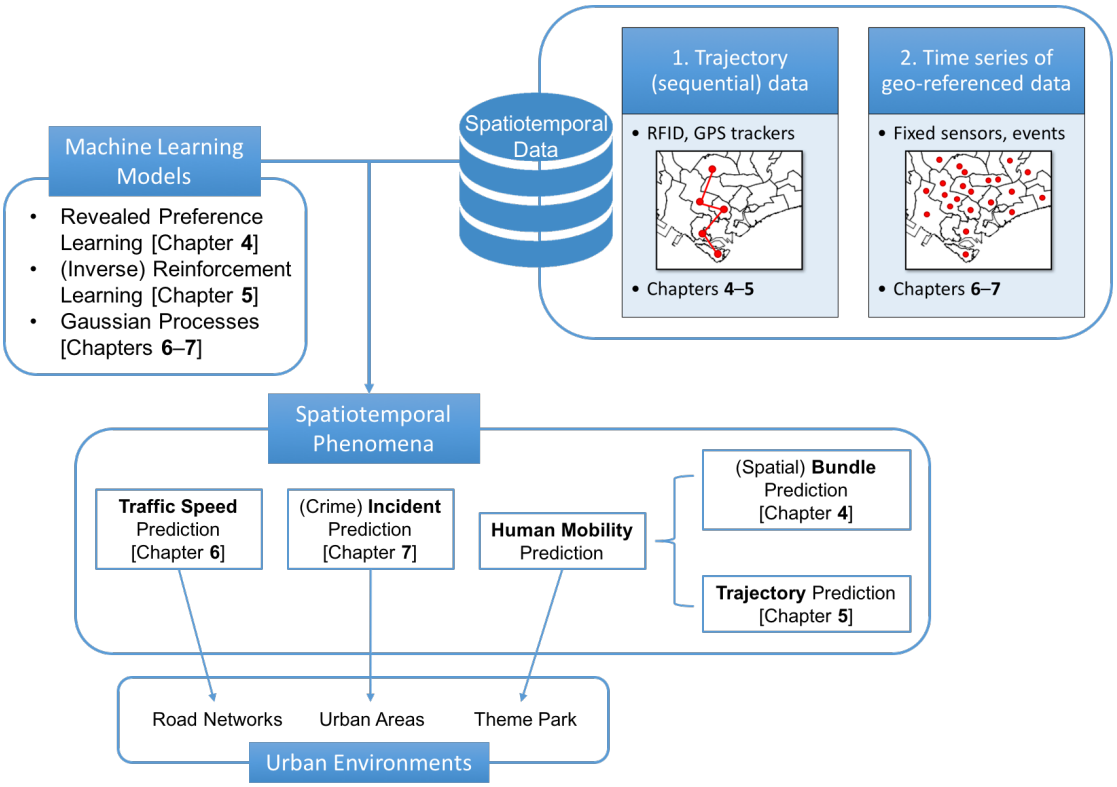

FIGURE 1.3: Summary of the machine learning models, spatiotemporal data and problems in urban environments studied in this thesis and their relationships.

Fig. 1.3 summarizes the machine learning models, spatiotemporal data and problems in urban environments studied in this thesis and their relationships. In essence, it is a combination of machine learning models and spatiotemporal data that solves a diverse set of problems in urban environments. Synthesized from those methods and data, a common solution framework can be integrated that "abstracts away"[2] the peculiar features of each of the individual problems. I call this an **integrated framework** because it provides a high-level abstraction of the problem solving process that can be generalized and extended to solve other problems of similar nature. In Fig. 1.3, even though the data and their corresponding problems are intrinsically tied together, the separation of data from problems gives rise to the synthesis and abstraction of processes that make up the integrated framework as will be elaborated in Chapter 3. In the next sections, I discuss in detail the components of Fig. 1.3: machine learning models, spatiotemporal data and problems in urban environments.

[2]In common computer science parlance, that means intentionally obscuring the details of how something works internally in order to simplify it conceptually.

1.2 Machine Learning Models

Tom Mitchell gave a classic definition of machine learning that says: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [71]. Here, the experience E is the provided dataset and the task T is the prediction of the instances of the spatiotemporal phenomenon under study. The performance measure P varies depends on what exactly is being predicted.

This thesis particularly explores machine learning models in the context of observational spatiotemporal datasets. Unlike classical statistics, with spatiotemporal data, the i.i.d. assumption is immediately violated. This is because the fundamental nature of spatiotemporal data is that observations at nearby locations in space and time are similar. This gives rise to the applications of kernel methods[3] such as **Gaussian processes** (GPs) (Chapters 6–7) that encapsulates such nearness without compromising on robustness. Furthermore, GP falls under the class of Bayesian methods as it allows for inference (i.e., prediction) as new evidence (i.e., experience) comes in without having to retrain the model. This proves essential for an application discussed in Chapter 7.

Spatiotemporal data can be modeled as a time series if they are produced by a sequence of actions (or "trajectory") taken by single actor (or agent). In this respect, predicting a sequence of spatiotemporal data boils down to modeling the sequential decisions being made by the actor. To this end, **reinforcement learning** [96] lends itself naturally to model such sequential decision-making (Chapter 5). Sequence alignment algorithms such as the edit distance [12] then becomes a viable performance measure. On the other hand, sequential decisions made under a certain time bound constraint also reveal the actor's preferences as the more preferred actions are more logically done first. Such big data of sequential decisions coupled with the classical economic theory of **revealed preference** [87] gives rise to new machine learning models that can learn and predict the agent's behaviors under different budget constraints (Chapter 4). Furthermore, the link between sequential decisions and preferences also gives rise to an application of **inverse reinforcement learning** [84] for trajectory prediction as discussed in Chapter 5. These are among the machine learning models employed in this thesis.

[3]The "kernel trick" largely solves the i.i.d. problem. See Sect. 2.4.2.

1.3 Spatiotemporal Data

Kisilevich et al. [50] gives a classification of spatiotemporal data along the two axes of space (static vs. dynamic) and time (snapshots vs. time series), which gives rise to four types of spatiotemporal data. Using this classification, the real-world data employed in this thesis fall under two types: static × time series and dynamic × time series.

As depicted in Fig. 1.3, the **first type** (dynamic × time series) represents the spatiotemporal trajectories typically produced by RFID-enabled devices or GPS trackers. This manifests in the sources of data provided in Chapters 4 and 5, where trajectories of visitors to a large theme park are collected via RFID-enabled devices. Whereas, the **second type** (static × time series) represents a time series of "readings" at static geo-referenced locations. This is typically produced by fixed-location sensors that monitor the phenomenon of interest at regular intervals, such as traffic speed sensors discussed in Chapter 6, or the incident reports (e.g., based on emergency calls) in Chapter 7. Specifically, the incidents (or events) can be viewed as a time series because if we bin them into a spatial grid and count the number of incidents per grid square over time, this is analogous to having a fixed-location "sensor" at each square that counts the number of incidents at each time step. Chapter 7 elaborates more on this data processing.

1.4 Spatiotemporal Phenomena

Motivated by some of the most prominent problems cities face today and the availability of rich real-world data, this thesis explores three big problems as depicted in Fig. 1.3. The first problem is **human mobility prediction**, which itself has two subproblems. In this problem, I seek to develop decision-making models for rational agents who visit a finite set of points of interest (POIs) in space given a limited time budget. The first subproblem (Chapter 4) seeks to predict an *unordered* set of POIs (called a "bundle") that the agent would visit. The second one (Chapter 5) predicts the precise *sequence* of visits (called a "trajectory"). Such ability to model and predict spatiotemporal behaviors can give rise to models of crowd distribution in urban areas, which could in turn assist a city planner in real-time crowd management. This thesis tackles the problem in a simplified version of an urban system, in which a real-world theme park and its visitors are used to represent a microcosm of a large urban area.

The second problem is **traffic speed prediction** (Chapter 6), which seeks to develop an accurate predictive model for how traffic speed evolves over space (i.e., the road network) and time in rhythm with the urban residents' daily activity patterns. To this end, I develop "smart" learning strategies that use matrix factorization to localize subsets of relevant training data in order to reduce the training set size. This effectively makes real-time learning and predictions of traffic speed feasible. Such ability can effectively assist traffic routing and crowd management, which in turn helps reduce urban congestions.

The third problem is **incident prediction** (Chapter 7), in which an accurate generative model for the distribution of crime incidents over space and time in a city is sought for. This has significant utility for the urban law enforcement agency to plan and deploy their resources efficiently and effectively in anticipation of emergent incidents. It also plays a crucial role in stress testing the agency's resource planning model in diverse scenarios in order to gauge the quality of their model. Due to its similarity to the traffic speed prediction in terms of data structure, the generative model in this problem can significantly benefit from the model developed in Chapter 6 as will be shown.

1.5 Urban Environments

An urban environment is a **built environment** that provides a setting for human activities in urban areas. Urban environments are typically highly developed, i.e., there is a high density of man-made structures such as houses, commercial buildings, roads, bridges, and railways and their interconnectedness [38]. In a broad sense, urban environments encompass the interactions of both the physically and socially built environments (e.g., demographics, cultures, value systems, laws and policies, etc.) under which urban activities take place [38]. In this thesis, the urban environments considered are restricted only to the physical environments underlying the spatiotemporal phenomena.

In this respect, the main challenge here is that the spatiotemporal data collected are not immediately usable. That is, they have to be mapped to the physical environment under which the phenomenon happens in order to be useful (for feature extraction and training the machine learning models). Such an environment represents the infrastructure underlying the phenomenon and requires an additional layer of data modeling. In the human mobility prediction problem, it is the "frame of reference" under which a visitor makes

their decisions (e.g., how to assign values to POIs and estimates their costs), which can be modeled using a hidden Markov model (HMM). In the traffic speed prediction problem, it is the road networks and their spatial features that underlie the traffic flows, which can be modeled using a directed graph coupled with kernel functions. Similar technique (i.e., kernel function) can be used to model the urban areas (with artificial boundaries and features) in the incident prediction problem.

1.6 Contributions & Organization of the Thesis

On a high level, the contribution of this thesis lies in the field of **urban computing**, which "embeds computational intelligence into the built environment via unobtrusive and ubiquitous urban sensing" [38]. Or more succinctly, Zheng et al. [112] puts it as "unlocking the power of big data collected in urban spaces to solve major issues cities face today". In particular, I make the following contributions in this thesis:

- I propose an integrated framework that combines machine learning methods to solve a diverse set of spatiotemporal problems in urban environments. This is achieved by synthesizing the common features of the individual problems and datasets and abstracting their peculiarities. The framework can be easily extended to solve other problems of similar nature in urban settings.
- The framework models the built environments underlying the phenomena and addresses the scalability issue of big data using spatiotemporal clustering.
- The framework is applied to solve three specific problems in urban environments: human mobility prediction, traffic speed prediction and incident prediction.
- All the problems discussed in this thesis make use of real-world data to vigorously evaluate the effectiveness of the proposed solution framework.
- Therefore, in this respect, the contribution of this thesis is to extend the Urban Data Analytics aspect, which comprises of data mining, machine learning and visualization, of the General Framework for Urban Computing Research proposed by Zheng et al. [112]. This is achieved by making it an realizable problem-solving process using machine learning. Sect. 2.1 explains more on this.

More concretely, the technical contributions of each chapter are given as follows:

- Chapters 4–5 use a theme park setting as a microcosm of an urban area. Based on visitors' trajectories, I apply the integrated framework to solve two related subproblems: (a) Predict a finite set of POIs that a visitor chooses to visit given a time budget, and (b) Predict a visitor's trajectory taking into account both the uncertainty of their preferences and time budget. In both problems, trajectory clustering is used to model the heterogeneity of the visitor population.

- In Chapter 4, trajectory data are used to extract the visitors' "revealed preferences" via a revealed preference learning model. A hidden Markov model (HMM) is then proposed to model the "frame of reference" under which a visitor makes their decisions. Based on which and the given time budget, the classic 0/1 knapsack problem is used to predict the optimal spatial bundle that the visitor chooses. This chapter is adopted from the following publication:

  – Truc Viet Le, Siyuan Liu, Hoong Chuin Lau & Ramayya Krishnan. Predicting Bundles of Spatial Locations from Learning Revealed Preference Data. *The 14th International Conference on Autonomous Agents and Multi-agent Systems* (AAMAS 2015). Istanbul, Turkey.

- In Chapter 5, HMM is again used to model the environment under which a visitor makes their sequential decisions. Decision models based on reinforcement learning (i.e., Markov decision processes) are proposed to model and predict the visitor's trajectory under budget constraint. For this purpose, inverse reinforcement learning is used to model the uncertainty of the visitor's preferences. Contents in this chapter are adopted from the following publication:

  – Truc Viet Le, Siyuan Liu & Hoong Chuin Lau. A Reinforcement Learning Framework for Trajectory Prediction under Uncertainty and Budget Constraint. *The 22nd European Conference on Artificial Intelligence* (ECAI 2016). The Hague, Netherlands.

- Chapter 6 solves a very common urban phenomenon: traffic speed prediction. In particular, I attempt predict the traffic speed distribution over road networks and time periods. To this end, I use spatiotemporal clustering coupled with Gaussian processes (GPs) to make efficient predictions of traffic speed. Specifically,

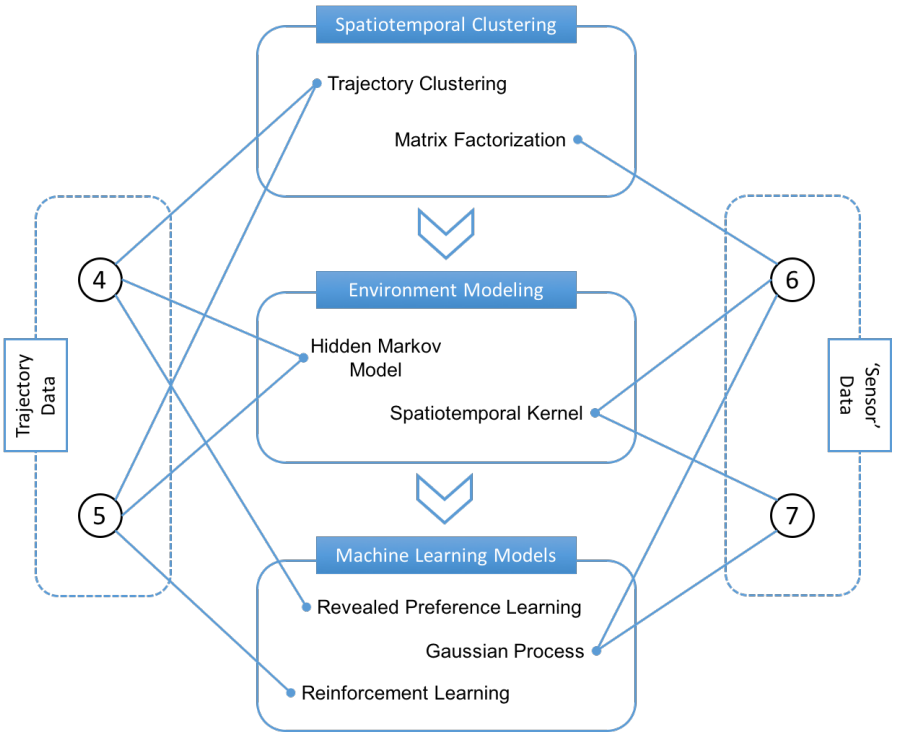

FIGURE 1.4: Overview of the methods used in each chapter. The methods are grouped by three identified themes: spatiotemporal clustering, environmental modeling and machine learning models. Arrow heads indicate the sequential flow of the problem solving process. Lines indicate "use" relationships. Each chapter is denoted by a corresponding circled number. The chapters are further grouped by their spatiotemporal data type.

matrix factorization is used to cluster and localize subsets of training data to efficiently learn "local" GPs and make real-time predictions. I also propose GP kernel functions that incorporates "side information" reflecting the features of the road networks for more accurate modeling and predictions. Contents in chapter are adopted from the following publication:

  - Truc Viet Le, Richard J. Oentaryo, Siyuan Liu & Hoong Chuin Lau (2017). Local Gaussian Processes for Efficient Fine-grained Traffic Speed Prediction. *IEEE Transactions on Big Data* (TBD), *3*(2), 194–207.

- Benefiting from the GP model and kernel function developed in the previous chapter, Chapter 7 attempts to predict the spatiotemporal distribution of crime incidents in urban areas. Such capabilities prove crucial to the design and implementation of a data-driven resource allocation model for law enforcement agencies. Parts of this chapter result in the following publication:

Fig. 1.4 summarizes the methods and techniques used in each chapter. It also illustrates the common themes among those methods. These themes abstract away the peculiarities of each method and its problem and give rise to the integrated framework. The figure also illustrates the flow of the problem solving process as the themes emerge, which should become clearer in Chapter 3.

The rest of the thesis is organized as follows. Chapter 2 reviews to literature related to the concepts, methods and problems used throughout this thesis. Chapter 3 introduces the integrated framework, the datasets and how the framework can be applied to each of the problems. Chapter 4 describes the bundle prediction problem. Chapter 5 elaborates on the trajectory prediction problem. Chapter 6 discusses on the third application: real-time traffic speed prediction. Chapter 7 illustrates the final application of the framework: predicting the spatiotemporal distribution of crime incidents in an urban area. Finally, Chapter 8 concludes the thesis with insights drawn and future directions.

Chapter 2

Literature Review

2.1 Overview

The high-level contribution of this thesis is in the area of *urban computing*. Urban computing is defined as the "process of acquisition, integration and analysis of big and heterogeneous data generated by diverse sources in urban spaces to tackle the major issues that cities face" [112]. In other words, the goal of urban computing is to help us understand the nature of urban phenomena and predict the future of cities [38, 112, 113]. An example of which is how autonomous vehicles would transform the future of urban mobility. This is briefly discussed in the future work in Chapter 8.

Following the general framework for urban computing research established by Zheng et al. [112], which is reproduced in Fig. 2.1 for clarity, my contribution in this thesis can be summarized as follows:

- In the **urban sensing** step, real-world spatiotemporal data are first obtained from private data partners (Sect. 3.2 describes this in detail).
- In the **data management** step, the provided data are fused with other sources of data (e.g., GIS shapefiles) to derive useful features for the problems.
- In the **data analytics** step, machine learning models are proposed coupled with clustering techniques to effectively and efficiently solve the problems, which are then formulated into a *framework* of its own. This can be seen as a more concrete

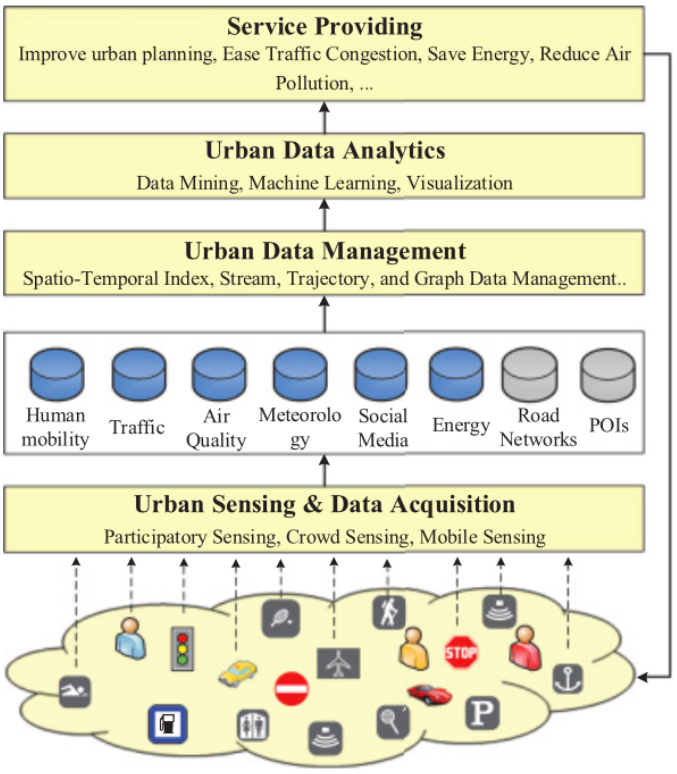

FIGURE 2.1: The general framework for urban computing research. Adopted from Zheng et al. [112].

contribution of the thesis to the urban computing literature as it extends the abstract component of "Urban Data Analytics" of the general framework into a generalizable and actionable problem-solving process using machine learning.

Finally, in the **service providing** step, the predicted values (in the data analytics step) are to be used for particular urban problems. This is not discussed in the thesis due to its limited scope. Appendix A illustrates such a typical application in law enforcement.

Thus, my urban data source is a cross-domain fusion of public sources (e.g., GIS shapefiles for road networks and boundaries of urban areas) and privately sourced data (e.g., trajectories, sensor readings). This can be considered as feature-level-based direct concatenation data fusion method according to Zheng [111] because it concatenates the selected features from different data sources and domain knowledge into a single dataset, from where the features are further scaled and regularized for model fitting.

In each of the following sections, I review the related work and literature to the thesis along the three axes: data, problem and method. I first briefly describe the concepts and

problems, and then review what has been done and highlight the differences between the contribution in this thesis and the previous work.

2.2 Spatiotemporal Data

2.2.1 Trajectory Data

A trajectory is an *ordered* sequence of spatiotemporal data produced by a moving object that has both a spatial and a temporal component vector [50]. The spatial vector contains the information of the locations where certain events of interest occur. The temporal vector contains the corresponding timestamps of such occurrences. In this thesis, the two component vectors are necessarily of the same length. I call the length of a trajectory its *sequence length*. Given a set of trajectories, one of the fundamental tasks is to identify *clusters* of similar trajectories [46, 50, 60, 63]. As it shall become clear, trajectory clustering plays a crucial role in identifying distinct mobility patterns, which in turns significantly reduces the complexity of the modeling problem.

Typical examples of trajectory data includes GPS trajectories produced by moving vehicles [16, 18, 62, 109] and sequences of human activities in a city throughout a day [46]. In this thesis, the trajectory data explored are more similar to the latter, i.e., sequences of attraction visits in a theme park by visitors during a defined time period.

2.2.2 Traffic Speed Data

Traffic speed data can be calibrated from a wide variety of sources: traffic cameras, GPS trajectories, speed sensors, etc. [61, 66, 72, 89, 104]. In this thesis, traffic speeds are obtained from the readings of fixed-location speed sensors placed along road segments. Thus, my data source is more closely related to the area of congestion and speed estimation. Congestion and speed estimation have been studied using various mathematical tools, ranging from flow patterns [61] to Markov chain forecasting [91], path oracles for spatial networks [88], and shortest path and distance queries on road networks [44, 100, 115]. Among those, there are generally two main categories of traffic data: (1) dynamic traffic measurements obtained from GPS trajectories or low-bandwidth cellular updates associated with individual "floating" vehicles [16, 44, 81, 100], and (2) static

traffic sensor readings associated with fixed locations (e.g., traffic cameras or speed sensor networks) [6, 14, 48]. In this respect, my data belongs to the second category.

2.2.3 Incident Data

An incident is a spatiotemporal data point in which a geo-referenced location and timestamp of a certain event of interest is recorded. Incident data thus belongs to the class of spatiotemporal events [50]. Both the spatial and temporal information associated with the event are static, since no movement or any other kind of evolution is recorded. Examples include earth tremors captured by sensors or geo-referenced records of a disease epidemic [50, 93]. Incident data and spatiotemporal clustering have a long history of applications in areas such as epidemiology and criminology, from the classic identification of the sources of the cholera outbreak in 1854 by John Snow [93], to the invention of spatial scan statistics [52] and the clustering of crime hotspots in urban areas [69].

In this thesis, incidents are crime reports obtained from emergency calls or organic police responses. Therefore, besides the spatiotemporal information, an incident here also records the textual description of the event and the police response to it.

2.3 Spatiotemporal Problems

2.3.1 Spatial Bundle Prediction

An optimal bundle problem is an instance of the classic 0/1 knapsack problem. In the knapsack problem, we are given n distinct and indivisible items and a knapsack of capacity $W > 0$. Each item i has a value $v_i > 0$ and carries a weight $w_i > 0$. Our task is to select the items to put into the knapsack such that the sum of the values of the selected items is maximized and the knapsack's capacity is not exceeded. Let $x_i \in \{0, 1\}$ be a binary decision variable, the problem can be written as:

$$\begin{aligned} &\max \sum_{i=1}^{n} x_i v_i \\ &\text{s.t.} \sum_{i=1}^{n} x_i w_i \leq W. \end{aligned} \tag{2.1}$$

In the spatial bundle prediction problem proposed in Chapter 4, the items are the POIs (i.e., locations) and the capacity constraint is the time budget B_i of an agent i. Each location j has a positive utility v_j to i, and i's task is to select a subset of locations to visit within its time budget B_i in order to maximize its sum of utilities. In the spatial setting, this problem is further complicated by the fact that the cost p_j of each location is *dynamic*, i.e., it changes depending on i's current location as p_j approximates the travel distance (hence, time cost). Chapter 4 proposes effective heuristics based on spatiotemporal clustering and hidden Markov models to solve this problem.

The spatial bundle prediction problem itself is novel to the best of my knowledge [57]. Besides that, another contribution of this thesis is the derivation of pairwise preferences between all pairs of locations for each agent. This in turn derives the location's utility to the agent. This is achieved via revealed preference learning [87, 106] from trajectories.

2.3.2 Trajectory Prediction

The problem of predicting the trajectory of a mobile agent is not entirely new. Krumm and Horvitz [51] initially propose a naive Bayes model called *Predestination* to predict the final destination of a driving trip given its partially observed GPS trajectory. In most recent the work, some form of Markov model is proposed to learn the trajectories and make inferences of future locations. Mathew et al. [70] use hidden Markov models (HMMs) to identify clusters of locations from raw GPS data, where each cluster is a POI and corresponds to a hidden state of the HMM. They make inferences of the next locations using the forward algorithm of HMMs. Trajectories are in turn clustered into groups of similar patterns to reduce variance and improve predictions. Gambs et al. [35] propose a mobility model call MMC (Mobility Markov Chain) to incorporate knowledge of the previous n visited locations and develop an inference algorithm based on n-th order Markov chains. Gao et al. [36] takes a Bayesian approach to the problem, but still within the Markovian framework. Sadilek and Krumm [86] predict an individual's location far into the future (in terms of months, years) using Fourier and principal component analysis (PCA), which is similar to that of Jiang et al. [46].

A common thread along these work is some variant of Markov models and some kind of spatiotemporal clustering (e.g., K-means, PCA, HMMs) to extract mobility patterns. I

adopt both themes in this thesis coupled with optimal decision models (e.g., knapsack and Markov decision process) for both spatial bundle and trajectory prediction.

2.3.3 Traffic Speed Prediction

The spatiotemporal correlation structure of traffic data can be exploited to predict the speed over unobserved road segments at any time using the observed data at the sensors' locations. Existing Bayesian filtering frameworks [19, 99] that utilize various handcrafted parametric models to predict traffic flows along highway stretches can only correlate with adjacent highway segments. Thus, their predictive performances could be compromised when the actual spatial correlation spans multiple segments. Moreover, their strong Markov assumption makes these models hard to generalize to arbitrary road network topology. Existing multivariate parametric models [49, 72] do not quantify uncertainty estimates of the predictions and impose rigid and unrealistic spatial locality assumptions.

In this thesis, I model traffic speed as a spatiotemporal Gaussian process (GP) that characterizes the spatiotemporal correlation structure of the phenomenon over a defined road network. Neumann et al. [74] maintained a mixture of two independent GPs for traffic speed prediction, such that the correlation structure of one GP utilizes road segment features and that of the other GP depends on manually specified relations. Xie et al. [104] used GPs to predict the time series of traffic volume over highways, and asserted GPs' superior performance over other parametric alternatives. Liu et al. [65] used GPs to model uncertain congestion environments for adaptive vehicle routing. More recently, Chen et al. [20] applied GPs for urban mobility demand sensing in a decentralized and distributed fashion. These approaches (except for [20]) do not scale well with big traffic data for real-time applications because of GP's $\mathcal{O}(n^3)$ computational complexity (in training set size). In contrast to the distributed GPs [20], the approach in this thesis is much simpler as it does not rely on complex decentralized mechanism.

2.3.4 Crime Incident Prediction

In this problem, given a location and a future time label (e.g., period, interval), I would like to know how many incidents of crime would occur there and then for each type[1]

[1]Typically, each type is modeled separately.

of incident. I would also like to know the distribution (i.e., mean and variance) of such prediction. This is a well-known problem in predictive criminology [80], where police officers proactively patrol crime "hotspots" in anticipation of incidents [69]. Refer to [80] for a comprehensive survey on predictive policing, including incident prediction.

In short, such predictions are highly sensitive to how the spatial and temporal labels are defined, i.e., how fine-grained is the urban area that I wish to predict. This depends on several factors, most of which are problem-specific, e.g., the boundaries of the areas of interest. Hence, they are left as input parameters in this thesis (Chapter 7). Earlier approaches to the problem are mostly game theoretic [108], particularly involving Stackelberg game, that model the interactions between the adversaries (i.e., criminals) and police officers. The game-theoretic approaches, however, mostly concern with generating patrol strategies while assuming a given, often simplistic, underlying generative model for the incidents. Examples include Zhang et al. [107] and Mukhopadhyay et al. [73], which model the interactions between criminals and officers using a dynamic Bayesian network (DBN) and use survival analysis to generate the incidents (more precisely the times to incidents), respectively. The latter represents the distribution of times to event as a function of arbitrary spatial and temporal features.

Machine learning methods have recently been applied to solve the problem. Most of these typically include some form of spatiotemporal clustering (e.g., spatial scan statistics) [69] and kernel methods (e.g., SVM) [80]. Recently, spatiotemporal Gaussian processes (GPs) with spectral mixture kernels have been applied to solve the problem that can make predictions "far into the future that goes well beyond what is currently believed to be possible" [30]. In this thesis, I also take the GP approach, but with a different (simpler) kernel function as there is no need to predict so "far into the future".

2.4 Machine Learning Methods

2.4.1 Spatiotemporal Clustering

Spatiotemporal clustering is the process of grouping objects based on both their spatial and temporal similarity. The technique has gain substantial popularity in recent years due to the emergence of various kinds of IoT technologies that record location, time

and environmental properties of various objects in real-time [50]. Indeed, through the availability of reliable and affordable sensors, we have witnessed an exponential growth in the amount of fine-grained geo-tagged data at small sampling intervals. Therefore, the challenge here is as big as the opportunity itself. It is the scalable clustering of big and fine-grained spatiotemporal data for effective knowledge discovery.

Various clustering techniques have been used to analyze a variety of traffic and human mobility phenomena. For example, Weijermars [101] applied a hierarchical clustering algorithm to identify typical urban traffic patterns that serve as basis for traffic forecasting. Jiang et al. [46] proposed a framework to cluster the human activity patterns in urban areas by combining principal component analysis and K-means clustering.

In this thesis, I adopt the method by Jiang et al. [46] with modifications for trajectory clustering (see Section 4.4) in Chapters 4–5. In Chapter 6, I employ non-negative matrix factorization (NMF) [23, 59], which assumes soft memberships to clusters, to cluster traffic phenomenon in urban environments. Indeed, Ding et al. [25] have shown that, by imposing certain constraints, NMF translates to "soft" K-means or spectral graph cuts. I also put NMF into a novel application to localize training data for local GPs, making the approach scalable to big spatiotemporal data. My approach thus offers a simpler and more generic alternative to the sparsification of GP kernels [15, 92], which tries to make the GP covariance matrices sparse in order to reduce complexities.

2.4.2 The Kernel Trick

A kernel function $k(\mathbf{x}, \mathbf{y}) : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ is a mathematical function that encapsulates the **similarity** between two vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^n$ in an n-dimensional feature space. Let ϕ be a *feature map* that transforms $\mathbf{x}$ and $\mathbf{y}$ into a higher dimensional feature space: $\phi : \mathbb{R}^n \mapsto \mathbb{R}^m$, where $m > n$. The kernel trick avoids the explicit mapping ϕ that is needed to learn a linear decision boundary in $\mathbb{R}^m$ in order to separate (or classify) $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$, assuming they are not linearly separable in $\mathbb{R}^n$. This is possible because kernel k effectively computes the dot products in $\mathbb{R}^m$ while remaining in $\mathbb{R}^n$:

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_m, \tag{2.2}$$

where $\langle \cdot, \cdot \rangle_m$ is an inner product in $\mathbb{R}^m$. This result directly follows from the Mercer's theorem [83], which is not presented here due to its mathematical complexity.

In other words, we can compute the dot products between $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^m$ using a function k that works exclusively in $\mathbb{R}^n$. This is a significant result, because, otherwise, ϕ would incur prohibitive computational cost of increasing the dimensionality from $\mathbb{R}^n$ to $\mathbb{R}^m$, if m grows very quickly with respect to n. Hence, the only effect on computational complexity is computing $k(\mathbf{x}, \mathbf{y})$. Depending on k, this can be minimal. Sect. 6.5.1 illustrates typical kernel functions used to model spacetime processes.

A particularly important implication of the kernel trick in this thesis is that while the i.i.d. assumption doesn't normally hold for spatiotemporal data, which invalidates the use of dot products in its original space $\mathbb{R}^n$. In a higher dimensional space $\mathbb{R}^m$, we can still safely use the dot product to compute the similarity between two data points in space and time without worrying about the i.i.d. assumption in $\mathbb{R}^n$.

2.4.3 Revealed Preference Learning

The seminal work of Paul Samuelson [87] has generated a voluminous body of work in the economics literature on revealed preference (RP) theory. See [97] for a comprehensive survey. A classic result is Afriat's theorem [3], which formulates a system of inequalities that has positive solution iff the demand data is rationalizable. A recent line of work emerging from the intersection of algorithmic game theory and machine learning has established some theoretical groundwork for the problem of learning utility functions from RP data [9, 54, 106]. Beigman and Vohra [9] use statistical learning analysis to address the problem of learning utility functions with the explicit goal of prediction. They show that the sample complexity (in the probably approximately correct sense) of learning a utility function from RP data is infinite, assuming monotonicity and concavity of utility functions. Lahaie [54] applies kernel methods to rationalize RP data assuming non-linear prices and incomplete price information, where prices of non-demanded bundles are unknown. The method reduces the problem to fitting utility function to observations in the transformed high-dimensional space using the "kernel trick".

Notably, Zadimoghaddam and Roth [106] recently propose a simple and efficient algorithm to learn utility functions from RP data for the class of linear and linearly separable

concave utility functions in polynomial sample complexity. Because of its simplicity and efficiency, I adopt one of the learning algorithms in Zadimoghaddam and Roth [106] in Chapter 4 for the bundle prediction problem (refer to Fig 4.2). My contribution here can be viewed as a extension of Zadimoghaddam and Roth [106] to the spatial setting. It is, however, not straightforward how the original algorithm can be adapted to solve the proposed problem, especially where costs are unobserved.

2.4.4 Reinforcement Learning

Modeling human sequential actions has been traditionally studied in the domain of human-computer interaction. For instance, mining sequential behaviors has been used to discover mobile users that share similar habits [68], or to imitate human behaviors in order to provide better automated care to the disabled and elderly [40]. In this respect, modeling sequential decisions as Markov processes is commonly used to simplify the representation of the user's knowledge [116]. A common shortcoming here is the lack of modeling of the users' decision processes in order to explain the discovered patterns.

Reinforcement learning (RL) is a powerful AI framework inspired by behaviorist psychology, which asserts that behaviors are learned from interacting with the environment in a trial-and-error fashion [96]. An agent learns optimal policies by observing the outcomes of its interactions with the environment and attaching rewards or punishments to different outcomes. Therefore, understanding human behaviors requires finding the reward function that motivates the observed actions. Inverse reinforcement learning (IRL), first proposed by Russell [85], provides an elegant framework to identify the reward function being optimized by the agents given their observed activities. Ng and Russell [75] propose the original algorithms to tackle the problem based on linear programming. Ever since, there has been a wealth of algorithms developed to solve IRL [114].

IRL has enjoyed diverse applications in automated control systems that try to imitate the behaviors of expert users (a.k.a. "learning from demonstrations") such as learning how to drive a car [2], controlling helicopters [1], and predicting mouse movements [116]. In this respect, my framework (proposed in Chapter 5) integrates IRL to model the stochasticity (i.e., distribution) of rewards in trajectory prediction. Based on Markov decision process (MDP), a specialized model of RL in which the environment is known, I propose sequential decision models to solve the problem with budget constraint.

2.4.5 Gaussian Process

Gaussian processes (GPs) have been consistently shown to be an effective tool for modeling various spatiotemporal phenomena in urban environments such as traffic flow [20, 44, 104] and event prediction [30]. As a result, they are also employed in this thesis for the corresponding problems: traffic speed prediction (Chapter 6) and incident prediction (Chapter 7). Because of their fully non-parametric Bayesian formulation, GPs advantageously allow for the explicit probabilistic interpretation of the model outputs and confidence interval estimations [20, 92, 104]. However, GPs also admit cubic time complexity in the size of the training set, which make them inefficient to model big spatiotemporal data [20, 65, 67]. This thesis attempts to overcome that challenge.

In Chapter 6, I propose a *local* GP approach to gain efficiency in model training. The idea of localizing the training data by clustering was first proposed by Snelson and Ghahramani [92], who developed a localization approach by dividing the training data into (disjoint) blocks via a simple farthest-point clustering. Nguyen et al. [76] proposed a local GP for online regression, where the training data are incrementally partitioned into local regions. For each local region, an individual local GP is trained, and prediction is performed by weighting the nearby local models. In Chapter 6, the localization is done on the response space (i.e., speed) instead of the feature space. Doing so enables us to train more accurate local GPs, each specializing in a specific traffic response regime.

Chapter 3

The Integrated Framework

3.1 Introduction

In this thesis, I propose multiple solutions to solve a diverse set of spatiotemporal problems in urban environments. However distinct and peculiar these problems are, their tailored solutions can be synthesized into common themes (as depicted in Fig. 1.4) and integrated into a general framework shown in Fig. 3.1. In this framework, the peculiar features of the individual problems have been abstracted and reduced into a generic process of problem solving that is highly extensible.

As shown in Fig. 3.1, we are given a dataset $\mathcal{D}$ that can be split into a training set $\mathcal{S}$ and a test set $\mathcal{T}$ that correspond to two phases of the framework: training (a.k.a. "learning") and test (a.k.a. "prediction"). $\mathcal{S}$ is used to train a machine learning model $\mathcal{M}$ for the phenomenon under study. $\mathcal{T}$ is used to evaluate $\mathcal{M}$ with unseen instances of the phenomenon. In the **training phase**, the following steps are performed:

Step 1 [Spatiotemporal Clustering] If necessary[1], $\mathcal{S}$ is split into K homogeneous subsets (or clusters) using its feature vector $\mathbf{f}$, then j denotes the index of each cluster.

Step 2 [Environment Modeling] For each j, the underlying environment $\mathbf{H}_j$ of the phenomenon is modeled using a mathematical model or data structure.

[1]This step is optional. As it turns out in Chapter 7, it is not necessarily performed.

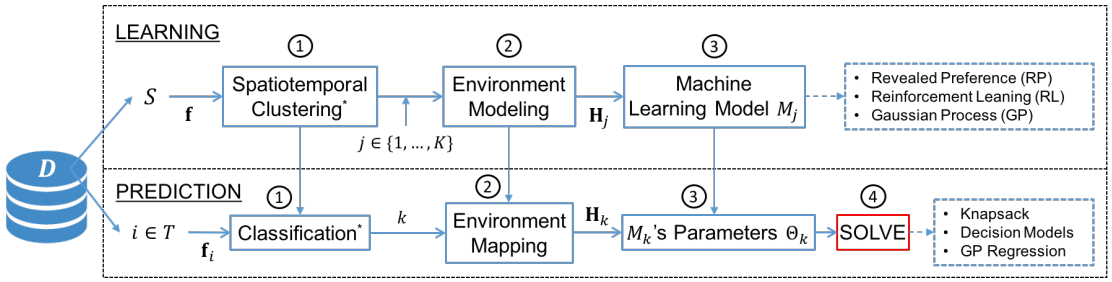

FIGURE 3.1: The integrated framework for modeling and prediction of spatiotemporal phenomena in urban environments. Dashed lines illustrate specific instantiations in this thesis. Asterisk (*) indicates an optional step. Circled numbers indicate steps.

Step 3 Finally, a machine learning model $\mathcal{M}_j$ is learned from the modeled examples in each subset j. Instances of $\mathcal{M}$ are: revealed preference learning (Chapter 4), reinforcement learning (Chapter 5), or Gaussian processes (Chapters 6–7).

After training, the **test phase** consists of the following steps:

Step 1 [Classification[2]] Given a test instance $i \in \mathcal{T}$ and its features $\mathbf{f}_i$, i is mapped into one of the K clusters learned from $\mathcal{S}$. Let k denote the mapped cluster index.

Step 2 [Environment Mapping] The observed environment of i is mapped into a set of environmental variables $\mathbf{H}_k$ that have been modeled in Environment Modeling.

Step 3 The corresponding set of parameters Θ_k of machine learning model $\mathcal{M}_k$ are then retrieved to generate an output response to "solve" the instance i.

Step 4 [SOLVE] Instance i is solved (i.e., predicted) either by using $\mathcal{M}_k$ directly (e.g., GP regression as in Chapters 6–7) or by a combination of $\mathcal{M}_k$ and optimization models such as knapsack (Chapter 4) or a sequential decision models (Chapter 5).

Finally, an appropriate performance measure P is proposed to evaluate $\mathcal{M}$ that quantifies the difference between each predicted value of instance i and its true (test) value, $\forall i$. Let x_i denote i's true value and $\hat{x}_i$ its predicted. Let $\Delta_P(x_i, \hat{x}_i)$ denote the difference between x_i and $\hat{x}_i$ under P. We say that $\mathcal{M}$ is a "good enough"[3] model under P if for all i, there exist arbitrarily small numbers $\epsilon, \delta > 0$ such that:

$$\Pr(\Delta_P(x_i, \hat{x}_i) \leq \epsilon) \geq 1 - \delta. \tag{3.1}$$

[2]This step is only performed if Spatiotemporal Clustering is performed in Training.
[3]Meaning $\Pr_{x_i \sim \mathcal{D}}(x_i \neq_P \hat{x}_i) \leq \epsilon$ [71].

TABLE 3.1: Summary of common notations used in this thesis.

Notation	Description
$\mathcal{D}, \mathcal{S}, \mathcal{T}$	Total dataset, training set and test set, respectively
i	Instance of the prediction problem
j	Spatiotemporal cluster index
K	Total number of clusters
k	The mapped cluster index (in Prediction)
$\mathbf{f}_i$	Feature vector of each instance i
$\mathbf{H}$	Set of parameters that model the underlying environment
$\mathcal{M}, \Theta$	Machine learning model and its set of parameters, respectively

Table 3.1 summarizes the common notations used in this thesis.

3.2 Datasets

I now describe the sources and background information of the real-world datasets used in this thesis, where the integrated framework is applied to solve their respective problems.

3.2.1 Human Mobility

In this problem, I make use of the proprietary dataset provided by the Sentosa Development Corp. (called "Sentosa" for short) on two of their attraction bundling schemes: Sentosa Choice Pass (called "Choice Pass" for short) and Sentosa Day Play Pass (called "Day Pass" for short). Sentosa itself develops and maintains a large theme park in the resort island of Sentosa, Singapore. Fig. 3.2 illustrates all of the attractions in Sentosa participating in these two bundling schemes. The attractions are clustered based on their features and spatial locations as designated by the theme park.

Under **Choice Pass**, each visitor can select any 4 attractions out of a defined set of 16 participating attractions and pay upfront a fixed bundle price (independent of their choice) before visiting. Visitors can then redeem their chosen attractions on a chosen day and during a specified period from 9 a.m. to 7 p.m. Each chosen attraction can only be visited once. Section 4.7.1 describes in detail the Choice Pass dataset.

Under **Day Pass**, each visitor pays upfront a fixed bundle price in order to redeem up to 14 participating attractions in Sentosa. (This set of attractions is specified by the theme park.) Visitors can then redeem the attractions during the specified 10-hour period from 9 a.m. to 7 p.m. on their chosen day. Each attraction can only be visited

FIGURE 3.2: Attractions in Sentosa participating in the Choice Pass and Day Pass bundling schemes. Clusters are designated based on their functionality and locations.

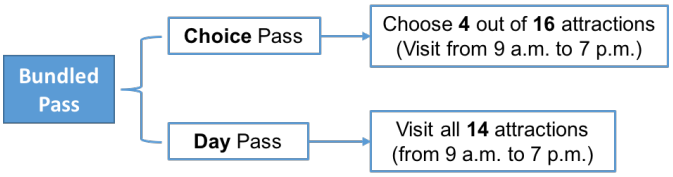

FIGURE 3.3: Summary of attraction bundling schemes by Sentosa studied in this thesis.

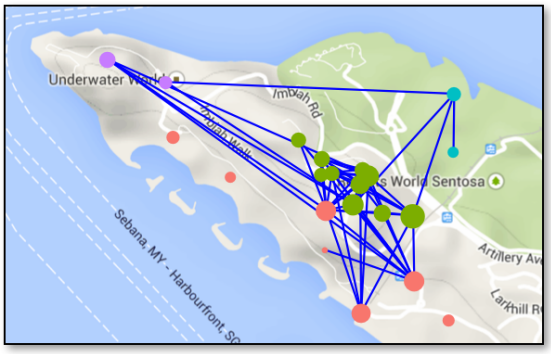

FIGURE 3.4: Popular pairwise transitions between Sentosa attractions. Popularity of each attraction is proportional to its node size. Colors illustrate attraction clusters arising from strongly associated pairwise transitions based on the provided trajectories.

once. Section 5.6.1 describes in detail the Day Pass dataset. Note that this bundle price is necessarily different from (i.e., more expensive than) that of the Choice Pass.

Figure 3.3 summarizes the two attraction bundling schemes studied in this thesis. For each dataset, the provided data is a set of visitor trajectories. Each **trajectory** is an *ordered* sequence of attraction visits (i.e., locations) and corresponding timestamps. Thus, each trajectory is a spatiotemporal sequence of visits. We are also provided with

other features such as sales (e.g., purchase method, point of sale, etc.) and demographic information of the visitors (adult/child ticket type, group information, etc.) that can be matched with the visitors' trajectories to form rich consumer behavioral datasets. The time period of both datasets is from January to April, 2014.

Given these datasets, our research questions are:

1. Can we model and predict a visitor's choice of attraction bundle (i.e., an *unordered* set) given their features and "known intention" (to be defined in Chapter 4)?

2. Can we model and predict a visitor's trajectory (i.e., an *ordered* sequence) given their partially observed trajectory and features?

Chapters 3 and 4 answer these two questions, respectively. A native method to both questions would be to "mine" the association rules from these trajectories and learn a generative model from those rules to make predictions. Fig. 3.4 illustrates such "strong" pairwise associations (i.e., ≥ 0.50) learned from the trajectories of the Day Pass dataset. We will see later that such a naive approach would fall short of our goals.

3.2.2 Traffic Speed

Our second data partner is Nokia HERE[4] (or "Here" for short). Here provides us fine-grained traffic speed measures along road segments in the two U.S. cities of Pittsburgh, Pennsylvania (P.A.) and Washington, D.C. recorded for several months in 2014. These speed measures are synthesized from a more complex data collection mechanism called TMC (Traffic Message Channel) that combines both privately crowd-sourced and public sources of traffic information. Such fine-grained traffic speed measures along road segments approximate a network of speed sensors for each city. Hence, these speed values themselves form a complex spatiotemporal phenomenon over the urban road network. Section 6.6.1 describes this in detail. Fig. 3.5 visualizes the daily average speed distribution along the road network of Pittsburgh for the whole month of August, 2014.

Fig. 3.5 shows certain dark patches in the city's road network, where, due to the lack of "sensors", no speed values were recorded during that time. For certain realistic

[4]Nokia sold Here to a consortium of German automotive companies (Audi, BMW and Daimler) in December, 2015.

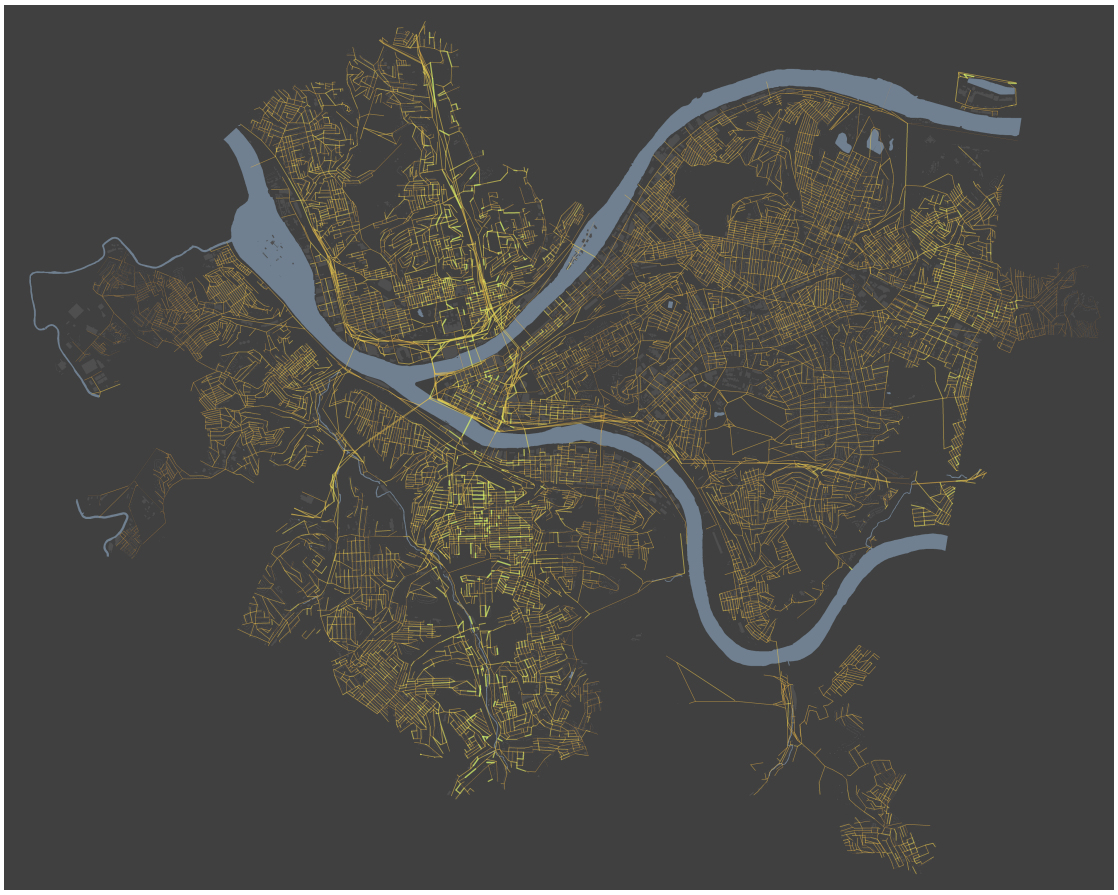

FIGURE 3.5: Visualization of the daily average traffic speed values along road segments in Pittsburgh, P.A. in August, 2014. Brighter colors illustrate higher speeds. Dark areas are either outside the city's boundaries or where no speed values were recorded.

applications, e.g., vehicle routing, it might be useful to know the speed distributions over those unobserved segments. Or given the daily average speed distribution for the month of August, we would like to know what the daily average speed distribution would be like in September. In other words, we ask the following questions:

1. At any moment, given the observed speeds over certain segments, what is the traffic speed distribution over other *unobserved* segments? I call this the **spatial inference** task.

2. At any location, given the observed speed over a particular segment (and others), what will be the traffic speed distribution here in a future time period? I call this the **temporal prediction** task.

I call both tasks the **spatiotemporal inference** of traffic speed. Chapter 5 provides an efficient solution framework to these questions based on Gaussian processes.

3.2.3 Crime Incidents

Our third dataset is crime incidents obtained from a national law enforcement agency in a large Asian city. The dataset spans over one year period in 2013–2014 and contains more than 200,000 reported incidents in total. Each incident is either recorded from an emergency call or from a police officer on patrol responding to it. For each incident, the data records the detailed information of the location (latitude, longitude and postal code), timestamp (date and time), type (e.g., traffic accident, quarrel, murder, etc.) and priority (i.e., urgency classification) as well as police dispatch and response information such as travel time and engagement time. In other words, the data tells us where, when, and what happened and how they were responded. Due to the highly confidential nature of the data, it cannot be described in detail in this thesis.

There are nine local centers of the agency (called "neighborhoods"), each contains 2–3 sectors as the base locations for the officers. Thus, a sector is part of a neighborhood, and each has a defined boundary. We are additionally provided with the neighborhood and sector boundaries via GIS shapefiles. Given this, our research tasks are:

1. To predict the distribution of the number of incidents in each sector and neighborhood in the future time periods.
2. To generate the distribution of the number of incidents if the given (sector/neighborhood) boundaries are changed.

As it will be clear in Chapter 7, such capabilities play an essential role in the design and implementation of an optimal resource allocation for the law enforcement agency. They are particularly useful for testing the robustness of the model in which the response time (to the incidents) plays the role of a key performance indicator.

3.3 Applications

In this section, I give a high-level description of how the framework depicted in Fig. 3.1 can be applied to solve each of the real-world problems studied in this thesis.

3.3.1 Bundle Prediction

In this problem, we are given some prior information about the agent's "initial intention" (e.g., first check-in time in the them park), we wish to predict the bundle of attractions (i.e., an unordered set of attractions) that the agent is going to visit during the day. Chapter 4 describes in detail the problem and its solution framework.

Learning. The following steps are performed:

1. Use trajectory clustering to divide the training set $\mathcal{S}$ into K distinct clusters, each represents an "agent type".
2. For each cluster, a hidden Markov model (HMM) is used to model the environment under which the agents make decisions (i.e., evaluate the time cost for each visit).
3. From the observed decisions made, a revealed preference model is learned for each agent type that reveals the agent's preference for every pair of attractions. These preferences are then stored in the "value ratio matrices" $\mathbf{R}$.

Prediction. Given a test agent i's initial intention and its feature vector $\mathbf{f}_i$, the following steps are performed:

1. Classify i into one of the K agent types using a logistic regression.
2. Map i into a particular environment HMM_k under which it makes decisions.
3. Heuristically derive the time costs to the (unvisited) attractions in i's consideration set and retrieve the pairwise preferences for the attractions (via matrix $\mathbf{R}_k$).
4. Given i's time budget B_i, its retrieved preferences and the cost of each attraction, the agent chooses the most optimal bundle by solving a knapsack problem.

3.3.2 Trajectory Prediction

In this problem, we are given the observed *partial* trajectory of an agent (e.g., the first sequence of n attraction visits), we wish to predict its remaining trajectory. Chapter 5 describes in detail the problem and its solution framework.

Learning. The following steps are performed:

1. Use trajectory clustering to divide the training set $\mathcal{S}$ into K distinct clusters, each represents an "agent type".

2. For each agent type, train a separate HMM to model the underlying environment they interact with. Each hidden state of the HMM is essentially a spatiotemporal cluster that models the agent's "state of mind" when it makes a decision.

3. Use Markov decision processes (MDPs) to model the sequential decisions of the agents. Under the MDP framework and given the observed sequential decisions, use inverse reinforcement learning (IRL) to learn the reward distribution of each state and each attraction visit for each agent type. Those parameters are stored in K separate reinforcement learning (RL) models.

Prediction. Given a test agent i's observed partial trajectory and its feature vector $\mathbf{f}_i$, the following steps are performed:

1. Classify i into one of the K agent types using a logistic regression.

2. Use the Viterbi algorithm to infer the (partial) sequence of hidden states of the HMM_k from i's observed partial trajectory.

3. Derive i's "expected reward level" (using the Bellman equation) that it is trying to achieve with the remaining attractions.

4. Two decision models are proposed to predict i's remaining trajectory: one based on (optimal) MDP and the other based on a suboptimal greedy heuristic. Both take into account the i's budget constraint and the uncertainties of the rewards.

3.3.3 Traffic Speed Prediction

In this problem, we are given a traffic speed query $\langle s, t \rangle$, and we wish to predict the traffic speed $f(r, t)$ along the road segment r at a future time t in a real-time fashion. Chapter 6 describes in detail the problem and its solution framework.

Learning. The following steps are performed:

1. Cluster the observed speeds along road segments and time periods into K^2 spatiotemporal clusters using matrix factorization (where K is an input parameter).

2. For each cluster j, derive its cluster "centroid" heuristically using the spatiotemporal features $\mathbf{f}_j$. The underlying environment here is the road network, which can be modeled using a directed graph $\mathcal{G}$ and a kernel function $k(\cdot)$ over $\mathcal{G}$.

3. Each cluster then serves as a "local" subset of training data for each *local* Gaussian process (GP) to be learned in real-time to make predictions.

Prediction. For each test query $\langle s, t \rangle$, the following steps are performed:

1. Use a simple heuristic to map the query to the closest cluster centroid.

2. A local subset of training data is then retrieved together with the spatial features of the clustered segments (called "side information") .

3. A local GP is trained in real-time (using the local training set) incorporating the side information via $k(\cdot)$ to make a prediction (i.e., performing a GP regression).

3.3.4 Incident Prediction

In this problem, we are given an arbitrary spatiotemporal query (x, y, t), where (x, y) represent the longitude and latitude coordinates, and t the time. We wish to predict the expected number of incidents that would occur at location (x, y) and time t. Chapter 7 describes in detail the problem and its solution framework. Note that in this application, Spatiotemporal Clustering and Classification steps are not necessary.

Learning. The following steps are performed:

1. Divide the continuous spatial and temporal dimension into K equal "bins" using spatial gridding and temporal intervals (whose parameters are given as inputs).

2. Bin the incidents into K bins and count the number of incidents within each bin. Each bin j has its centroid coordinates (x_j, y_j) and time interval t_j.

3. Using (x_j, y_j, t_j) as the spatiotemporal features of the bin, coupled with its derived side information $\mathbf{f}_j$, train a global GP to model the count variables over the bins.

Prediction. The following steps are performed:

1. Map the query (x, y, t) into a bin i.

2. Derive its spatiotemporal features (x_i, y_i, t_i) and side information $\mathbf{f}_i$.

3. Perform a GP regression to compute the distribution (mean and variance) of the number of incidents in i.

Chapter 4

Predicting Spatial Bundles from Revealed Preference Data

4.1 Introduction

I begin by first considering the problem of predicting a bundle of goods, where the goods are spatial locations that an agent wish to visit (a.k.a. "spatial bundle"), given knowledge of the costs of all goods considered and their budget constraint. This scenario typically arises in the travel industry where attractions in a certain geographical area can be packaged together by the developer and sold at a (discounted) bundled price. An example is CityPASS, where the company sells booklets (bundles) of attractions in 11 cities throughout North America. Bundles typically include transport passes and tickets to places of interest that can be redeemed for a specified duration of visit. Another is Eurail, a European-based company that markets bundled train passes to a defined set of European countries that share borders and for a specified period of travel. When prices of bundles differ depending on the combinations of the included goods and their quantities, the problem becomes that of classical *revealed preference* analysis.

Revealed preference (RP) is a consumer behavior theory pioneered by economist Paul Samuelson [87]. It is built on the premise that intrinsic preferences are unobserved; however, a consumer's preferences can be *revealed* through their observed purchasing behaviors. That is, it is possible to predict consumer behaviors on the basis of variable prices and income (budget constraint). A consumer with a given income will buy a

certain mixture of goods; but as their income changes, the mixture of goods will change accordingly. The theory assumes that a rational consumer has considered a set of all possible alternatives according to some well-defined utility function before making their decision. Thus, given a consumer chooses a option out of this set, this option must be the most preferred (i.e., the utility maximizer) that they can afford. The basic question of RP analysis is to recover a utility function that best explains (or *rationalizes*) the observed consumer behaviors [9, 54, 87, 106].

I consider the scenario where the developer allows buyers to "mix and match" a fixed quantity $Q > 0$ of items at preset price p and the chosen bundle can be consumed during a specified period B. For instance, a theme park developer would sell a bundle of attractions that visitors can choose from a fixed set (e.g., choosing $Q = 4$ out of 16 attractions) at price p; and once chosen, the attractions can be visited in any order during period B. If I consider all those who go for bundles at price p, then RP analysis is no longer feasible because of price uniformity. In other words, the cost information has become *latent* or *unobserved*. In order to apply RP analysis, I need to find a proxy to the costs that consumers take into consideration when making decisions. In my setting, cost information may be approximated by the physical distances of the visited locations revealed from an agent's trajectory in the absence of any other sources of information (e.g., means of transport, queue length at each location). This is because a rational agent would plan their visit such as to minimize the total distance traveled (or the time cost) over their chosen locations subject to budget constraint B. Finding such proxies is thus a challenge in RP analysis for spatial goods in the absence of complete information.

Given its spatial nature, the problem can benefit from the rich literature of the location prediction problem in spatiotemporal analysis. Motivated from the massive growth of spatiotemporal data generated by location-aware devices, the problem seeks to predict the next location(s) that an individual would travel to given their current and past trajectories. A trajectory is defined as an ordered sequence of timestamped locations. A common approach is to apply a wide variety of Markov models, most commonly Markov chains and hidden Markov models (HMMs), to model the sequential movements and make predictions. A common subtask is to cluster the locations using the hidden states of an HMM to discretized the space into finite points of interest (POIs) [35, 70]. Another is to cluster the trajectories into groups of similar mobility patterns and model each separately to reduce variance and improve predictions [46, 70].

In this chapter, I integrate techniques from spatiotemporal analysis to solve the proposed problem. In particular, I use trajectory clustering to divide the agents into groups of similar preferences for variance reduction. I then use HMMs to extract clusters of locations that are frequently visited together in order to establish the *reference points* from where the agents based their decisions on to approximate the costs. Finally, I leverage on a recent line of work emerging from the intersection of algorithmic game theory and statistical learning theory [9, 106], that has established the conditions and algorithms for efficiently learning the utility functions from RP data. I evaluate my proposed solution using real-world data collected from a theme park, which indeed outperforms the baselines, one of which was proposed for the next location prediction problem [70].

Applications of such predictions are plentiful and include predicting the aggregate demand in response to changes in costs of the goods (e.g., changing certain locations to further/nearer distances) or the set of all available goods itself (including/excluding some locations to/from the consumer's choice set), resource planning in anticipation of such changes in demand, and developing location-aware services or marketing.

4.2 Problem Statement

Consider a set $\mathcal{D}$ of agents and a finite set $\mathcal{G}$ ($|\mathcal{G}| = d$) of POIs of arbitrarily large capacity[1] each. Each agent $i \in \mathcal{D}$ faces a cost vector p_i, where p_{ij} is the cost of visiting location $j \in \mathcal{G}$ for i. Each i also has a personal value vector v_i over each $j \in \mathcal{G}$, where v_{ij} is the value of j for i. In other words, v_i reflects i's intrinsic preference over all $j \in \mathcal{G}$. Finally, agent i comes with a budget constraint B_i and i wishes to visit a subset $s_i \subseteq \mathcal{G}$ such that $|s_i| \leq Q$ for some $Q > 0$ and $\sum_{j \in s_i} p_{ij} \leq B_i$. Without loss of generality, suppose that i makes a vector of binary decisions $x_i \in \{0,1\}^d$ of which location j to include in the bundle s_i. The preference of i over all possible bundles is defined by a non-decreasing, non-negative concave utility function $u : \{0,1\}^d \rightarrow \mathbb{R}^+$. I assume throughout that i's utility function belongs to the class of linear utility functions, i.e., $u(x_i) = x_i \cdot v_i$. Thus, i chooses his most preferred bundle s_i^* (or equivalently x_i^*) by

[1]This is to make a simplified assumption that a visitor does not have to wait to get in an attraction that is full, which also simplifies the utility function to consider only the distance cost.

solving the classic 0/1 knapsack problem:

$$x_i^* = \arg\max_{x_i}\{u(x_i)|x_i \cdot \mathbf{1} \leq Q \wedge x_i \cdot p_i \leq B_i\}. \tag{4.1}$$

Following the conventions in machine learning, I derive a training set $\mathcal{S} = \{(p_i, B_i, x_i^*)\}_{i=1}^m$ drawn i.d.d. from $\mathcal{D}$ and a test set $\mathcal{T} = D - S$. Assuming linear utilities of the agents, I wish to learn the value vectors $\hat{v}_i$ from $\mathcal{S}$ in order to predict the chosen bundles in $\mathcal{T}$ with good enough accuracies. Let $x_i^*(p_i, B_i, v_i)$ be the chosen bundle and let $\hat{x}_i(p_i, B_i, \hat{v}_i)$ be the predicted one, my accuracies are good enough if for all i and for some $\delta > 0$:

$$\Pr(x_i^*(p_i, B_i, v_i) \neq \hat{x}_i(p_i, B_i, \hat{v}_i)) \leq 1 - \delta. \tag{4.2}$$

4.3 Solution Overview

Fig. 4.1 illustrates the overall solution framework to the above problem. For learning, I first split all training agents in S into K clusters using trajectory clustering. For each cluster Cl_j $(1 \leq j \leq K)$, I train a separate HMM_j that best describes the sequential movements of those in Cl_j. I then propose a heuristic to approximate the perceived costs faced by each agent called the "centroid heuristic". I derive a set of "centroids" C_j for each Cl_j using the hidden states of HMM_j such that each agent $i \in \text{Cl}_j$ can be mapped to each centroid (a.k.a. "reference point") $r_k \in C_j$ depending on their intention I_i (to be defined). Each r_k corresponds to a perceived cost vector p_k shared by all the agents having the same intention. I can then efficiently learn the value ratio matrix R_j given the chosen bundle x_i^* and cost vector p_k of all agents $i \in \text{Cl}_j$ using an RP learning algorithm, e.g., the one due to Zadimoghaddam and Roth [106].

To make predictions, for each agent $i \in \mathcal{T}$, I first predict which cluster Cl_k that i most likely belongs to – call this Cl_k^i. In this work, I don't address the problem of class prediction due to restricted scope. I suppose it is feasible, and most of the time it is through some established method such as logistic regression and decision tree. Given i's intention I_i, I map I_i to the nearest reference point r_j^i, from where I can derive i's cost vectors p_j^i. Let $\hat{v_k}$ be any row vector of R_k corresponding to Cl_k. Given i's budget B_i and learned value vector $\hat{v}_k$, I predict i's chosen bundle when facing p_j^i by solving (4.1).

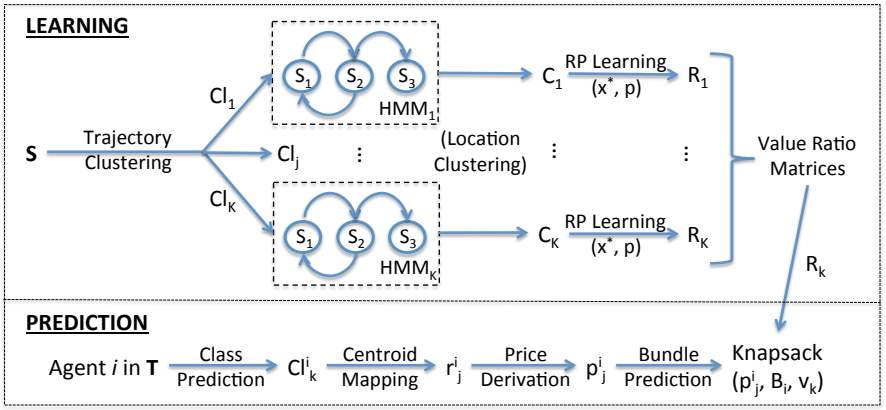

FIGURE 4.1: The learning and predictive framework for spatial bundle prediction.

TABLE 4.1: Summary of additional notations used in this chapter. Reintroduced notations override those introduced earlier.

Notation	Description
$\mathcal{G}$	Set of POIs, where $\lvert\mathcal{G}\rvert = d$
v_{ij}, p_{ij}	Value and cost of location $j \in \mathcal{G}$ for agent i
s_i, l_i	Trajectory and trajectory (sequence) length of agent i
B_i	Budget of agent i
Q	Maximum bundle size ($1 \leq l_i \leq Q$)
x_i	The selected bundle of agent i
R_k	Value ratio matrix of agent type k ($1 \leq k \leq K$)

Table 4.1 summarizes the additional notations used in this chapter. The following sections elaborate on the components of the proposed framework in Fig. 4.1.

4.4 Trajectory Clustering

One important challenge is that I cannot simply learn the preferences of each agent $i \in \mathcal{S}$ and predict for another $j \in \mathcal{T}$ because: (1) that is highly inefficient, and (2) it would most likely overfit the training data and lead to poor predictions. On the other hand, nor can I expect everyone to behave the same under the same prices and budget constraint as implied by RP theory because empirical data shows a great diversity of behaviors. I seek a solution in between where the agents can be divided into groups of similar behaviors such that I could learn the preferences from and predict the behaviors of those of the same group. This is the rationale for trajectory clustering.

For each agent $i \in \mathcal{S}$, let l_i be the sequence length of i, I denote the sequence of locations visited by i as $y^{(i)} = \{y_t^{(i)}\}_{t=1}^{l_i}$ and the sequence of timestamps for each $y_t^{(i)}$ as $\tau^{(i)} = \{\tau_t^{(i)}\}_{t=1}^{l_i}$. I define i's **trajectory** as $s^{(i)} = \{(y_t^{(i)}, \tau_t^{(i)})\}_{t=1}^{l_i}$. Hence, a trajectory is

a spatiotemporal sequence of spatial locations and their corresponding timestamps. A spatial location is a place in the physical world that can be located using its coordinates. A timestamp τ_t indicates when the agent visited location y_t, but does *not* necessarily indicate how long they had stayed there (i.e., the duration of visit).

Suppose there exist an upper bound B_U and a lower bound B_L on the timestamps of all the trajectories, then the duration between B_U and B_L can be discretized into a finite number of T segments $T = \lceil (B_U - B_L)/\Delta_\tau \rceil$, where Δ_τ is an arbitrary duration of each time segment. I can derive a categorical vector a_i of finite and uniform length T for each agent i from their original trajectory $s^{(i)}$. Each element $a_{it} \in a_i$ $(1 \leq t \leq T)$ indicates i's location at time t. If no location is recorded for i at t, then $a_{it} = 0$ by convention; otherwise, $a_{it} \in \{1, \ldots, |\mathcal{G}|\}$. I finally assume that i spends at least time Δ_τ and at most an integral multiple of Δ_τ at any location in its trajectory.

It is feasible to cluster the agents based on their similarity of behaviors by clustering the trajectories, or equivalently the vectors a_i for all $i \in \mathcal{S}$. To this end, there exist a wide variety of methods for sequence clustering. In this work, I make use of the well-known method of hierarchical clustering because of its simplicity and the ability to incorporate domain knowledge in selecting the number of clusters K. In particular, I use the agglomerative approach that clusters the trajectories recursively in the bottom-up fashion. I use the edit distance[2] to quantify the dissimilarity between any two vectors a_i and a_j with substitution cost being the physical distance between the pair of locations that differ in a_i and a_j and arbitrary insertion/deletion cost (because they are essentially two vectors of categorical variables of the same length). To select the number of clusters K, the hierarchy tree is "cut" at some height that would break up $\mathcal{S}$ into K clusters, which can be determined based on my domain knowledge.

There are many other more advanced methods for sequence and spatiotemporal clustering; I am using one of the simplest and most popular here because clustering is not my final goal, but a means to an end. An example of a more advanced method is due to Jiang et al. [46], in which trajectories are clustered by combining both PCA and

[2]Edit distance is a concept in computer science that quantifies how dissimilar two given strings (i.e., words) are to one another by counting the number of operations (i.e., edits) required to transform one string into another. Many algorithms have been proposed to compute the edit distance. In this particular application, I use the Levenshtein algorithm [12], which is based on dynamic programming and allows for the deletion, insertion and substitution of characters.

K-means clustering. Refer to Kisilevich et al. [50] and Fraley and Raftery [33] for comprehensive surveys on spatiotemporal and model-based clustering, respectively.

4.5 Revealed Preference Learning

In the traditional RP analysis, I am given a sequence of observations $\mathcal{D} = \{(p_i, B_i, x_i^*)\}_{i=1}^N$, the problem is to recover the utility function that best explains or *rationalizes* $\mathcal{D}$. Under the assumption of linear utility, I wish to recover the vector v_i for each agent $i \in \mathcal{D}$. Furthermore, not only do I wish to explain the observed data, I also wish to predict future chosen bundles of the agents given the recovered utility function. The latter goal is much broader and harder than the former, because being able to rationalize observations often does not generally indicate being able to predict unobserved outcomes [106].

Suppose I am able to categorize $\mathcal{D}$ into K clusters ($K \ll N$). For simplicity, I also call an agent belonging to cluster k ($1 \leq k \leq K$) an agent of type k. The problem can be solved by applying the All Pairs Comparison (APC) algorithm due to Zadimoghaddam and Roth [106], where for each agent type k, there is a value ratio[3] matrix R_k learned from $\mathcal{S}$ of dimension $d \times d$. The APC algorithm is a very simple and efficient algorithm to learn the value ratios v_i/v_j from RP data for all pairs of goods $i, j \in \mathcal{G}$. The main idea is to bound the pairs v_i/v_j such that if item i is *preferred to* item j in some chosen bundle x^*, then $v_i/p_i \geq v_j/p_j$, or equivalently $v_i/v_j \geq p_i/p_j$. Fig. 4.2 describes the APC algorithm, where δ is an input accuracy paramter.

Unlike an unordered bundle of goods, spatial locations have to be visited in a sequential order; thus, x^* has an intrinsic ordering nature. Given locations $i, j \in x^*$, I denote $x_i^* > x_j^*$ if i was visited before j by the considered agent, which also implies the agent's preference of i over j in x^*. Thus, v_i/v_j can be upper and lower bounded given the purchase decisions $x^{(k)}$ and price vectors $p^{(k)}$ of all agents of type k. Given a test agent of type k, I would choose any row i of R_k to obtain a value ratio vector $\hat{v} = v_i/v_j$ for all $1 \leq j \leq d$ (whose elements are arbitrarily in between the bounds) and predict an optimal bundle $\hat{x}(p, B, \hat{v})$ by solving (4.1) using the given price p and budget B.

[3]The term "value ratio" is used here to mean that each entry (i, j) of the matrix represents the ratio of the value of goods i over that of goods j from the perspective of agent type k. This, in turn, allows for the derivation of the preference relation for all pairs of goods.

AllPairsLearn(δ):
Training Phase:

1. **Let** E be a set of $m = O\left(\frac{n^2 \ln(n^2/\delta)}{\delta}\right)$ observations $(p, B, x^*(p, B, v))$.
2. **Initialize** bounds $(L_{i,j}, U_{i,j})$ for each $i \neq j \in [n]$. Initially $L_{i,j} = 0$ and $U_{i,j} = \infty$ for all i, j.
3. **For** each $(p, B, x^*) \in E$:
   (a) **For** each $i \neq j \in [n]$:
      i. **If** $x_i^* > x_j^*$, **Let** $L_{i,j} = \max(L_{i,j}, \frac{p_i}{p_j})$
      ii. **If** $x_j^* > x_i^*$, **Let** $U_{i,j} = \min(U_{i,j}, \frac{p_i}{p_j})$

Classification Phase:

1. On a new example (p, B) let $v' \in [0, 1]^n$ be any vector such that for all $i \neq j \in [n]$ $\frac{v'_i}{v'_j} \in [L_{i,j}, U_{i,j}]$. Predict bundle $x'(p, B, v')$ that results from maximizing v' with respect to prices p and budget constraint B.

FIGURE 4.2: The All Pairs Comparison (APC) algorithm adopted from Zadimoghaddam and Roth [106]. Notice that n (used originally in the paper) in the figure means d in this chapter (i.e., the number of distinct POIs) and $i, j \in [d]$ are simply the indices.

I use physical distance throughout to approximate the time cost of traveling from one location to another. Hence, my budget constraint B is defined as the total time cost required to go through the all the locations in the chosen bundle. Physical distances are different depending on from *where* they are measured, i.e., the **reference point**. One way to compute the total cost of a trajectory $s^{(i)}$ is to sum all the distances of the segments in $s^{(i)}$. This method does not scale because there are an exponential number of ways to choose Q locations from the set $\mathcal{G}$. On the other hand, suppose I know i's intention I_i of approximately where i would go, I would be able to map I_i to a particular reference point r_k in space from where I can approximate the total cost as the sum of distances from r_k to all the locations in $s^{(i)}$. An optimal reference point r_k for i is one that minimizes i's total distance derived from r_k assuming i's goal is to minimize the total cost. However, this is not feasible in the absence of complete information of $s^{(i)}$. I define i's **intention** I_i as any form of incomplete information about $s^{(i)}$ that I may have. In the following, section, I propose heuristics to derive r_k given I_i.

4.6 Heuristics for Cost Derivation

If I know i's first visited location, call it $y_1^{(i)}$, then I can take $y_1^{(i)}$ as the reference point for $s^{(i)}$. The rationale is that the first location in a sequence is often the one having the highest priority (i.e., the most preferred) and a rational agent would plan their itinerary in such a way to minimize their total time cost as seen from $y_1^{(i)}$. By this, I am making

other locations that are distant from $y_1^{(i)}$ costly and less likely to be included in the bundle. This heuristic aligns with the assumption that agents try to minimize the total distance due to budget constraint. I simply call this the **first-location heuristic**.

Let d_i be the vector of physical distances from r_i, where r_i is i's reference point, to all the locations in $\mathcal{G}$. I can easily derive p_i, the vector of time costs to travel from r_i to all the locations in $\mathcal{G}$ from d_i. For example, if the primary means of travel is on foot, then d_i can be converted to p_i using the average human walking speed of 5 km/h. Thus, for each location $j \in \mathcal{G}$, p_{ij} is the average time cost for i to go from r_i to j. Furthermore, suppose I know the upper bound on the duration of visit at each location $j \in \mathcal{G}$, call this b_j, then the proper price vector $\hat{p}_i$ seen by i is $\hat{p}_{ij} = p_{ij} + b_j$, which reflects the true time cost at j (i.e., the total time of traveling to j and the duration of visit at j).

Often, we may not know for certain what an agent i may want to include in their itinerary due to incomplete information. Instead, we may only know i's intention I_i of such. In such cases, I would want to divide my physical space (that covers all of $\mathcal{G}$) into non-overlapping sub-areas and map I_i to one of such sub-areas. For each sub-area, I would derive a reference point from where I can compute the price vector p_i. My rationale comes from the empirical observations that businesses of similar nature tend to cluster together geographically in an area in order to compete. Therefore, identifying such clusters of locations (or sub-areas) is the first step to identifying sensible reference points to infer costs in the absence of complete information.

To this end, I make use of HMM to derive clusters of locations. Locations within a cluster should be physically close to one another and tend to be visited together in short temporal sequence (i.e., without much delay). I use the hidden states of an HMM to identify those clusters such that each state corresponds to a cluster. I then derive the reference point of each cluster using its *centroid* (to be defined later). Given a centroid r_k, I use the nearest-neighbor method to assign locations to clusters: I assign location j to cluster k such that the physical distance from j to r_k is the nearest among all other centroids. For each agent i, given I_i, I map I_i to the nearest cluster centroid r_k and calculate p_i as before. I call this the **centroid heuristic**. The following subsections elaborate on the proposed method, beginning with the preliminaries of HMMs.

4.6.1 Hidden Markov Model (HMM)

An HMM describes the relationship between two stochastic processes: an observed process and an unobserved (or hidden) underlying process. The hidden process is assumed to follow a Markov chain, and the observations are considered conditionally independent given the sequence of hidden states. Let $\{Y_t\}_{t=1}^T$ and $\{X_t\}_{t=1}^T$ be the time series representing the observations and the corresponding hidden states of an HMM respectively. I denote $f(y_t|\Theta_{x_t}) = \Pr(Y_t = y_t; \Theta|X_t = x_t)$ the probability density function of observation y_t parameterized over vector Θ given hidden state x_t. An HMM with finite N hidden states is completely specified by:

1. The finite set of hidden states $S = \{S_1, S_2, \ldots, S_N\}$;
2. The state transition matrix $\mathbf{A} = \{a_{ij}\}$, where $a_{ij} = \Pr(X_t = S_j|X_{t-1} = S_i), 1 \leq i, j \leq N$;
3. The parameter vector Θ_i of the response (or emission) density function $f(y_t|\Theta_{x_t})$ for each S_i; and
4. The vector of initial (state) probabilities $\pi = \{\pi_i\}$, where $\pi_i = \Pr(X_1 = S_i)$ and $\sum_{i=1}^N \pi_i = 1$.

It is common to use the compact notation

$$\mathbf{\Lambda} = (\pi, \mathbf{A}, \{\Theta_i\}) \tag{4.3}$$

to represent the complete parameter set of an HMM. The problem of estimating the parameters of an HMM given an observed sequence $\{y_t\}_{t=1}^T$ can be formulated as a maximum likelihood (ML) problem:

$$\mathbf{\Lambda}^* = \arg\max_{\mathbf{\Lambda}} \prod_{t=1}^{T} \Pr(Y_t = y_t|\mathbf{\Lambda}). \tag{4.4}$$

The well-known method to estimate $\mathbf{\Lambda}^*$ is the Baum-Welch algorithm, which is a special case of the EM algorithm, which in turn makes use of the forward-backward algorithm [8] to compute the marginal log-likelihood. Refer to [82] for more details on HMMs.

4.6.2 Centroid Heuristic Using HMM

Because of the spatiotemporal nature of the trajectories, each response variable is a tuple (y_k, τ_k) with the spatial component y_k being the discrete locations drawn from $\mathcal{G}$ as a multinomial distribution[4], and the temporal component τ_k being the continuous timestamp drawn from a Gaussian distribution $\mathcal{N}(\mu_k, \sigma_k)$ $(1 \leq k \leq N)$. Timestamp can be modeled as a continuous random variable because I can set a continuous temporal range from the earliest timestamp B_L to the latest one B_U for all $i \in \mathcal{S}$.

I fit the HMM using the trajectories $s^{(i)}$ for all $i \in \mathcal{S}$ using (y_k, τ_k) as the bivariate response. To select the optimal number of states N^*, I use the Bayesian Information Criterion (BIC), a penalized likelihood criterion for model selection [33]. I begin fitting with the simplest model where $N = 2$. At each iteration, as long as the BIC_N of this step is still less than that of the previous BIC_{N-1} (i.e., BIC keeps decreasing as the fitness improves while accounting for model complexity), I keep incrementing N. I stop when the current BIC becomes greater than the previous, i.e., it has reached the "elbow". The optimal number of states N^* is that of the previous step.

I use the set of states S to define the clusters of locations, where each $S_k \in S$ forms a cluster. For each S_k, I extract the parameter vector $\Theta_k = (\theta_1, \ldots, \theta_d)$ of the discrete multinomial response, which is a vector of probabilities of each location $j \in \mathcal{G}$ being visited while the agent is in the cluster S_k. Let C_j be the coordinates (latitude and longitude) of each location $j \in \mathcal{G}$, I compute the coordinates of the **cluster centroid** r_k of S_k as the weighted sum $r_k = \sum_{j=1}^{d} \theta_j C_j$. As a result, locations with high probabilities (i.e., likely to be in the cluster) have more weights, while those with low probabilities (i.e., unlikely to be in the cluster) have less weights.

Fig. 4.3 illustrates the concept. It shows the real-world locations of attractions in the theme park considered in the experiments being mapped to their nearest cluster centroids derived from the hidden states of a 4-state HMM. The HMM was fitted using real-world trajectories of visitors to the theme park. In the figure, the attractions are indicated by filled circles and the mappings indicated by straight lines emanating from the centroids. Coordinates of the centroids are computed by the weighted sums as described above.

[4]This is because, at any time, a location $j \in \mathcal{G}$ has a certain probability of being visited by the agent, and the probabilities of all locations must sum up to 1.

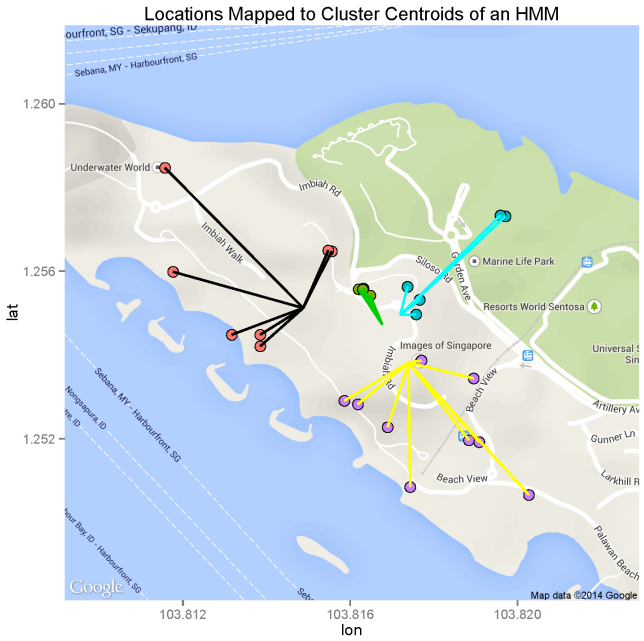

FIGURE 4.3: Illustration of POIs being mapped to their nearest cluster centroids (i.e., reference points) derived from the hidden states of a 4-state HMM. Four states of the HMM form four distinct clusters, each having a cluster centroid as illustrated. Mappings are indicated by straight lines emanating from the respective centroids. The HMM was fitted using real-world trajectories collected in the Sentosa theme park.

Attractions filled with the same colors are in the same cluster (i.e., they having the same centroid mapping) according to the heuristic.

4.7 Experiments

4.7.1 Dataset

I collaborated with the Sentosa theme park in Singapore to collect data from their visitors. My dataset contains the visitors' trajectories for the first 4 months of 2014. The dataset comes from an attraction bundling scheme marketed by the developer under which visitors can select any $Q = 4$ attractions out of a set of 16 and pay upfront a fixed price (independent of their choice). Visitors can redeem their chosen attractions on a chosen day and during a specified period from 9 a.m. to 7 p.m. of the day only. Each chosen attraction can only be redeemed once.

The dataset contains the trajectories of $n = 6,400$ unique and independent visitors (i.e., if a visitor is observed to have traveled in a group of the same trajectories, I take only one

member of the group). I also have certain demographic features of the visitors, which are not discussed here for brevity. Table 4.2 summarizes the sequence length (l) and the first timestamp (τ_1) variables of the dataset. It shows that not everyone managed to redeem all 4 attractions they had chosen, although the majority did. Indeed, about 74.69% of the visitors managed to redeem all 4. Variable τ_1 measures the number minutes since the reference time (9 a.m.) to the first redemption, which can partially explain: while those who arrived early enough could redeem all 4, while those who came "late" couldn't (as their ticket expired at 7 p.m. on the day).

	Min.	Q_1	Median	Mean	Q_3	Max.
l	1.00	4.00	4.00	3.78	4.00	4.00
τ_1	8.57	173.60	254.70	259.70	343.10	604.90

TABLE 4.2: Summary statistics of the sequence length (l) and first timestamp τ_1 variables. Q_1 and Q_3 means the first and the third quartile, respectively.

4.7.2 Baseline Methods

I use the following baseline methods for comparison. In all of my experiments, I base my predictions on the knowledge of the first redemption of some form. The first baseline is to select 3 unique attractions randomly out of the set of 15 (16 less one) given the first attraction in the bundle. I call it the **Random** baseline. The second baseline is to choose $k = 3$ (physically) nearest attractions to the first redemption, which I call the k**-NN** baseline because it is essentially the k-nearest neighbors algorithm.

The third baseline is based on a recent method proposed by Mathew et al. [70] to predict future locations of a mobile agent based on past and current trajectories. The method can be concisely described as follows: (1) Cluster the set of trajectories into K clusters (something similar to Sect. 4.4); (2) Train a separate HMM_k for each cluster k; (3) Given a test agent i, his class label Cl_k^i, and the current trajectory, derive the most likely current state S_t^i of the HMM_k that i is in using Bayes' rule; and (4) Using the forward algorithm, derive the next sequence of 3 most likely locations conditioned on S_t^i. In my case, the current trajectory is simply the first known location and timestamp. I call this the **HMM** baseline because it is heavily based on HMM inference.

4.7.3 Proposed Methods

My first two methods are the implementations the proposed framework using the two heuristics: first-location and centroid heuristic. I call them **VR1** and **VR2** respectively. ("VR" stands for value ratio, which is the central concept of the solution. Refer back to Sect. 4.5.) For VR2, given an agent i's first location y_1^i, I map that to the nearest centroid r_j^i to derive p_j^i. By doing so, I do not need to know the explicit information of y_1^i, but which centroid it is nearest to. I call this the *implicit* information of y_1^i.

The third method is the partial implementation of my proposed framework using the centroid heuristic. Instead of using the full set of centroids derived from the hidden states of an HMM, I take randomly a fraction of that. In particular, given a fitted HMM_k, I select randomly 60% of the number of states of HMM_k to derive a partial set of centroids C'_k. My rationale for this is to empirically estimate the optimality of the full set of centroids, i.e., I want to see how much the accuracy will be decreased (if any) if a partial set of centroids is used for predictions. In other words, I am asking whether the full set of centroids is an optimal set or can I achieve the same level of accuracy using less information? I call it **VR3** for convenience.

For these, I derive a test agent i's class label Cl_k^i using a decision tree trained on their features and first timestamps. Budget constraint B_i is calculated as the remaining time from their first timestamp until 7 p.m. It is worth stressing that for all these methods (including the baselines), except for VR2 and VR3, explicit information of the first location was used for make predictions; hence, the task reduces to predicting 3 locations out of 4. Whereas for VR2 and VR3, implicit information of the first location was used; the task remains predicting a full bundle given incomplete information.

4.7.4 Evaluation

For each agent $i \in \mathcal{T}$, let x_i^* and $\hat{x}_i$ be i's actual and predicted bundle, respectively. Note that x_i^* and $\hat{x}_i$ may not be of the same size. I construct a weighted complete bipartite graph $G = (U = x_i^*, V = \hat{x}_i, E)$ where each edge $e = (x_{ij}^*, \hat{x}_{ik}) \in E$ is weighted by the physical distance between any pair of locations $x_{ij}^* \in x_i^*$ and $\hat{x}_{ik} \in \hat{x}_i$. Denote the weight of e as $w(e)$. Let $\delta(x_i^*, \hat{x}_i)$ be the distance between x_i^* and $\hat{x}_i$, I calculate $\delta(x_i^*, \hat{x}_i)$ using Algorithm 1. The rationale for using physical distance as the benchmark for prediction

accuracy is because my costs are approximated by such distances. Also because businesses of similar nature tend to cluster geographically in real life, two locations are likely close semantically if they are physically close.

Using Algorithm 1, I calculate the distance $\delta(x_i^*, \hat{x}_i)$ for each agent $i \in \mathcal{T}$. To evaluate all the predictions, I take the mean and median distance ($\bar{\delta}$ and $\tilde{\delta}$ respectively) over all $\delta(x_i^*, \hat{x}_i)$. Hence, the lower $\bar{\delta}$ (or $\tilde{\delta}$) is, the more accurate my predictions are on the whole.

Algorithm 1 The evaluation procedure

1: $\delta(x_i^*, \hat{x}_i) \leftarrow 0$
2: **while** $|U| > 0$ and $|V| > 0$ **do**
3: $\quad e^* \leftarrow \min_e(E)$
4: $\quad \delta(x_i^*, \hat{x}_i) \leftarrow \delta(x_i^*, \hat{x}_i) + w(e^*)$
5: $\quad E \leftarrow E - e^*$
6: **end while**

4.7.5 Results

The trajectory clustering results in $K = 4$ clusters (class labels) using $\Delta_\tau = 5$ minutes (refer to Sect. 4.4) for all the agents. The value of K was chosen based on a combination of my domain knowledge and choosing the best clustering consistency index (i.e., the silhouette coefficient). Fig. 4.4 visualizes those 4 clusters. The horizontal axis of each cluster represents the discretized timeline (by Δ_τ) from 9 a.m. to 7 p.m. and the vertical axis represents the probability of each agent belonging to each class being in any one of the 16 attractions at any time interval. The attractions are identified by their unique ID's and color codes shown in the legend at the bottom of the figure. I denote "0" (white) when I don't know the precise location of an agent during a period (i.e., he was not at any particular attraction during the time interval according to the data).

Fig. 4.4 shows that the 4 clusters have rather distinct temporal behaviors: Cl_3 has its peak of activities the earliest, which is followed by Cl_1, then Cl_4, and finally Cl_2. This suggests the existence of 4 different "waves" of visitors that flow through the attractions in the park, from entering, peaking, to exiting. Visually, Cl_3 are the "early birds" and Cl_2 are "latecomers". I also observe certain differences in the preferences for the attractions across the clusters represented by the probabilities of attraction visits. However, these differences are not very distinguishing on the whole: popular attractions

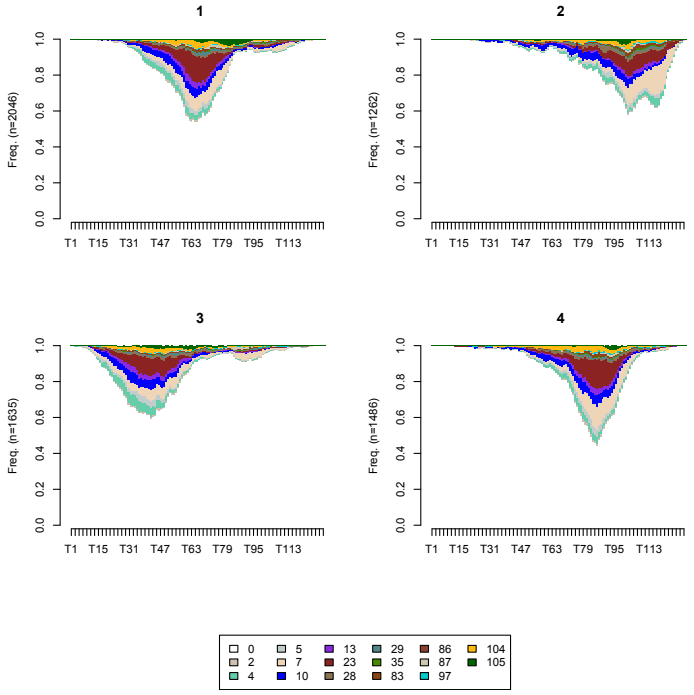

FIGURE 4.4: Visualization of 4 clusters (i.e., "class labels") 1–4 of the trajectory data. Horizontal axes represent the timeline in discrete intervals of 5 minutes from 9 a.m. to 7 p.m. Vertical axes represent the probability of the visitors of each class being in each of the 16 attractions (or at some unknown location "0"), represented by their corresponding color codes whose legend are shown at the bottom of the figure.

remain (more or less) popular across the clusters and unpopular ones remain unpopular. This is particular true for clusters 1, 2, and 4; while for cluster 2, there is a sudden surge in demand for attraction 7 towards the end, which distinguishes it more from the rest.

For each method, I perform a 10-fold cross-validation (CV) to measure its accuracy on predicting bundles. For each fold, I compute the mean $\bar{\delta}$ and median distance $\tilde{\delta}$ of the predictions. I finally compute the average accuracy (i.e., the mean of both $\bar{\delta}$ and $\tilde{\delta}$) over the 10 folds for each method. Fig. 4.5 shows the mean and median accuracies of all the methods considered averaged over their 10-fold CV.

4.7.6 Discussion

Fig. 4.5 shows that my proposed methods (VR1 – VR3) have the most accurate predictions (lowest distances) on average. In particular, the proposed method (VR2) is more accurate than the baselines by at least 20% (i.e., comparing to HMM). The baseline methods are (in the order to decreasing accuracy): HMM, k-NN, and Random, which

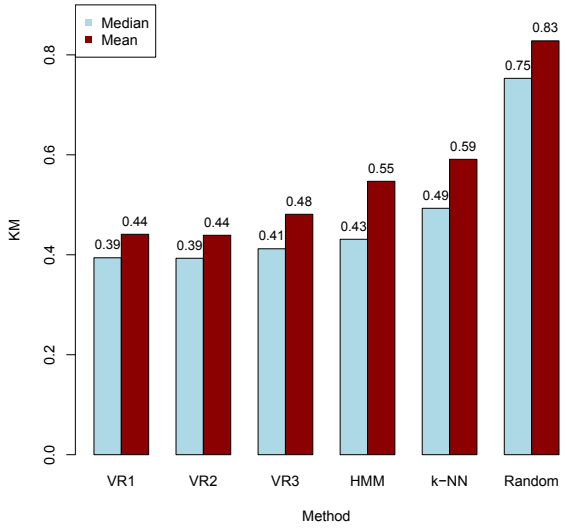

FIGURE 4.5: Accuracies of all the methods benchmarked averaged over 10-fold CV. Accuracies are measured by the mean (and median) distance between predicted and actual bundles in kilometers (KM). The proposed methods (VR1 – VR3) result in better accuracies in general compared to the baselines (HMM, k-NN, and Random).

is not surprising because that is also the decreasing order of their sophistication. Remarkably, using implicit information (VR2), I have achieved as much accuracy as using explicit information (VR1). This empirically supports my centroid heuristic: I only need to know implicitly where an agent intends to visit to make a good enough prediction. At the same time, the centroid heuristic requires much less information to make inferences (i.e., N^* cluster centroids as opposed to the full 16 first locations as in VR1, where N^* is in the range 7–9 in my experiments).

Another notable observation Fig. 4.5 is that randomly selecting 60% of the set of centroids (VR3) does make predictions less accurate, even though by a small amount (for both the mean and median distance). This shows that the full set of centroids is indeed an optimal one such that using less information (VR3) leads to decreased accuracy and using more information (VR1) does not increase the accuracy. On the other hand, while VR3 is technically less accurate than VR2, the difference is really small compared to the reduction in information requirement (VR3 requires 40% less information than VR2). This suggests that my proposed centroid heuristic is also quite resilient to missing information as long as I get most of the reference points right.

4.8 Conclusion

In this chapter, I have introduced the problem of predicting a bundle of goods, where the goods here are a set of spatial locations that an agent wishes to visit. I look at the problem from an economic point of view where agents choose their bundles by optimizing the values of the goods considered over some utility function subject to their budget constraints. To this end, there exists a rich literature to address the problem called revealed preference (RP) analysis. The fundamental problem of RP analysis is to recover the unknown utility functions of the agents given observations of their purchased bundles at the prevailing prices and budget constraints. In this work, I assume the agents have linear utility functions so that the problem reduces to recovering the vector of values of the agents for the goods considered. Motivated by a recent line of work that has established efficient algorithms for learning values from RP data, I adopt and adapt one such algorithm to solve my problem. I also blend in two important techniques from spatiotemporal analysis: trajectory clustering and location clustering in order to make the problem feasible in my particular setting where cost information is unobserved. For location clustering, I propose the centroid heuristic, in which I use HMMs to derive the reference points as cluster centroids based on where the agents use to infer their perceived costs. I experiment my proposed methods with real-world trajectory data collected from a theme park, my predictions are significantly more accurate than the baseline methods. I also see that the proposed centroid heuristic not only requires less information, but it is also resilient to missing information.

There are limitations to this work. First, I have only considered *unordered* sets of spatial locations; however, in reality, agents consume spatial goods by visiting them in sequence. There is an intrinsic *ordering* nature of the goods that I haven't yet taken into account. As a result, comparing between predicted and actual bundles should also consider the sequential order of the goods. Second, the proposed problem and solution may not be applicable to predicting long sequences (both in quantity and geographically) as in such cases, agents typically decide their next future location based on the current one only and not on past locations (i.e., the Markov property). Finally, I have not been able to establish the relationship between the amount of information required to make predictions and its accuracy. These are the work left for the next chapter.

Chapter 5

Trajectory Prediction under Uncertainty & Budget Constraint

5.1 Introduction

How does a rational agent decide to visit a set of locations in space? Assuming there are distinct points of interest (POIs), then the act of visiting them has to happen sequentially. I call it *spatial* sequential decision-making. It is reasonable to assume that each location bears a non-negative utility (reward) to the decision-maker that would not be fully realized until it is visited. Until then, utilities remain *uncertain* and reflect the agent's prior preferences. When making sequential decisions, a rational agent should also weigh in the long-term costs of visiting each of the locations in order to make an optimal plan, where "costs" here are assumed to be proportional to physical distances. Hence, answering the question above would require a model of the agent's sequential decisions for selecting locations, whose utilities remain uncertain and costs are dynamic, and weighing in their long-term consequences into the decision-making [55].

In practice, the agent typically has a limited amount of resources (e.g., time) to run its plan, which I call a *budget*. Such a budget constraint can significantly shape the agent's decision-making process and outcomes in non-obvious ways. This chapter follows up from the previous one to propose a framework based on reinforcement learning [96] to model the agent's spatial sequential decision-making, taking into account the uncertainty of the utilities and the budget constraint. Using the framework, I could discover the

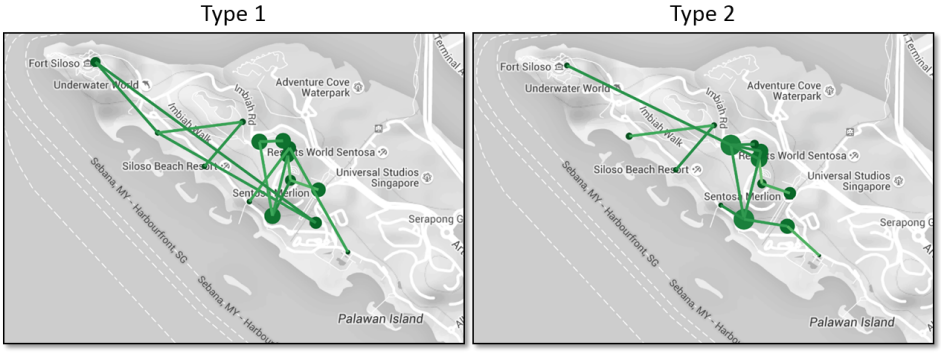

FIGURE 5.1: Visualizing the attractiveness of the same set of POIs in a real-world theme park environment and the pairwise transition probabilities (only those probabilities ≥ 0.20 are drawn) between them as observed by two groups of agents: "Type 1" and "Type 2". Each group is given a certain amount of time budget to visit the set of POIs, where Type 1 has, on average, 114 minutes more than Type 2. The size of each POI is drawn to reflect its relative popularity (attractiveness) among members of each group.

underlying processes that drive real-world behaviors such as the condition for making long-term optimal decisions. Indeed, traditional economic view of rational decision-making as solving an optimization problem often fails to predict reality due to *bounded rationality* [37]. Such discoveries would give insights into real-world human behaviors and help bridge the gap between human and machine intelligence [55, 118].

My motivation comes from the problem of predicting the *next* sequence of location visits (called *trajectory*) of a mobile agent knowing its current trajectory and past observed trajectories of other similar agents. Accurate predictions of the agent's next locations can enable numerous applications of location-based services such as real-time prediction of visitor arrivals and congestion at POIs or devising real-time advertising or adaptive recommendation system for a mobile agent knowing its probable future trajectory.

Consider the example illustrated in Fig. 5.1, whose data were collected from real-world human trajectories in a theme park (to be described in Section 5.6.1). In this setting, suppose there are two groups of agents (human visitors) of equivalent sizes called "type 1" and "type 2". Each agent in each group is to visit the same set of POIs within a given time frame (budget). Agent type 1 is given, on average, 114 minutes more than type 2. Such a budget difference can translate into starkly different behaviors as illustrated in the figure. Not only is the relative attractiveness of each of the POIs different, but the pairwise transition probabilities among them also become discernibly distinct. Type 1 appears to have a larger "coverage" of the POIs through their sequential transitions, while type 2 tends to visit those POIs that are clustered together. These observations

reflect the inherently different underlying decision processes used by these agent types. Thus, in order to make accurate trajectory predictions, it suffices to model the sequential decision-making process of each group separately.

In this chapter, I develop on and extend the capabilities of the framework proposed by Le et al. [57] in the previous chapter for spatial decision modeling. Specifically, I set out to predict an *ordered* sequence of an agent's future locations (as opposed to an *unordered* bundle). Furthermore, my novel contribution is that I do not rely solely on a generative model as previously proposed to generate sequential actions (e.g., naive Bayes [51] or hidden Markov models (HMMs) [70]). Instead, I integrate one of such (i.e., HMMs) into a reinforcement learning framework to model an agent's sequential decisions. I further propose decision models based on the learned utilities resulted from the framework for trajectory prediction. Doing so enables us to explain the underlying processes of the predicted outcomes, the effects of budget constraint on decision-making, and evaluate the appropriateness of the proposed decision models.

5.2 Problem Statement

I consider a set $\mathcal{D}$ of agents and a finite set $\mathcal{G}$ ($|\mathcal{G}| = d$) of POIs (locations). Each agent $i \in \mathcal{D}$ has a utility vector v_i over each location $j \in \mathcal{G}$, where $v_{ij} \in \mathbb{R}_{\geq 0}$ is the utility of j to i. Agent i has a budget constraint B_i and wishes to visit a subset $s_i \subseteq \mathcal{G}$ such that $\sum_{j \in s_i} c_{ij} \leq B_i$, where c_{ij} is i's cost of visiting j. I denote s_i as agent i's *trajectory* that contains the *ordered* sequence of locations visited by i and the corresponding timestamps. Without loss of generality, I assume throughout that the costs and budget constraint are in terms of travel time and i makes a binary decision vector $s_i \in \{0, 1\}^d$. Hence, c_{ij} is a *dynamic* cost for each j that depends on the previous location in the sequence. I additionally assume the proportionality between distance and travel time, where all distances considered in this chapter are spatial Euclidean distance.

Suppose $\mathcal{D}$ can be divided into non-overlapping subsets called *agent types*, where each "type" implies homogeneous preferences and behaviors. Given an agent of a certain type, his partial trajectory (say the first n location visits) and the current budget, my goal is to predict the agent's remaining trajectory. The notion of agent type comes from the idea that modeling each individual agent is impractical. It is much more feasible to

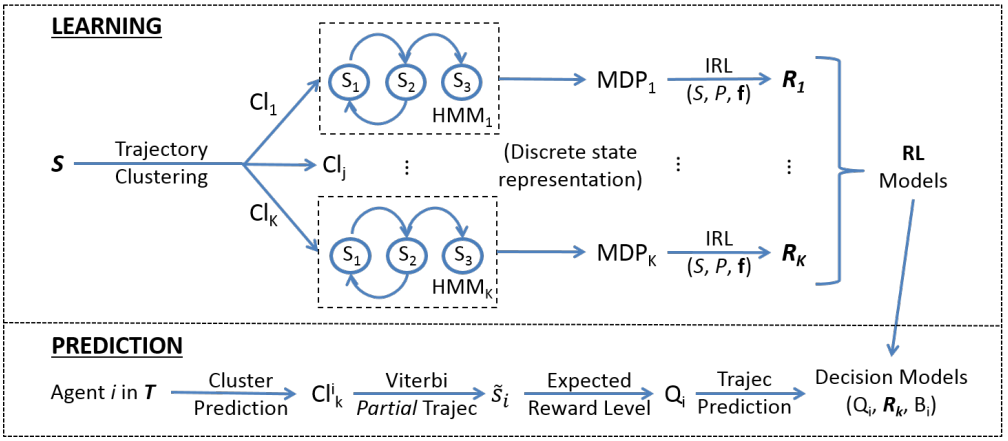

FIGURE 5.2: The framework to model and predict the remaining trajectory of a test agent $i \in \mathcal{T}$ given its observed partial trajectory $\tilde{s}_i$, current budget B_i, and trajectories of the agents in the training set $\mathcal{S}$. S is the finite set of states, P is the matrix of state transition probabilities, and $\mathbf{f}$ is the set of feature vectors of the states in S.

divide them into finite and disjoint *clusters* of similar preferences and behaviors. Thus, I also use the terms "cluster" and "(agent) type" interchangeably.

Predicting an agent's remaining trajectory requires sequential decision modeling under uncertainty and budget constraint. The uncertainty comes from the utility distributions of the remaining locations. While the relative attractiveness of the locations can be easily worked out using a simple frequency count, it is not straightforward how to learn their utility distributions from the observed trajectories and how to incorporate them into a sequential decision-making model.

5.3 Solution Overview

I propose an integrated framework to model and predict the next sequence of locations given an agent's observed partial trajectory and budget constraint. The framework consists of two components: learning and prediction. Learning clusters the agents into finite types, models their sequential visits using a discrete-state transitions and learns from which the utility distribution of each of the locations. Prediction maps a given agent to a type, derives the most probable state sequence of its observed trajectory, estimates a goal that the agent is trying to achieve, and generates the next sequence of visits that would meet that goal. Fig. 5.2 illustrates the overall framework. Table 5.1 summarizes the additional notations used in this chapter.

TABLE 5.1: Summary of additional notations used in this chapter. Reintroduced notations override those introduced earlier.

Notation	Description
v_{ij}, c_{ij}	Utility and cost of location $j \in \mathcal{G}$ for agent i
s_i, $\tilde{s}_i$, B_i	Trajectory, partial trajectory, and current budget of i
l_i	Trajectory (sequence) length of agent i
S	Finite set of (hidden) states for each agent type ($\|S\| = N$)
$P_a(s, s')$	Probability of going from state s to s' by taking action a in s
$\mathbf{f}_s$	Feature vector of each state $s \in S$
$\mathbf{R}_k$	State-reward matrix $\forall$ agent type k ($1 \leq k \leq K$)
$\mathbf{R}_a$	Location-reward matrix for each location $a \in \mathcal{G}$
Q_i	Expected reward level ("personal goal") of agent i

Learning. I first divide the agents in the training set $\mathcal{S}$ into K finite clusters, where each cluster Cl_j ($1 \leq j \leq K$) represents an agent type. K is typically chosen heuristically via some clustering coefficient (e.g., the silhouette index). Using the agents' observed features and the K clusters as class labels, I train a multi-class classifier (e.g., multinomial logistic regression). I also model the environment that the agents interact with as a finite set of states S, where each state $s \in S$ has a distinct vector of features $\mathbf{f}_s$. I use hidden Markov models (HMMs) to transform the observed trajectories into finite sequences of states. Such a representation can then be modeled as a Markov decision process (MDP). The utility of each action (i.e., location visit) can then be derived via the process of inverse reinforcement learning (IRL) using the agents' observed actions (represented in the transition probability matrix P of the MDP). The final outcomes of IRL are the reward matrices $\mathbf{R}$.

Prediction. Given the observed partial trajectory and features of an agent i in the test set $\mathcal{T}$, I first predict i's type Cl_k^i using the trained classifier above. I then use the Viterbi algorithm [32] to find the most probable sequence of states $\tilde{s}_i$ for the observed trajectory. I am then able to model i's goal Q_i (also called the "expected reward level") and predict the next sequence of visits that can meet this goal within budget B_i. I finally propose two decision models that take into account the uncertainty of the utilities (represented by the matrix $\mathbf{R}_k$ for each type k) and budget B_i.

I next elaborate on each of the components of the framework shown in Fig. 5.2.

5.4 Learning

5.4.1 Environment Modeling

I use hidden Markov models (HMMs) to model the environment the agents interact with as a finite set of states $S = \{S_1, S_2, \ldots, S_N\}$. Refer to Section 4.6.1 for background and notations of HMMs, which are reused in this chapter. In my modeling, each emission y_t of a hidden state x_t is a tuple (y_k, τ_k) with the spatial component y_k being a discrete location drawn from $\mathcal{G}$ and the temporal component τ_k being a continuous timestamp drawn from the Gaussian distribution $\mathcal{N}(\mu_k, \sigma_k)$ $(1 \leq k \leq N)$.

Each hidden state of the HMM can be thought of as a *spatiotemporal cluster* of the visiting activities. Empirical observations confirm that nearby locations are much more likely to be visited sequentially in short periods of time, i.e., having "high" emission probabilities. I fit the HMMs using the trajectories $s^{(i)}$ $\forall i \in \mathcal{S}$. A well-known method to estimate the parameters of an HMM is the Baum-Welch algorithm [8]. For each HMM_j $(1 \leq j \leq K)$, I select the optimal number of states N_j^* using the Bayesian Information Criterion (BIC) [33] as described in Section 4.6.2. An important inference problem is that given a sequence of observations, find the most probable sequence of hidden states that produces it, which can be solved using the Viterbi algorithm [32].

5.4.2 Inverse Reinforcement Learning

5.4.2.1 Preliminaries

Markov decision processes (MDPs) [11] provide an elegant framework to model sequential decisions in an environment represented as a finite state space S. At each state $s \in S$, the agent chooses an action $a \in A$. Upon which, the process transitions into the next state $s' \in S$ according to the probability $P_a(s, s') = \Pr(S_{t+1} = s' | S_t = s, a_t = a)$. The agent then receives a reward $R_a(s, s')$. The main concern of MDP is to find an optimal policy $\pi^* : S \mapsto A$ that maximizes the long-term *cumulative* reward $\sum_t R_{a_t}(s_t, s_{t+1})$.

Let $P_{\pi(s)}$ represent the transition probability matrix corresponding to the application of some policy π. A finite-horizon MDP is completely described by the tuple $(S, A, P_{\pi(s)}, R)$. The value function $V^\pi(s)$ of policy π at state s represents the expected

cumulative reward from s. Thus, my goal is to find an optimal policy π^* such that $V^{\pi^*}(s)$ is maximized. It can be shown that there exists at least one optimal policy such that $V^{\pi}(s)$ is maximized for all $s \in S$ [96] that can be expressed as:

$$\pi^*(s) \in \arg\max_{a \in A} \sum_{s' \in S} P_a(s, s')[R(s, s') + \gamma V^{\pi}(s')]. \tag{5.1}$$

A fundamental property of the value function is, for any policy π and any state s:

$$V^{\pi}(s) = R_{\pi(s)}(s) + \sum_{s' \in S} P_{\pi(s)}(s, s') V^{\pi}(s'). \tag{5.2}$$

Eqn. (5.2) (famously called the Bellman equation) directly gives rise to efficient dynamic programming (DP) formulations to find a long-term optimal policy π^* [11].

Inverse reinforcement learning (IRL) is the inverse problem to MDP, whose goal is to determine the reward function R that is being optimized given observations of the sequential decisions. Ng and Russell [75] originally propose LP formulations to solve the problem with constraints leading to the optimal observed policy. Abbeel and Ng [2] later propose a strategy of *matching feature expectations* between an observed policy and an agent's behaviors. The strategy is both necessary and sufficient to achieve the same performance as if the agent were in fact solving an MDP with reward function linear in the features of the states. Denote ξ_i a state-based trajectory (a.k.a. a "path"), $\mathbf{f}$ the sequence of feature vectors of a path, and $\bar{\mathbf{f}} = \frac{1}{m} \sum_i \mathbf{f}_{\xi_i}$ the empirical expected feature count based on m trajectories. Matching feature expectations is described by:

$$\sum_{\xi_i} \Pr(\xi_i) \mathbf{f}_{\xi_i} = \bar{\mathbf{f}}. \tag{5.3}$$

In this work, I adopt the maximum entropy (MaxEnt) IRL algorithm [117] to learn the reward[1] distribution of each state. MaxEnt IRL is an effective framework for modeling and understanding human activities, where the recovered reward function intuitively encodes an individual's set of preferences [43]. The notion of reward distribution comes from the fact that different people, even if classified into types, would still have different preferences (utilities) for the same thing. Such diversity in tastes can be best modeled as a probability distribution.

[1] In the context of the problem studied, reward is a positive utility received by visiting a POI. Given that a "state" is a cluster of POIs, reward of a state is the sum of the utilities of the POIs in the cluster.

5.4.2.2 Maximum Entropy IRL (MaxEnt IRL)

Given a state-action sequence $\xi = \{(s, a)\}_i$, where $s_i \in S$ and $a_i \in A$, agent i is optimizing some function that linearly maps the features of each state $\mathbf{f}_{s_j} \in \mathbb{R}^k$ to a reward value that represents i's utility of visiting that state. This function is parameterized by some weight vector θ and the reward of a trajectory is simply the sum of all the state rewards along the path. The reward weights are applied to the path feature counts $\mathbf{f}_\xi = \sum_{s_i \in \xi} \mathbf{f}_{s_j}$ such that the reward of the trajectory is the weighted sum of the feature counts along the path:

$$R(\mathbf{f}_\xi) = \theta \cdot \mathbf{f}_\xi = \sum_{s_j \in \xi} \theta \cdot \mathbf{f}_{s_j}. \tag{5.4}$$

Since many distributions of paths may match the feature counts and any one distribution from among this set may exhibit a preference for some of the paths over others not implied by the path features. Such ambiguity is solved using the principle of *maximum entropy* by choosing the distribution that does not exhibit any additional preferences beyond matching feature expectations. The resulting distribution over the paths is parameterized by the weights θ:

$$\Pr(\xi_i|\theta) = \frac{1}{Z(\theta)} e^{\theta \cdot \mathbf{f}_{\xi_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \xi_i} \theta \cdot \mathbf{f}_{s_j}}, \tag{5.5}$$

where $Z(\theta)$ is some *partition function* for the parameter weights. This distribution also provides a *stochastic policy* (i.e., a distribution over the actions at each state). Refer to [117] for more details. The probability of each action is weighted by the expected exponentiated rewards of all paths that begin with that action. Maximizing the entropy of the distribution over paths subject to feature constraints implies that I maximize the likelihood of the observed data under maximum entropy distribution, which is convex and whose optima can be obtained using gradient-based optimization methods [117].

I now build an MDP model (S, A, P, R) for each agent type, where S is the set of states of the corresponding HMM and A is the set $\mathcal{G}$ of locations. I then need a set of state sequences in order to derive the transition matrix P and reward function R. To this end, I convert each trajectory into its most probable sequence of (hidden) states using the Viterbi algorithm [32]. P is then derived by sampling the observed state transitions and action taken at each state.

MaxEnt IRL additionally requires a set of features $\mathbf{f}_s$ for each state $s \in S$. I use the spatiotemporal characteristics of each state as its features. Specifically, recall that each state S_i of the HMM is both a spatial cluster (i.e., what locations are likely to be visited) and a temporal cluster (described by the Gaussian mean μ_i). I use the tuple $(lo_i, la_i, \mu_i, \sigma_i)$ as the features $\mathbf{f}_{S_i}$ of S_i, where lo_i and la_i are the "mean"[2] longitude and latitude coordinates of S_i and μ_i and σ_i are the mean and standard deviation of the Gaussian emission, respectively. Such weighted sum of the coordinates are referred to as the "cluster centroids" of the states. Hence, each state S_i admits a unique cluster centroid C_i described by its (lo_i, la_i).

Each run j of MaxEnt IRL produces a unique reward function $R_j : S_i \mapsto \mathbb{R}^+, \forall 1 \leq i \leq N$. In order to produce a *distribution* of reward for each state, I split the trajectories into subsets and run MaxEnt IRL on each subset to get a unique reward function. The probability of each reward value is the proportion of the subset in the original set. Towards this end, I split the trajectories into subsets of equal sequence lengths and run MaxEnt IRL on each of them.

The distribution of reward for each location is computed as follows. Let $\mathbf{R}_s$ be a state-reward matrix. $\mathbf{R}_s$ is of dimension $l \times N$, where N is the number of states and l is the maximum sequence length. For each state S_k $(1 \leq k \leq N)$, let p_k of length $d = |\mathcal{G}|$ be the vector of multinomial emission probabilities of the HMM. Let Π be the multinomial emission matrix of dimension $N \times d$ whose row vectors are p_k. I compute the location-reward matrix $\mathbf{R}_a$ as:

$$\mathbf{R}_a = \mathbf{R}_s \times \Pi. \tag{5.6}$$

Assuming the stochastic reward $R(a)$ of each location a follows a Gaussian distribution, its mean and variance can be derived from the corresponding column vector of $\mathbf{R}_a$.

5.5 Prediction

I present two decision models to the problem of trajectory prediction: Adaptive MDP (AMDP) and Value Ratio (VR). The former follows the long-term optimal policy of an MDP and the latter uses myopic greedy heuristics to make decisions.

[2]Precisely, lo_i and la_i are the sum of the coordinates of the locations weighted by the multinomial emission probabilities at S_i.

5.5.1 Adaptive MDP

Empirical evidence shows that the sequence lengths of the trajectories typically follow normal distributions [55, 56, 64]. I take advantage of this to introduce stochasticity of reward and policy into my model by splitting the training set into subsets of the same sequence lengths. For each subset, I learn a unique reward/policy function. In the end, I come up with a reward/policy matrix, where for each matrix, the columns are the states and the rows are the sequence lengths whose probability distribution follows that of the sequence lengths.

I now obtain the following matrices from the training set for each agent type:

1. $\mathbf{R}_s$ (or $\mathbf{R}$): each entry is the reward (column) of each state that corresponds to each sequence length (row);
2. $\mathbf{V}$ ($l \times N$): each entry is the value (column) of each state that corresponds to each sequence length (row);
3. Optimal policy matrix Π^* ($l \times N$): each entry is an optimal action $a \in A$ at each state (column) that corresponds to each sequence length (row).

From $\mathbf{R}$, I am able to derive the Gaussian distribution of reward $R(s)$ at each state $s \in S$ using the probability distribution of the sequence length (i.e., the rows).

An important consideration in my model is the agent's **expected reward level**. This comes about from the observation that an agent may finish its trajectory even when there is sufficient budget to go on. Such behavior may come from an intrinsic expected reward level, such as a "personal goal", having been met. Once such goal is met, the agent would just be happy to finish there and then and not go on to maximize the cumulative reward any further. In order to model such a personal goal, I make use of the value function. From Eqn. (5.2), the value function at state s is sum of the immediate reward $R_{\pi(s)}(s)$ and the future expected reward. I use this future expected reward to model agent i's expected reward level Q_i:

$$Q_i = V^{\pi}(s) - R_{\pi(s)}(s). \tag{5.7}$$

Algorithm 2 Adaptive MDP decision model for agent i

1: Given agent i's partial trajectory $\tilde{s}_i = \{(s, a)\}_i$ of current length n and i's current budget $B_i > 0$
2: Let $s = \tilde{s}_i[n]$ be the current state
3: Sample reward $R(s)$ from Gaussian distribution
4: Retrieve current state's value $\mathbf{V}[n, s]$
5: Let $Q_i = \mathbf{V}[n, s] - R(s)$ be i's expected reward level
6: Initialize i's future cumulative reward $U_i \leftarrow 0$
7: Let $\hat{s}_i \leftarrow \varnothing$ be the predicted sequence
8: **while** $U_i < Q_i$ and $B_i > 0$ **do**
9: Sample an action a from policy $\Pi^*[n : l, s]$
10: **while** $a \in \tilde{s}_i$ {a has been visited} **do**
11: Repeat Step 9
12: **end while**
13: Sample next state s' from $P_a(s, s')$
14: Update $n \leftarrow n + 1$; $s \leftarrow s'$
15: Update $\hat{s}_i \leftarrow \hat{s}_i \cup (s, a)$; $\tilde{s}_i \leftarrow \tilde{s}_i \cup \hat{s}_i$
16: Sample reward $R(s)$ from Gaussian distribution
17: Let t_a be the travel time from current location to a
18: Let Δ_a be the minimum duration to be spent at a
19: Update $U_i \leftarrow U_i + R(s)$; $B_i \leftarrow B_i - (t_a + \Delta_a)$
20: **end while**
21: Return the sequence of actions in $\hat{s}_i$

Since both $V^{\pi}(s)$ and $R_{\pi(s)}(s)$ are given (by $\mathbf{V}$ and $R(s)$, respectively), I can derive Q_i for each agent i knowing its current state s and the sequence length n. Furthermore, the optimal policy matrix Π^* is *stochastic* because, given a state s, each column vector of policies $\Pi^*[:, s]$ is distributed according to the Gaussian distribution of the sequence length. Algorithm 2 describes the Adaptive MDP decision model.

Algorithm 2 follows the long-term optimal policy of an MDP because it makes use of the optimal (stochastic) policy function to make decision at each step. The policy function is long-term optimal as a result of solving the Bellman equation (5.2).

5.5.2 Value Ratio

At each time step, the agent samples a random reward value r_j from the Gaussian distribution $R(a_j)$ of each of the *remaining* locations a_j. Given its current location, the agent heuristically maps itself to the nearest *cluster centroid* (refer to Sect. 5.4.2.2) as a "point of reference" and derives the distances d_j from the cluster centroid to each of the remaining locations. The agent then chooses to visit the location j^* that has the largest ratio r_j/d_j (i.e., the ratio of the immediate reward to its cost) and repeats until

its budget runs out or there is no unvisited location left. This is the well-known best "bang-for-the-buck" greedy heuristic [7]. Algorithm 3 describes the model.

Algorithm 3 Value Ratio decision model for agent i

Given agent i's current location a_i, its current set of *unvisited* locations $\mathcal{G}_i \subseteq \mathcal{G}$ and the current budget $B_i > 0$
Let $\hat{s}_i \leftarrow \varnothing$ be the predicted sequence of visits
while $|\mathcal{G}_i| > 0$ and $B_i > 0$ **do**
  Sample reward r_j from Gaussian distribution for each $a_j \in \mathcal{G}_i$
  Let $C_{k^*} = \arg\min_k \texttt{distance}(a_i, C_k)$ $(1 \leq k \leq N)$
  Let $d_j = \texttt{distance}(a_j, C_{k^*})$, $\forall a_j \in \mathcal{G}_i$
  Select a_{j^*} where $j^* = \arg\max_j r_j/d_j$, $\forall a_j \in \mathcal{G}_i$
  Update $\hat{s}_i \leftarrow \hat{s}_i \cup \{a_{j^*}\}$; $\mathcal{G}_i \leftarrow \mathcal{G}_i \setminus \{a_{j^*}\}$
  Let t_{j^*} be the travel time from a_i to a_{j^*}
  Let Δ_{j^*} be the minimum duration to be spent at a_{j^*}
  Update $B_i \leftarrow B_i - (t_{j^*} + \Delta_{j^*})$; $a_i \leftarrow a_{j^*}$
end while
Return $\hat{s}_i$

5.6 Experiments

5.6.1 Dataset

I collaborated with the Sentosa theme park in Singapore to conduct experiments and collect demographic and behavioral data from their visitors from January to April, 2014. The dataset contains the visitors' trajectories tracked using RFID devices. In the experiments, visitors pay upfront a fixed amount in order to redeem up to 14 participating attractions. Visitors can only redeem the attractions during the specified 10-hour period from 9 a.m. to 7 p.m. on a chosen day. Each attraction can only be visited once.

My dataset $\mathcal{D}$ contains trajectories of $3,867$ unique and independent visitors together with their demographic features. The empirical distribution of the sequence length of these trajectories follows a typical bell-shaped characteristic of a Gaussian distribution.

5.6.2 Trajectory Clustering

I perform cross-validations[3] (CVs) on $\mathcal{D}$. For each fold, the training set $\mathcal{S}$ is used for trajectory clustering and decision modeling. My hierarchical clustering results in $K = 2$

[3]Precisely, I performed 3-fold CV to ensure a large enough training/test partition per fold.

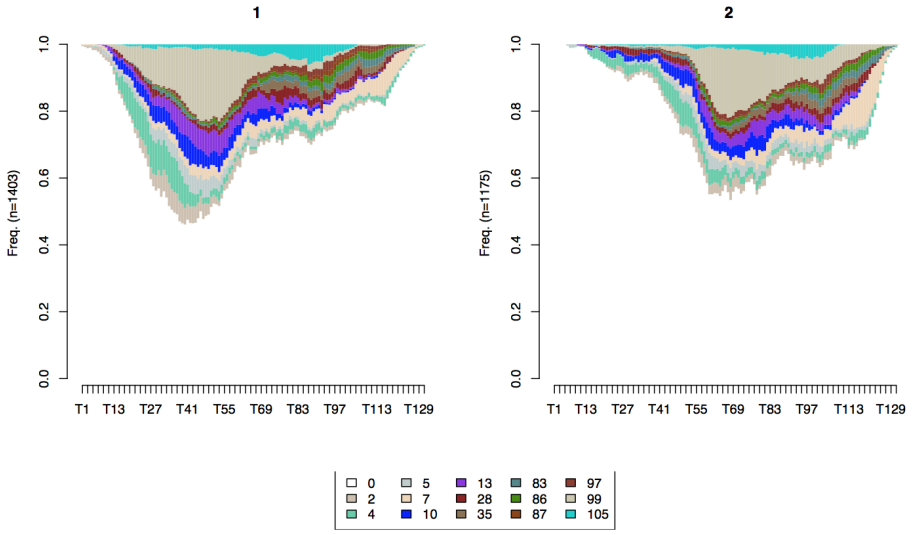

FIGURE 5.3: Visualization of the two clusters (agent types) of the training data in the experiments. Horizontal axes represent the timeline in discrete intervals of 5 minutes from 9 a.m. to 7 p.m. Vertical axes represent the probability of agent of each type being at each of the attractions (or at some unknown location "0") shown in the legend.

clusters using the interval $\Delta_\tau = 5$ minutes (refer to Section 4.4) for all the agents. The value of K was chosen based on inspection of the hierarchical tree and empirical goodness of clustering via the silhouette coefficient [46] (i.e., partitions of comparable sizes and good in-group cohesiveness).

Fig. 5.3 visualizes the 2 clusters using training data of one of the random folds. The horizontal axes represent the discretized timeline (by Δ_τ) from 9 a.m. to 7 p.m. for each cluster and the vertical axis represents the probability for each agent of each cluster to be at any one of the 14 attractions at any interval. (Note that even after 7 p.m., some activities can still be recorded in the park.) The attractions are identified by their numbers whose color codes are shown in the legend at the bottom. I denote "0" (white) when I do not know for sure the location of an agent during a given time interval (i.e., he was not observed at any known attraction during the interval). It can be seen that, most of the time, visitors hang out in the park without visiting any specific attractions.

The trajectory clustering reveals that the main differences between the two agent types are their temporal behaviors. Agent type 1 tends to arrive earlier and has their peak of visiting activities earlier in the day (around 12–1 p.m.), and then (their visit frequency) sharply drops off. Whereas, agent type 2 tends to arrive much later and reaches their peak later (at round 3–4 p.m.), and then gradually declines. If budget is defined as the

duration from the time of entry until the closing time (7 p.m.), then agent type 1 has, on average, 114 minutes more than agent type 2. As a result, I call agent type 1 the "early birds" and agent type 2 the "latecomers". The two clusters have roughly comparable sizes with cluster 1 being 54.42% and cluster 2 being 45.58% of the set training $\mathcal{S}$.

5.6.3 Evaluation

For each cluster in $\mathcal{S}$, I learn the matrices $\mathbf{R}$, $\mathbf{V}$, and Π^*. The test set $\mathcal{T}$ is used to validate the predicted trajectories. For each agent $i \in T$, let l_i be i's final sequence length. I first predict i's type using its features and first timestamp via a multinomial logistic model. Given i's partial trajectory of length n, I predict i's remaining trajectory while varying $n \in [2, l_i - 1]$. Let s_i^* and $\hat{s}_i$ be i's actual and predicted remaining trajectory, respectively. I use the Levenshtein edit distance [12] to quantify the similarity between s_i^* and $\hat{s}_i$. Each match receives a fixed positive score and each mismatch incurs a negative penalty proportional to the distance between the two locations.

The following baseline models are used for evaluation: At each time step,

1. **HMM.** Predict agent i's current state s, generate an *unvisited* location based on the state's multinomial probabilities p_s and repeat until its budget B_i runs out. This method is based on [70].
2. **Nearest neighbor.** Agent i redeems a remaining location that is nearest to its current location and repeats until B_i runs out.
3. **Random.** i redeems a random unvisited location and repeats until B_i runs out.

5.6.4 Results

My experimental results are summarized in Fig. 5.4 and 5.5. In Fig. 5.4, the mean reward per attraction learned from IRL and Eqn. (5.6) is plotted together with its 95% confidence interval (top panel). The figure shows that the mean rewards, in general, faithfully reflect their respective empirical probabilities of attraction visit for both agent types (i.e., their preferences – in the bottom panel).

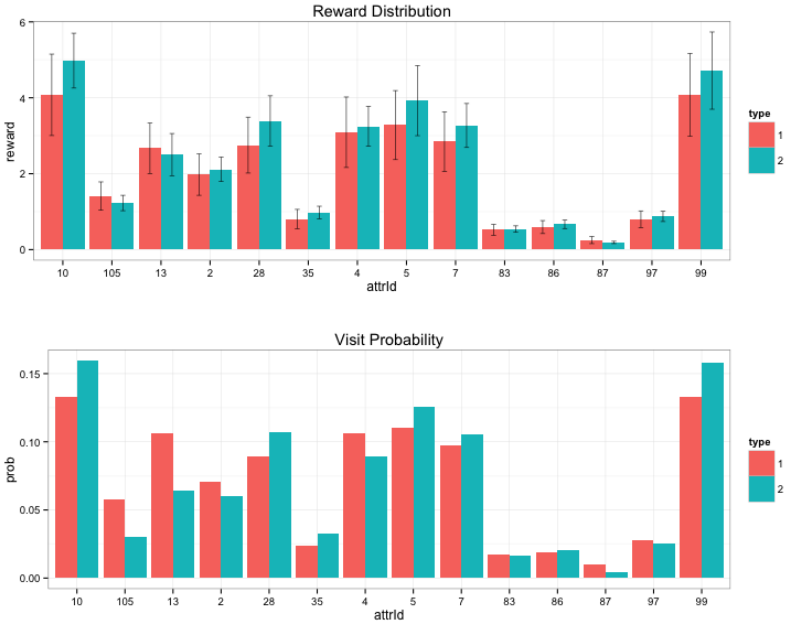

FIGURE 5.4: Comparison between the estimated reward distribution for each attraction ("attrId") (top) and the empirical visit probability for each attraction (bottom).

It is noteworthy to observe in Fig. 5.4 that agent type 2 has, for the most part, higher (absolute) *immediate* reward per attraction (top panel) than agent type 1. This consequentially differentiates the underlying decision processes employed by the two agent types. Fig. 5.5 shows the distributions of the similarity measures (means and variances – represented by 95% confidence bars) across the models. Each distribution is computed from the cross-validation while varying the observed partial trajectory length $n \in [2, l_i - 1]$. A higher mean similarity implies a more accurate prediction, on average. These distributions (in Fig. 5.5) are empirically verified to be Gaussian.

For agent type 1, Fig. 5.5 shows that the Adaptive MDP model has the most accurate prediction, on average. The Value Ratio and HMM model both have about the same second best average prediction score. The Random baseline model has the least accurate average prediction, which is quite reasonable, followed by the Nearest Neighbor model. For agent type 2, the figure shows that the Adaptive MDP model performs marginally worse than the Value Ratio model, even though it still fares much better than the other baselines. In other words, the Value Ratio model makes the most accurate prediction on average, for this group of agents. This is an interesting result worth further discussion.

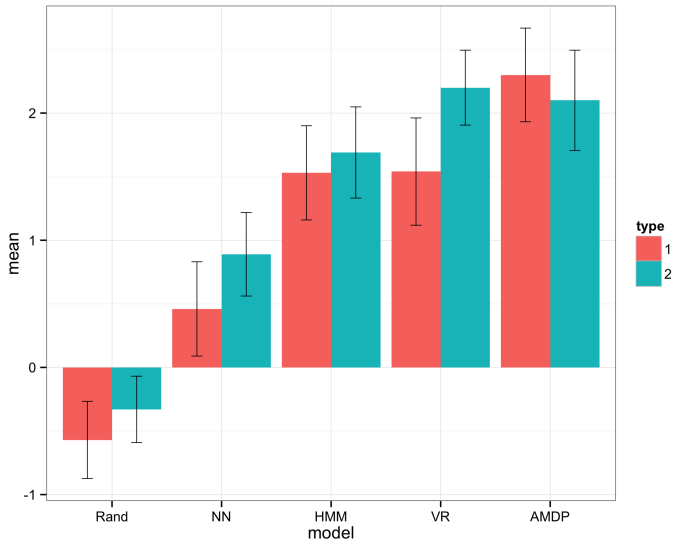

FIGURE 5.5: Similarity measures between the actual and predicted trajectories for proposed and baseline models: Random ("Rand"), Nearest Neighbor ("NN"), HMM, Value Ratio ("VR") and Adaptive MDP ("AMDP").

5.6.5 Discussion

From trajectory clustering, I have discovered that agent type 1 are the "early birds" and agent type 2 are the "latecomers". From the perspective of modeling, agent type 1 has a much larger budget (by 114 minutes, on average) than agent type 2 does. Larger budget means more flexibility, more foresight and better long-term planning, which is what the Adaptive MDP model reflects: it embodies the long-term optimal policy of MDP. This indeed performs better than other short-sighted baseline models.

Whereas, a smaller budget, which agent type 2 has, translates into less flexibility and less time for careful planning, which ultimately results in more myopic and suboptimal decisions (i.e., resorting to greedy strategies). This is reflected in the experimental results, where the greedy and myopic Value Ratio model performs the best for agent type 2 (even though just marginally better than Adaptive MDP). This myopic decision-making corroborates with the observations in Fig. 5.4, where most of agent type 2's immediate rewards are larger (in absolute terms) than agent type 1's such that it sees less values in *delayed* (future) rewards and finds more incentives to act greedily [96]. This is also evidenced in Fig. 5.1, where agent type 2 has much stronger tendencies to visit nearby attractions (i.e., maximizing the value ratio) than type 1.

5.7 Conclusion

In this chapter, I address the problem unresolved from the previous chapter: trajectory prediction. I use reinforcement learning to model the agent's sequential decisions. By doing so, I have discovered from real-world trajectories how people make decisions: they make more optimal decisions when given enough time to do so. This is perhaps not surprising in retrospect, because it is reasonable that foresighted decisions and careful plans need time to coordinate, while myopic ones do not (as only the immediate rewards are considered). On the other hand, this also validates my framework's ability to model real-world behaviors by finding out what makes reasonable sense in real life.

My main shortcoming here is the simplistic handling of the budget constraint. I would like to see if handling it in more sophisticated ways would improve predictions. For example, for foresighted agents, I would like to experiment with decision models other than MDP in my future work. One of which is the adaptive stochastic knapsack [45], which it is similar to a traditional knapsack model except for the sequential decisions and stochastic reward of each item. Another shortcoming of this work is the simplistic Value Ratio model for myopic decision-making (type 2), which yields just slightly better prediction than the Adaptive MDP for agent type 2. Hence, for myopic agents, a more sophisticated decision model may be desirable to better model and predict their behaviors. One of such model for sequential decisions has been proposed in the operations research literature [94]. This is also worth investigating in the future work.

Chapter 6

Fine-grained Traffic Speed Prediction Using Local Gaussian Processes

6.1 Introduction

Big data captured in densely populated urban environments can provide multi-scaled perspectives at the complex behaviors of urban systems in both space and time. Recent advances in big data technologies such as sensor networks and the Internet of Things (IoT) have accelerated the pace of spatiotemporal data collection in urban settings at ever finer-grained scale. Such wealth of data can be turned into valuable knowledge and insights that can be used to make cities more efficient, safer and enhance the living standard of urban residents. This is a significant utility of big data as it has been forecasted that, by 2050, 66% of the world's population will be urban dwellers [41].

Traffic speed is a key measure of the efficiency of a city's transportation system and the mobility of its residents. Accurate modeling and prediction of traffic speed in a city are therefore crucial to the city's intelligent transportation systems (ITS) [104, 110]. Traffic speed data are typically obtained from **two main sources**: one from GPS trajectories generated by moving vehicles equipped with GPS trackers (e.g., taxicabs), and another from static traffic readers or sensors located at fixed locations (e.g., traffic cameras or loop detectors). GPS trajectories are often used as active mobile probes that can directly

measure travel times and speeds along road segments [16, 29, 44, 81, 100]. However, using such active probes also incurs high measurement variance due to inconsistent driving behaviors and lack of control over route choices. Hence, a critical mass of probes is needed for each road segment to obtain reliable measurements. Meanwhile, static traffic sensors typically provide sparse spatial coverage due to their high installation and maintenance costs. This leaves many road segments uncovered and unobserved and makes it hard to accurately infer traffic speed. Indeed, recent surveys have indicated that in most modern cities, only a few main roads have loop detectors installed [18, 89].

I address the problem of **fine-grained** traffic speed modeling and prediction in *real-time*, where "fine-grained" here means extensive spatial coverage and fine temporal scales. With fast and reliable traffic prediction, travelers can optimize their routes dynamically. Traffic managers can also use such information to quickly develop proactive traffic control strategies and make better use of the available transportation resources. Although many navigation systems currently provide live traffic information for routing services, their coverage is limited to major road segments and lacks the predictive capabilities of *future* traffic conditions based on recent observations and historical data [18, 81]. In addition, traffic speed in densely populated urban areas is often subject to short-term random fluctuations and perturbations due to exogenous events such as weather conditions, emergencies or traffic incidents [20]. As a result, I focus on *short-term* traffic prediction[1] because I find the problem more realistic and challenging.

Gaussian processes (GPs) have been repeatedly demonstrated to be an effective tool for modeling and predicting various traffic phenomena such as mobility demand [20], traffic congestion [65], short-term traffic volume [104], travel time [44], and pedestrian and public transit flows in urban areas [74]. Indeed, comparative studies on short-term traffic volume prediction showed that GPs outperform other methods such as autoregressive integrated moving average, support vector machine, and multilayer feedforward neural network for the task [104, 110]. A particularly attractive feature of GPs is their fully non-parametric Bayesian formulation, which allows for explicit probabilistic interpretation of the model outputs and confidence interval estimations [20, 92, 104]. Unfortunately, GPs admit cubic time complexity in the size of the training data. This has been a major limiting factor for the adoption of GPs to model big traffic data [20, 65, 67].

[1] "Short-term" can be subjectively defined based on the temporal scale of the sensor readings.

I address the problem of **efficient GPs** for real-time traffic speed prediction based on the idea of clustering spatiotemporal traffic data into "local" subsets of correlated traffic patterns. I call such clustering *localization* [76, 92]. From each subset, a local GP can be trained to make predictions of future traffic queries that could be heuristically mapped to it using some similarity measure. Speed in each local subset is assumed to have similar behaviors through space and time. To this end, I propose to use non-negative matrix factorization (NMF) for fast localization. The idea of using local GPs to infer data of clustered nature is not entirely new. Indeed, Snelson and Ghahramani [92] first proposed local GPs for non-linear regression, where clustering is done based on similarity of the responses in the training data. In this work, my adoption of the idea using NMF for efficient traffic speed prediction is novel to the best of my knowledge. I am able to empirically show significant improvements in both runtime performances and prediction accuracies in diverse urban and geospatial settings using the proposed approach compared with baseline methods. Thus, this work can be considered as a hybridization of [104] that uses GPs for short-term traffic flow prediction and [92] that uses the idea of clustering similarly behaved data to train local GPs.

In addition, I model traffic speed as spatiotemoporal GPs on road networks, by taking advantage of the expressiveness of the GP kernel functions. Such expressiveness allows us to model the topology and directedness of the road network, as demonstrated by Yu and Chu [105]. I further take advantage of the *additive kernel* feature of GPs [27] to incorporate *side information*[2], where side information can be any spatial feature of the road network that affects traffic speed through it. Through extensive experiments, I show that there exists an intrinsic tradeoff between model expressiveness and computational efficiency. Model expressiveness translates into more accurate predictions at the cost of increased runtime. In practice, one needs to consider carefully such tradeoff and chooses the most relevant side information to the traffic phenomenon being modeled.

6.2 Problem Statement

A city's *road network* is a system of interconnected segments and points that represents the land transportation network of a given urban area. A road network can thus be

[2]Strictly speaking, side information is defined as "the data that are neither from the input space nor from the output space of the function, but include useful information for learning it" [47].

TABLE 6.1: Summary of additional notations used in this chapter. Reintroduced notations override those introduced earlier.

Notation	Description
G, V, E, S	Road network $G = (V, E)$, and subset of segments $S \subset E$ that have traffic sensors installed
$\mathcal{S}, \mathcal{T}$	Set of spatial contexts ($\mathcal{S} \equiv E$) and temporal contexts (e.g., time of the day), respectively
$\mathbf{D}, \mathbf{W}, \mathbf{H}$	Matrix of observed speeds and its factors, where $\mathbf{D} \approx \mathbf{W} \times \mathbf{H}$ ($\mathbf{W}, \mathbf{H}$ are the spatial and temporal cluster, respectively)
N, M, K	Dimensions of $\mathbf{D}$ ($N \times M$), $\mathbf{W}$ ($N \times K$) and $\mathbf{H}$ ($K \times M$), where $N = \lvert S \rvert$ and $M = \lvert \mathcal{T} \rvert$
$\mathbf{Q}$	Set of traffic speed queries: $\mathbf{Q} = \{(r, t)\}$, where $r \in E$ and $t \in \mathcal{T}$
$\mathbf{X}$	Space of spatiotemporal contexts: $\mathbf{X} = \mathcal{S} \times \mathcal{T}$
$\mathbf{Y}$	Observed speeds in $\mathbf{D}$, i.e., $\mathbf{D} = (y_{ij})$
S_i, T_j	Spatial and temporal cluster label ($1 \leq i, j \leq K$)
k, $K(\mathbf{X}, \mathbf{X})$	GP kernel function and covariance matrix
$\mathbf{f}_u$, $\mathbf{f}_{(u,v)}$	Side information: node-wise ($\mathbf{f}_u$) and edge-wise ($\mathbf{f}_{(u,v)}$)
Δ, W	Temporal interval and sliding window

naturally modeled using a graph data structure $G = (V, E)$, where the set of edges E represents the road segments and the set of nodes V represents the intersections (points) among those segments. For many cities around the world, detailed road networks are often made publicly available (typically as GIS shapefiles) by the city's transportation authorities. Moreover, these shapefiles typically contain useful information about the road features such as speed limits, number of lanes, segment length, road type, etc.

Suppose I have a road network G and a subset $S \subset E$ of road segments is installed with some form of traffic sensors. Suppose I also have recent observations $\mathbf{D}$ of vehicular travel speeds measured by those sensors at a certain temporal granularity level Δ (i.e., the sampling interval) along the segments in S. Let $r \in E$ be a road segment and $\vec{v}_r$ be the observed speed over r, which is inherently a directional quantity.

Given $\mathbf{D}$ and a set $Q \subseteq E$ of querying segments, I seek to answer the questions:

1. What are the expected traffic speeds along the segments in Q *not* covered by traffic sensors at the current time? I call this the *spatial inference* task.

2. What are the expected traffic speeds along *all* the segments in Q in the *near future*[3]? I call this the *temporal prediction* task.

[3] "Near future" or "short-term" prediction is subjectively defined in this chapter as less than 10 sampling intervals.

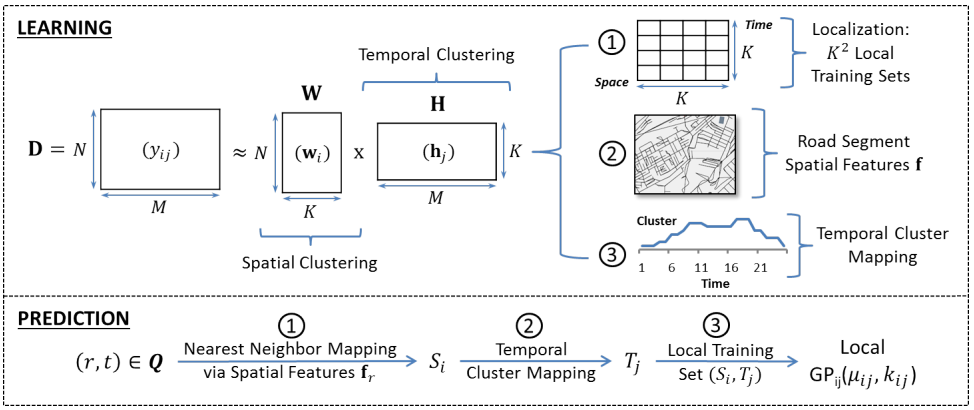

FIGURE 6.1: Framework for efficient spatiotemporal inference of traffic speed using non-negative matrix factorization (NMF) and local Gaussian processes (GPs).

The spatial inference task arises because the spatial coverage of traffic sensors in a city's road network is typically sparse, which may be attributed to their high installation and maintenance costs [18, 89]. The short-term temporal prediction task arises from many real-world applications such as real-time vehicle routing, where new routes are continuously being calculated in light of current and predicted traffic speed information [65, 104]. Thus, having answers to these questions are the necessary conditions for the solutions to many real-world problems in urban settings, where accurate and fine-grained prediction of the city's spatiotemporally varying traffic speed is crucial.

Table 6.1 summarizes the additional notations used in this chapter as well as their relations. Note that reintroduced notations override those introduced in Table 3.1.

6.3 Solution Overview

I address the efficiency issues of using spatiotemporal GPs for learning and predicting large-scale speed data. I draw inspiration from Tobler's first law of geography—"Everything is related to everything else, but *near* things are *more related* than distant things" [98] to cluster the recently observed traffic speeds in both space and time into "local" sets of training data. Each of those subsets corresponds to a local GP. I call such clustering *localization* for short.

Let $\mathbf{Q} = \{(r, t)\}$ be a set of querying road segments at a future time t. For each segment $r \in \mathbf{Q}$, I just need to learn a local GP using the segments "near to" r *w.r.t. the observed speeds* in order to make a good enough inference of r. Likewise, given a future time t, I

just need to know the data points that are "related to" t (w.r.t. the speed) in order to predict those at t. I use clustering to quantify such nearness and relatedness in space and time. I propose to use non-negative matrix factorization (NMF) for localization as spatiotemporal clustering is naturally obtained through factorizing the matrix of observed speeds $\mathbf{D}$. The meaning of "local" here is the subset of segments and time points in $\mathbf{D}$ that are assumed to have similar speeds to (r, t).

The gain in efficiency comes from the use of a much smaller subset of training data for each local GP. In addition, using *more relevant* training data could even improve prediction as will be demonstrated. Fig. 6.1 illustrates the proposed framework for efficient spatiotemporal inferences for big traffic data using local GPs. The framework consists of two components: *learning* and *prediction*.

Learning. Let $\mathbf{D} = (y_{ij})$ be a matrix of dimension $N \times M$, where y_{ij} is an observed speed value along segment i at time discrete time step j, $N = |S|$ is the total number of road segments, and $M = |\mathcal{T}|$ is the total number of regular intervals sampled per day by traffic sensors. The learning process consists of three steps:

Step 1 I factorize $\mathbf{D}$ into matrices $\mathbf{W} \in \mathbb{R}_{\geq 0}^{N \times K}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{K \times M}$, where $K \ll N, M$. I call K the number of spatial/temporal clusters of $\mathbf{D}$. That is, I could divide the road segments in S into K spatial clusters of similar traffic patterns throughout $\mathcal{T}$ and, likewise, I could divide $\mathcal{T}$ into K temporal clusters of similar traffic patterns throughout S. Thus, there are K^2 such spatiotemporal clusters, each corresponding to a local training set of a local GP.

Step 2 I normalize $\mathbf{W}$ row-wise. For each row $\mathbf{w}_i$ $(1 \leq i \leq N)$ of $\mathbf{W}$ that corresponds to a road segment r_i, I probabilistically assign r_i to one of K spatial clusters using the probability vector $\mathbf{w}_i$. Each r_i also has a vector of spatial features $\mathbf{f}_i$ that is used for spatial clustering mapping.

Step 3 I normalize $\mathbf{H}$ column-wise. For each column $\mathbf{h}_j$ $(1 \leq j \leq M)$ of $\mathbf{H}$ that corresponds to a time step t_j, I probabilistically assign t_j to one of K temporal clusters using the probability vector $\mathbf{h}_j$. I call this step *temporal cluster mapping*.

Step 2 and 3 perform "soft assignment" of each road segment and time interval to their respective cluster member. In this respect, NMF is essentially analogous to performing

simultaneous clustering on the rows and columns of $\mathbf{D}$, and probabilistically assigning each row and column vector of $\mathbf{D}$ to their respective cluster member. Because the rows of $\mathbf{D}$ represent the observed traffic patterns over $\mathcal{T}$ at specific road segments, I interpret Step 2 as *spatial clustering* of road segments according to the similarities of traffic patterns over time. Likewise, each column of $\mathbf{D}$ represents the observed traffic pattern over $S \subset \mathcal{S}$ at certain time interval. Therefore, Step 3 can be interpreted as *temporal clustering* of time intervals according their similarities of traffic patterns over space. Because the same K is used for both spatial and temporal clustering, I conceptualize such localization as binning the training data $\mathbf{D}$ into $K \times K$ partitions, where each of the partitions is a "local" set of training data that have similar traffic pattern in space and time. This concept is illustrated in Step 1 of Fig. 6.1.

Prediction. Given a query pair $(r, t) \in \mathbf{Q}$, where t is some future time, prediction involves the following steps:

Step 1 I compare the spatial feature vector $\mathbf{f}_r$ of r with each $\mathbf{f}_s$ of s, $\forall s \in S$ using the Euclidean distance. I choose the nearest segment $s^* \in S$ to r. From Step 2 in Learning, I know which spatial cluster s^* belongs to, here denoted as S_i $(1 \leq i \leq K)$. I deterministically assign r to S_i. I call this step *nearest neighbor mapping*.

Step 2 Given $t \in \mathcal{T}$, I simply look up which temporal cluster label T_j $(1 \leq j \leq K)$ it belongs to using the temporal cluster mapping (derived in Step 3 of Learning) and deterministically assign t to T_j.

Step 3 Given the cluster labels S_i and T_j of (r, t), I retrieve the corresponding local training set (S_i, T_j), train the local GP(i, j) model and make a spatiotemporal inference for (r, t).

For convenience, I shall hereafter use the term "spatiotemporal inference" to collectively refer to both the spatial inference (of unobserved segments) and the temporal prediction (of future traffic speed). Each local GP(i, j) can be further extended to consider the network structure and topology in its spatial "locality", as well as incorporate side information of the road segments via the its kernel function (see Section 6.5). I shall also use the term "*global* GP" to refer to the GP model whose training set is sampled uniformly at random from $\mathbf{D}$ without localization.

6.4 Non-negative Matrix Factorization for Localization

6.4.1 Preliminaries

Non-negative matrix factorization (NMF) is a popular technique for decomposing data into latent (hidden) components with physical meaning and interpretations [23, 59]. It has been widely used in dimensionality reduction, object detection, latent clustering, and blind source separation, involving image, text and signal data [23, 90, 95]. In this work, I use NMF to decompose matrix $\mathbf{D}$ into two non-negative matrices $\mathbf{W}$ and $\mathbf{H}$ that represent the spatial and temporal clusters of speed values in $\mathbf{D}$, respectively. These two matrices are then used for the localization of GPs during training and prediction.

NMF seeks to approximate $\mathbf{D} \in \mathbb{R}_{\geq 0}^{N \times M}$ by a product of $\mathbf{W} \in \mathbb{R}_{\geq 0}^{N \times K}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{K \times M}$ (i.e., $\mathbf{D} \approx \mathbf{W} \times \mathbf{H}$), where K is the number of clusters. Note that usually $K \ll \min(N, M)$. The non-negativity constraint imposed on the two matrices serves to provide meaningful interpretations for the spatial and temporal clusters. That is, each row of $\mathbf{W}$ can be interpreted as the *degrees of membership* to K different spatial clusters. Likewise, each column of $\mathbf{H}$ represents the degrees of membership to K different temporal clusters.

6.4.2 Optimization Objective

The quality of approximating $\mathbf{D}$ by $\mathbf{W} \times \mathbf{H}$ can be measured through various distance functions. In this work, I use the Frobenius norm, which leads to the optimization problem of *minimizing* the loss function $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{2}||\mathbf{D} - \mathbf{WH}||_F^2 = \frac{1}{2}\sum_{i,j}\left[\mathbf{D}_{i,j} - \sum_k \mathbf{W}_{i,k}\mathbf{H}_{k,j}\right]^2 \tag{6.1}$$

where $i \in \{1, \ldots, N\}$, $j \in \{1, \ldots, M\}$, and $k \in \{1, \ldots, K\}$.

To arrive at meaningful spatial and temporal clusters, I further impose *sparsity* constraints to $\mathbf{W}$ and $\mathbf{H}$ via L1-norm penalty. This yields the following regularized loss:

$$\mathcal{L} = \frac{1}{2}||\mathbf{D} - \mathbf{WH}||_F^2 + \lambda \underbrace{\left(\sum_{i,k}\mathbf{W}_{i,k} + \sum_{j,k}\mathbf{H}_{k,j}\right)}_{\text{L1-norm penalty}}, \tag{6.2}$$

where $\lambda > 0$ is the regularization parameter (set to $\lambda = 100$). Enforcing sparse $\mathbf{W}$ and $\mathbf{H}$ leads to sparse membership to different clusters, thus improving the model interpretability while retaining approximation quality.

It is also worth noting that $\mathcal{L}$ is convex with respect to the individual matrix $\mathbf{W}$ or $\mathbf{H}$, but not both. As a result, one can only expect to find a stationary point of $\mathcal{L}$, which is not necessarily a globally optimal solution. I next describe a fast coordinate descent algorithm to find a stationary solution to the optimization problem (6.2).

6.4.3 Coordinate Descent Learning

The key idea of the coordinate descent (CD) method is to update one variable at a time, while keeping the others fixed. The efficiency of the CD procedure has been demonstrated in several state-of-the-art machine learning methods [28, 34]. For NMF, the conventional ways of learning $\mathbf{W}$ and $\mathbf{H}$ are largely based on the alternative non-negative least squares (ANLS) framework [77], which converges to stationary points provided each sub-problem can be solved exactly. However, the ANLS-based methods usually take a significant amount of time to find an exact solution for each sub-problem. In contrast, the CD method can efficiently compute reasonably good solution for each sub-problem and move on to the next round [34].

Without loss of generality, I shall focus on the coordinate descent update for entries in $\mathbf{W}$; the update for entries in $\mathbf{H}$ can be similarly derived, i.e., by replacing $\mathbf{D}$ with $\mathbf{D}^\top$ and swapping $\mathbf{W}$ with $\mathbf{H}^\top$. The CD method solves each sub-problem by the following one-variable Newton update:

$$\mathbf{W}_{i,k} \leftarrow \max\left(0, \mathbf{W}_{i,k} - \frac{(\bigtriangledown_{\mathbf{W}}\mathcal{L})_{i,k}}{(\bigtriangledown^2_{\mathbf{W}}\mathcal{L})_{i,k}}\right), \tag{6.3}$$

where $\bigtriangledown$ and $\bigtriangledown^2$ denote the gradient (i.e., first derivative) and curvature (i.e., second derivative), respectively. The truncation $\max(0, x)$ serves to ensure non-negative $\mathbf{W}$.

With respect to the regularized loss (6.2), it is easy to show that the gradient $\bigtriangledown_{\mathbf{W}}\mathcal{L}$ resolves to:

$$\bigtriangledown_{\mathbf{W}}\mathcal{L} = \mathbf{W}\mathbf{H}\mathbf{H}^\top - \mathbf{D}\mathbf{H}^\top + \lambda, \tag{6.4}$$

and in turn the curvature $\bigtriangledown^2_{\mathbf{W}}\mathcal{L}$ is:

$$\bigtriangledown^2_{\mathbf{W}}\mathcal{L} = \mathbf{H}\mathbf{H}^\top. \tag{6.5}$$

Consequently, the CD update in (6.3) can be written as:

$$\mathbf{W}_{i,k} \leftarrow \max\left(0, \mathbf{W}_{i,k} - \frac{(\mathbf{W}\mathbf{H}\mathbf{H}^\top - \mathbf{D}\mathbf{H}^\top)_{i,k} + \lambda}{(\mathbf{H}\mathbf{H}^\top)_{i,k}}\right). \tag{6.6}$$

It can be seen from (6.6) that the regularization parameter λ plays a role in shifting the new $\mathbf{W}_{i,k}$ to a smaller (possibly negative) value. As such, a larger λ would foster more (zero) truncation and therefore result in a sparser solution.

Using the update rule (6.6), I carry out a *cyclic* coordinate descent. That is, I first update all entries in $\mathbf{W}$ in cyclic order, and then update entries in $\mathbf{H}$, and so on. With respect to $\mathbf{W}$, I traverse every cluster k, in which I update each variable $\mathbf{W}_{i,k}$ using (6.6). The same applies to each $\mathbf{H}_{k,j}$, with $\mathbf{W}$ swapped with $\mathbf{H}^\top$. The procedure is repeated until a maximum number of iterations (set to 200) is reached.

6.4.4 Efficiency Considerations

The aforementioned CD procedure can be carried out efficiently if certain quantities are pre-computed. Specifically, I calculate and store the matrix products $\mathbf{D}\mathbf{H}^\top$ and $\mathbf{H}\mathbf{H}^\top$ prior to entering the one-variable update loop for $\mathbf{W}$. (Similarly, I pre-compute $\mathbf{D}^\top\mathbf{W}$ and $\mathbf{W}^\top\mathbf{W}$ before updating $\mathbf{H}$). These would incur an additional memory with an order of $\mathcal{O}\left((N+M)\times K\right)$ and $\mathcal{O}\left(K^2\right)$, respectively. As such, the total memory complexity of the CD procedure is $\mathcal{O}\left((N+M+K)\times K\right)$. This, however, is still much smaller than the dimensionality of $\mathbf{D}$ (i.e., $N \times M$).

Meanwhile, thanks to caching, the time complexity of the CD procedure is *linear* with respect to N and M. In particular, the time needed to update all entries in $\mathbf{W}$ within a CD iteration is $\mathcal{O}\left(N \times K^2\right)$. Similarly, the time for updating $\mathbf{H}$ is $\mathcal{O}\left(M \times K^2\right)$. Thus, the overall time complexity is thus $\mathcal{O}\left((N+M)\times K^2 \times T_{\max}\right)$ (where $T_{\max}$ is the maximum number of iterations). As K and $T_{\max}$ are typically small, fixed values that are independent of the problem size, I conclude that the CD procedure is efficient. I empirically demonstrate its efficiency in Section 6.6.4.

6.4.5 Determining K

One practical problem in applying NMF is to determine the optimal number of clusters K. I use 10-fold cross validation (CV) procedure to determine K. Specifically, I randomly split all entries y_{ij} of $\mathbf{D}$ into 10 mutually exclusive folds, and for each CV iteration f, I use fold f as validation set for NMF, and the remaining (nine) folds as training set. I determine the optimal number of clusters by choosing K that gives the highest *fraction of explained variance* score [26] averaged over 10 validation sets.

For a target (speed) variable y and predicted (speed) variable $\hat{y}$, the fraction of expected variance $R^2(y, \hat{y})$ is:

$$R^2(y, \hat{y}) = 1 - \frac{Var[y - \hat{y}]}{Var[y]}, \tag{6.7}$$

where $Var[y] = \mathbf{E}[y^2] - (\mathbf{E}[y])^2$ is the variance of y.

Notably, the fraction of explained variance is a popular metric commonly used to evaluate a regression model [26]. For an optimal regression model $\hat{y}$ that perfectly matches the target variable y, the variance $Var[y-\hat{y}]$ will be zero, which in turn implies $R^2(y, \hat{y}) = 1$. On the other hand, the most naïve regression model is a constant function, which gives $Var[y-\hat{y}] = Var[y]$ and thus $R^2(y, \hat{y}) = 0$. In this case, the prediction $\hat{y}$ tells us nothing about the target y, in the sense that $\hat{y}$ does not covary with y.

6.5 Spatiotemporal Gaussian Processes for Traffic Speed

6.5.1 Preliminaries

Let $\mathcal{S}$ denote the space of spatial contexts (i.e., $\mathcal{S} \equiv E$ in this chapter) and $\mathcal{T}$ denote the space of temporal contexts (e.g., information about time of the day). I model the speed over road segment $r \in E$ under varying $t \in \mathcal{T}$ via the function $f : \mathcal{S} \times \mathcal{T} \mapsto \mathbb{R}_{\geq 0}$ that outputs a non-negative speed value for a given (r, t) pair.

I define a spacetime process as a stochastic process indexed by road segments $r \in \mathcal{S}$ and temporal labels $t \in \mathcal{T}$:

$$\{f(r, t) : r \in \mathcal{S}, t \in \mathcal{T}\}. \tag{6.8}$$

Thus, for a fixed spacetime location (r, t), $f(r, t)$ is a random variable. It is a fundamental nature of spatiotemporal data that observations at *nearby* locations in space and time are similar [83]. I need a mathematical model to quantify the extent to which things are related over space and time. Kernel functions provide such an elegant model. For example, given two spacetime locations (r, t) and (r', t'), the radial basis function (RBF) kernel has the following form:

$$k((r, t), (r', t')) = e^{-\|(r,t)-(r',t')\|/l^2}. \tag{6.9}$$

A spatiotemporal Gaussian process (GP) is a stochastic process over an index set $\mathbf{X} = \mathcal{S} \times \mathcal{T}$. It is entirely defined by a mean function $\mu : \mathbf{X} \mapsto \mathbb{R}_{\geq 0}$ and a covariance (kernel) function $k : \mathbf{X} \times \mathbf{X} \mapsto \mathbb{R}$. These two functions are chosen such that they jointly define a multivariate normal distribution whenever I draw $f|\mathbf{X}$ from a $\mathrm{GP}(\mu, k)$ on a finite set of spacetime locations $\mathbf{X} = \{x_1, \ldots, x_T\}$:

$$f|\mathbf{X} \sim \mathcal{N}(\mu(\mathbf{X}), K(\mathbf{X}, \mathbf{X})), \tag{6.10}$$

where $\mu(\mathbf{X})_i = \mu(x_i)$ and $[K(\mathbf{X}, \mathbf{X})]_{ij} = k(x_i, x_j)$.

By this construction, $\mu(\mathbf{X})$ is a T-dimensional non-negative vector and $K(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{T \times T}$ is a positive semidefinite covariance matrix. I now assume that f is sampled probabilistically from a GP prior $f \sim P(f)$ [83]. A GP prior is fully specified by its mean function:

$$\mu(r, t) = \mathbb{E}[f(r, t)],$$

its covariance (or kernel) function:

$$\begin{aligned} k((r, t), (r', t')) &= \mathbb{E}[(f(r, t) - \mu(r, t))(f(r', t') - \mu(r', t'))] \\ &= \mathrm{Cov}((r, t), (r', t')), \end{aligned}$$

and observation noise with variance σ^2.

A major computational benefit of GPs is that the posterior can be computed in a closed form. Suppose I have collected recent speed observations $\mathbf{Y} = [y_1, \ldots, y_T]^\top$ at $\mathbf{X} = [(r_1, t_1), \ldots, (r_T, t_T)]$. I can write the posterior distribution of f given $\mathbf{X}$ and $\mathbf{Y}$

also as a GP with mean:

$$\mu_{\mathbf{Y},\mathbf{X}}(r,t) = \mu(r,t) + \hat{k}_{\mathbf{X}}(r,t)^{\top}(\hat{K}_{\mathbf{Y},\mathbf{X}} + \sigma^2 I)^{-1}(\delta\mathbf{Y})^{\top} \tag{6.11}$$

and covariance $k_{\mathbf{Y},\mathbf{X}}((r,t),(r',t')) =$

$$k((r,t),(r',t')) - \hat{k}_{\mathbf{X}}(r,t)^{\top}(\hat{K}_{\mathbf{X}} + \sigma^2 I)^{-1}\hat{k}_{\mathbf{X}}(r',t'), \tag{6.12}$$

where $\delta\mathbf{Y}$ is the deviation of $\mathbf{Y}$ from its prior mean:

$$\delta\mathbf{Y} = [y_1 - \mu(r_1,t_1), \ldots, y_T - \mu(r_T,t_T)]^{\top},$$

$\hat{k}_{\mathbf{X}}(r,t)$ is a column vector of the kernel values between (r,t) and each observed location in $\mathbf{X}$:

$$\hat{k}_{\mathbf{X}}(r,t) = [k((r_1,t_1),(r,t)), \ldots, k((r_T,t_T),(r,t))]^{\top} \in \mathbb{R}^T,$$

and $\hat{K}_{\mathbf{X}}$ is the Gram matrix of all locations in $\mathbf{X}$:

$$\hat{K}_{\mathbf{X}} = [k((r_i,t_i),(r_j,t_j))]_{i,j\in[1,\ldots,T]} \in \mathbb{R}^{T\times T}.$$

The posterior variance of $f(r,t)$ is $k_{\mathbf{Y},\mathbf{X}}((r,t),(r,t))$.

Inference of continuous values with GP prior is known as GP regression (or *kriging*). When concerned with a general GP regression, it is assumed that for a GP f observed at location (r,t), $f(r,t)|\Theta$ is just one sample from the multivariate normal distribution of dimension $|\mathbf{X}|$, where Θ is the set of hyper-parameters of the kernel function $k((r,t),(r',t'))$. Thanks to its non-parametric nature, training a GP reduces to estimating Θ via the marginal likelihood function. Having identifying Θ, spatiotemporal inference $f(r',t')$ becomes a matter of sampling from the posterior distribution. A major computational bottleneck of GP is its $\mathcal{O}(|\mathbf{X}|^3)$ time complexity, which makes it impractical for large-scale spatiotemporal data [20, 67, 83].

6.5.2 Kernel Functions for Road Networks

Let $G = (V,E)$ be a directed graph representing a road network. G is directed because traffic on a road segment could possibly be one-way. On two-way segments, the corresponding links of G become bidirectional. Let $\mathbf{Y} = \{y_{(u,v)} : (u,v) \in E\}$ be the

speed values that I wish to model. An important nature of networks is that $\mathbf{Y}$ are highly correlated on known node and edge features. Following Yu and Chu [105], let $f : V \times V \mapsto \mathbb{R}_{\geq 0}$ be a GP(μ, k), then the kernel function between (u, v) and (u', v') can be written as:

$$k((u, v), (u', v')) = k(u, u')k(v, v'), \tag{6.13}$$

where $k : V \times V \mapsto \mathbb{R}$ is some kernel function between the nodes. Since a random function f drawn from GP(μ, k) is generally asymmetric, i.e., $f(u, v) \neq f(v, u)$, traffic directions along the links in G are automatically modeled.

Let $u, v \in V$ be identified by their respective pair of longitude and latitude coordinates (u_x, u_y) and (v_x, v_y), then equation (6.13) becomes:

$$\begin{aligned} &k((u, v), (u', v')) \\ &= k((u_x, u_y), (u'_x, u'_y))k((v_x, v_y), (v'_x, v'_y)). \end{aligned} \tag{6.14}$$

For spatiotemporal data, a natural way to formulate a spacetime kernel is to multiply the spatial kernel k_s and the temporal kernel k_t together. This feature is referred to as *separable kernel* of GPs [67, 83]. Let $r = (u, v), r' = (u', v') \in E$ and t be a time label, from (6.14), I have:

$$\begin{aligned} &k((r, t), (r', t')) \\ &= k_s((u_x, u_y), (u'_x, u'_y))k_s((v_x, v_y), (v'_x, v'_y))k_t(t, t'). \end{aligned} \tag{6.15}$$

6.5.3 Incorporating Side Information

I define *side information* as any spatial features of the nodes and edges of G other than the longitude and latitude coordinates of the nodes of G, which precisely specify the geolocation of a given edge (u, v) and quantify its geospatial nearness to another edge (u', v'). Therefore, side information could be any *other* spatial features of the nodes and edges of G that can be derived from the given GIS shapefile of the road network. I then classify side information into two types: node-wise and edge-wise side information, where *node-wise* side information contains the spatial features of the nodes of G and *edge-wise* side information contains the spatial features of the edges of G.

For each road segment $r = (u, v)$, let $\mathbf{f}_u$ and $\mathbf{f}_v$ denote the vectors of node-wise side information of r, which are necessarily of the same length. Likewise, let $\mathbf{f}_{(u,v)}$ denote the vector of edge-wise side information of r. The set of *all* side information of r is denoted as $\mathbf{f}_r = (\mathbf{f}_u; \mathbf{f}_v; \mathbf{f}_{(u,v)})$. I take advantage of the *additive kernel* feature of GPs [27] to incorporate side information into the kernel function. Following (7.1), the kernel function between (r, t) and (r', t') knowing their side information $\mathbf{f}_r$ and $\mathbf{f}_{r'}$ is given by:

$$\begin{aligned} &k((r, t, \mathbf{f}_r), (r', t', \mathbf{f}_{r'})) \\ &= k((r, t), (r', t')) + \sum_i k(\mathbf{f}_u^{(i)}, \mathbf{f}_{u'}^{(i)}) k(\mathbf{f}_v^{(i)}, \mathbf{f}_{v'}^{(i)}) \\ &+ \sum_j k(\mathbf{f}_{(u,v)}^{(j)}, \mathbf{f}_{(u',v')}^{(j)}), \end{aligned} \tag{6.16}$$

where i and j are the indices of the set of node-wise and edge-wise side information, respectively.

6.5.4 Complexity of Local GPs

For each local GP(i, j), without incorporating side information, the time complexity is $\mathcal{O}(|\mathbf{X}_{ij}|^3) = \mathcal{O}(|\mathcal{S}_i|^3 \times |\mathcal{T}|_j^3)$. The original sizes of $S \subset E$ and the space of temporal contexts $\mathcal{T}$ from matrix $\mathbf{D}$ are N and M, respectively. Due to clustering, each local training set (S_i, T_j) has $\mathbb{E}[|\mathcal{S}_i|] = N/K$ and $\mathbb{E}[|\mathcal{T}_i|] = M/K$ training data points on expectation. Thus, the expected time complexity of each local GP(i, j) is $\mathcal{O}((\frac{NM}{K^2})^3)$. If the prediction phase in Fig. 6.1 can be done in parallel for each spatiotemporal cluster (S_i, T_j), then $\mathcal{O}((\frac{NM}{K^2})^3)$ is the expected time complexity to predict an arbitrary set of queries $\mathbf{Q} = \{(r, t)\}$. Otherwise, if it is done serially, then the worst-case time complexity is $K^2\mathcal{O}((\frac{NM}{K^2})^3) = \mathcal{O}(K^2(\frac{NM}{K^2})^3) = \mathcal{O}(\frac{(NM)^3}{K^4})$, which is still a significant improvement over the original $\mathcal{O}((NM)^3)$ time complexity of global GPs without side information.

For GPs with side information, the total time complexity is added by the complexity of the kernel function of each "piece" of side information, each having complexity of $\mathcal{O}(N^3)$ and $\mathcal{O}((\frac{N}{K^2})^3)$ for global and local GPs, respectively. I will empirically demonstrate in the next section the effects of having side information on the "wall-clock" runtime performances of both local and global GPs.

6.6 Experiments

6.6.1 Datasets

TMC (Traffic Message Channel) is a technology used to broadcast traffic information in real-time to vehicles through the radio waves. TMC allows for silent delivery of dynamic traffic information, and is often integrated directly into the vehicle's navigation system for real-time estimation of speed and route calculation. I have acquired, through a commercial vendor of navigation systems, rich TMC datasets that record the average speeds along certain road segments in the cities of Pittsburgh, Pennsylvania (P.A.), and Washington, D.C. My TMC datasets cover a total of $1,190$ and $1,091$ unique road segments in the city's road network of Pittsburgh and Washington, respectively. Each record is an average speed measurement over a road segment every 5-minute interval (i.e., $\Delta = 5$ minutes) everyday for the whole month[4] of August, 2014. Each speed value also has a direction indicator (e.g., northbound, southbound, eastbound or westbound). Thus, my dataset is a close approximation to the city's traffic sensor network.

TMC technology fuses real-time traffic information from crowd-sourced networks of "floating cars" and mobile devices with public sources of information (e.g., from historical data). Under normal conditions, when no incidents are reported from crowd-sourced devices, TMC data capture publicly available sources of traffic information. Under irregular conditions, such as traffic incidents or congestion, crowd-sourced information is collected and broadcast to alert drivers in real-time. Still, TMC data can be missing for certain road segments when routing services are not usually called for. This happens typically in the late night or early morning hours. Hence, my data are temporally sparse for each road segment, i.e., there are many missing values in the temporal dimension.

I downloaded the shapefiles[5] representing the two cities' road networks and constructed a connected directed graph $G = (V, E)$ for each. My datasets cover approximately 5% and 8% of the city's road network for Pittsburgh and Washington, respectively. I extract useful spatial features of the road segments in G from the retrieved shapefiles and the network structure of G. Table 6.2 summarizes those spatial features. The table also shows two *network centrality* measures of G: (node) degree and (edge) betweenness.

[4]Traffic pattern typically remains the same during a season [81], which justifies my choice of data.

[5]The shapefile of Pittsburgh's road network can be downloaded from: http://pittsburghpa.gov/dcp/gis/gis-data-new, and Washington's from: http://opendata.dc.gov.

FIGURE 6.2: Speed distribution along road segments covered by my TMC dataset in downtown Pittsburgh on a typical weekday in August, 2014 at 8 a.m.

TABLE 6.2: The extracted features **f** of the road segments in Pittsburgh & Washington.

Feature	Description
Longitude, latitude	Longitude and latitude coordinates of the two endpoints (nodes) of a segment.
Segment length	Length (in miles) of a segment.
Number of lanes	The number of lanes a segment has in each direction.
Direction	Direction of a segment: northbound, southbound, eastbound, or westbound.
Degree	Degree of two end nodes of an edge (segment).
Betweenness	Edge betweenness centrality of a segment.
One-way	Is this segment one-way?
Road type	One of the 10 defined types: avenue, boulevard, bridge, lane, place, ramp, road, street, tunnel, and way.

Node degree is the (all) degree of a node in the directed network. Edge betweenness is the number of shortest paths from all pairs of nodes in the network that pass through a given edge [13]. Network centralities have been shown to greatly influence on the flow of information and traffic through diverse networked settings [13, 42, 58].

Fig. 6.2 visualizes the speed distribution over the road segments covered by my TMC data in downtown Pittsburgh on a typical weekday. Speed value along a segment is averaged over observations on all the weekdays in the month at 8 a.m. The figure shows smaller segments in the downtown area tend to have lower speeds during the morning commute. Whereas, larger segments are observed with higher speeds and faster flows.

Fig. 6.3 shows the time series of the average speed on all observed road segments in Pittsburgh during all the weekdays and weekends in the month. The figure clearly shows that traffic speed on the weekend is, on average, faster and less variable than that on the weekday. It also shows the rush hours effects on the weekday: average speed dips around 8 a.m. (morning rush hour) and 5 p.m. (evening rush hour) when people commute to work and go home, respectively. The traffic between those two rush hours

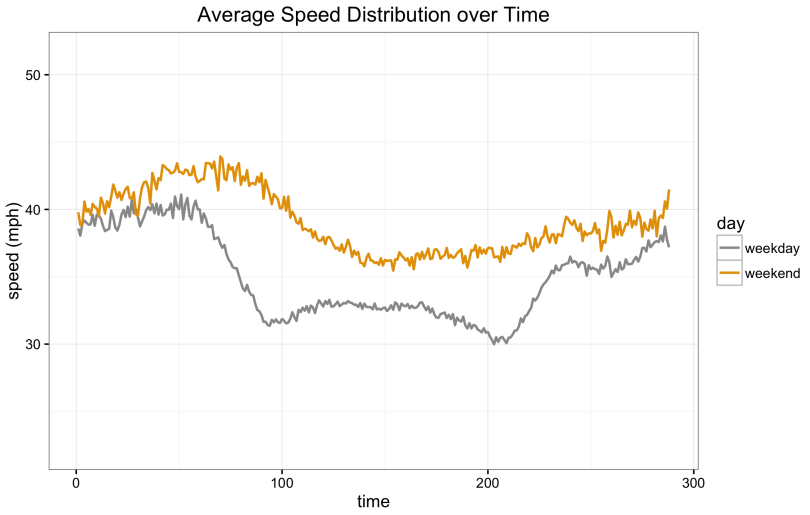

FIGURE 6.3: Time series of the daily average speed along road segments in Pittsburgh every 5-minute interval in August, 2014.

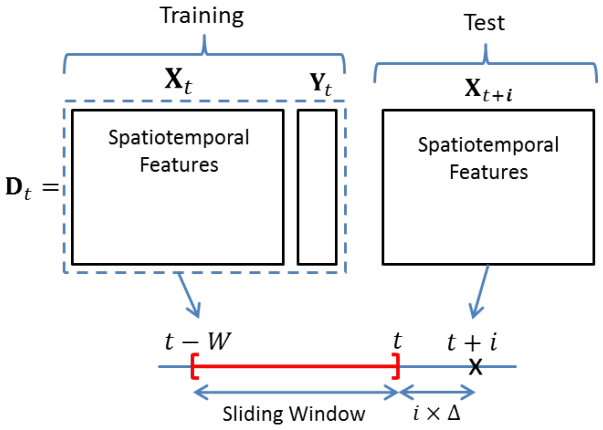

FIGURE 6.4: The "sliding window" experimental design: $t \in \{0, 1, \ldots, 22, 23\}$ on the test day, $t + i$ $(1 \leq i \leq 6)$ denotes the test time. W is the length of the sliding window: 5 days for weekday and 3 days for weekend. $\mathbf{D}_t$ denotes the training data containing the features $\mathbf{X}_t$ and the observed speeds $\mathbf{Y}_t$ in $\mathbf{D}_t$ averaged over 24-hour periods in W.

is generally much slower than in the late evening and early morning. On the weekend, by contrast, traffic is generally slower during the day when people tend to go out. The data of Washington, D.C., exhibit very similar patterns.

6.6.2 Experimental Design

Following the observations in Fig. 6.3 and the established procedures in modeling human mobility patterns in urban areas [29, 46, 104, 110], I split the data of each city into two sets: weekday (Monday through Friday) and weekend (Saturday and Sunday). I design

the following experiments to measure the performances of my local GP models in diverse spatiotemporal settings using both sets.

For each city, I designate Thursday, August 28, 2014 and Sunday, August 31, 2014 as the test weekday and weekend, respectively. I choose Thursday as a test weekday as previous studies have suggested the inherent differences in urban mobility patterns between Friday and the rest of the weekdays [29, 46, 81]. I call either date the *test day*. For each hour $t \in \{0, 1, \ldots, 22, 23\}$ on each test day, I designate the *test time* to be 1–6 intervals ahead of t, i.e., test time is $t + i \times \Delta$, where $\Delta = 5$ minutes and $1 \leq i \leq 6$. There are 24 trials per test day, where each trial predicts 6 test cases. I call each test case an i-step ahead prediction and simply denote the test time as $t + i$.

I adopt the "sliding window" method proposed in [104] to collect the training data for each trial t, denoted as $\mathbf{D}_t$. Given a test time $t+i$, $\mathbf{D}_t$ is the observations collected from time $t-W$ up to (and including) t, where W is the length of the window of observations. For weekday, W is a period of *exactly* 5 previous weekdays, i.e., $W = 5 \times 24 \times 12 = 1,440$ intervals. For weekend, W is a period of *exactly* 3 previous weekend days, i.e., $W = 3 \times 24 \times 12 = 864$ intervals. I choose such W for both sets in order to avoid the "cold start" problem[6] in matrix factorization [59] due to the temporal sparsity problem of my data. For each trial, $\mathbf{D}_t$ is the observed speeds averaged *over the days* in W.

To evaluate the spatiotemporal inferences of my models, I randomly select 40% of the segments out of the total number of segments as the training set and test on *all* the segments. Hence, each $\mathbf{D}_t$ is a (476×288)- and (436×288)-dimensional matrix for Pittsburgh and Washington, respectively. Fig. 6.4 illustrates my experimental design.

The following models are considered in my experiments to evaluate the effectiveness of side information, the efficiency of local GPs and the efficacy of NMF-based localization (as opposed to a naive grid-based approach):

1. **GP** – global GP without side information;
2. **GP**$^+$ – global GP with side information;
3. **LGP** – NMF-based local GP without side information;

[6]The cold start problem invalidates the factorization of $\mathbf{D}$ if there exists either an entire row of column of $\mathbf{D}$ that admits *all* missing values.

TABLE 6.3: Models evaluated in the experiments. X means 'Yes'; blank means 'No'.

Model	Baseline	NMF-based	Side Info	Grid-based
GP	X			
GP$^+$	X		X	
LGP		X		
LGP$^+$		X	X	
LGR	X			X
LGR$^+$	X		X	X

4. **LGP$^+$** – NMF-based local GP with side information;

5. **LGR** – grid-based local GP without side information;

6. **LGR$^+$** – grid-based local GP with side information.

All the above models implement spatiotemporal GPs defined on road networks (as described in Section 6.5.2) and use the RBF kernel functions. I use a global GP (with or without side information) as the baseline for each NMF-based local GP counterpart. For each global GP, exactly $T_{\max} = 600$ observations sampled uniformly at random from $\mathbf{D}_t$ are used as its training set. I heuristically choose such value of $T_{\max}$ based on the observed tradeoff between training time and prediction error. That is, too large $T_{\max}$ would induce impractically long training time for real-time purposes, whereas too small $T_{\max}$ would unacceptably increase the prediction error rate of global GPs (i.e., the under-fitting problem). The training set for each local GP consists of $\min\{T_{\max}, |S_l|\}$ observations sampled uniformly at random from the corresponding local subset S_l induced by the localization of $\mathbf{D}_t$. This is to ensure fairness when comparing prediction accuracies and runtime performances between global GPs and their local counterparts.

I also include two grid-based local GPs whose localizations are based on partitioning each city's road network into uniform spatial grids. Each local GP is learned *only* from the data points belonging to a given grid cell. I then compare each grid-based local GP with its NMF-based counterpart. I set the number of grids (for the grid-based local GPs) as K^2, i.e., the same number of clusters used by the NMF-based local GPs.

Table 6.3 summarizes all the six models evaluated in my experiments. All the spatial features listed in Table 6.2 are used as side information, except for longitude and latitude coordinates, which are used to define the spatiotemporal kernel function. Linear kernel functions are used for categorical variables (direction, one-way, and road type); otherwise, RBF kernels are used.

6.6.3 Evaluation

I use the root mean square error (RMSE), the mean absolute error (MAE), and the mean absolute percentage error (MAPE) to evaluate the models. The three metrics are respectively defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}, \tag{6.17}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|, \tag{6.18}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right|, \tag{6.19}$$

where $\hat{y}_i$ and y_i are the predicted and observed speed over road segment i, respectively, and N is the total number of road segments in the test set.

I also measure the runtime performances by looking at the "wall-clock time" (in seconds) for each model to train and make predictions at test time. This includes, whenever possible, matrix factorization, temporal cluster mapping, nearest neighbor mapping, and training and prediction time for each GP model. All the experiments were run on a CentOS Linux machine with 7-core Intel(R) Xeon 2.6 GHz processor and 70 GB RAM.

To evaluate the significance of the improvements due to local GPs, I use the non-parametric Wilcoxon signed-rank statistical test [102], which provides a robust alternative to the pairwise t-test when the measures cannot be assumed to be normally distributed.

6.6.4 Localization

Following the procedure described in Section 6.4.5, I find the optimal number of clusters K^* by taking K that gives the highest explained variance R^2. To this end, I perform 10-fold CV with K varying from 1 to 10, and then look for the "elbow" point that corresponds to the highest R^2 averaged over 10 folds. Fig. 6.5 shows the results. I see that the optimal K^* (i.e., the "elbow") for Pittsburgh are 5 for weekday and 2 for weekend. The optimal K^* for Washington are 3 for weekday and 2 for weekend.

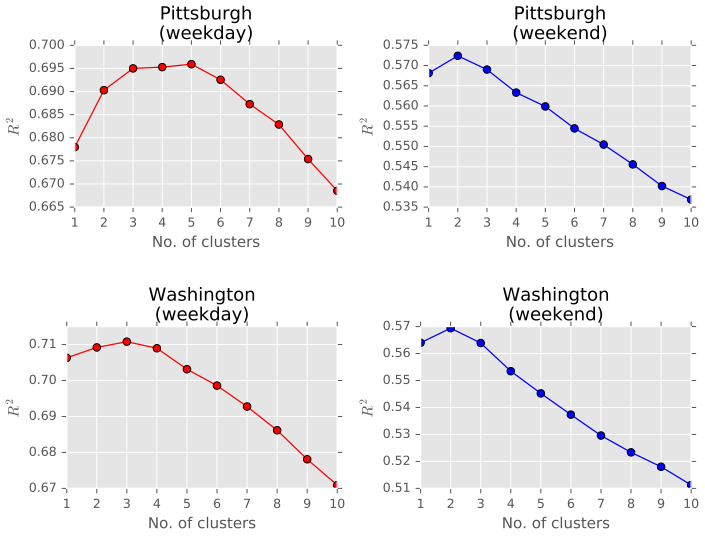

FIGURE 6.5: Results of parameter search procedure for determining K for each dataset.

The higher K^* for weekday suggests that the traffic patterns on the weekday are more complex than those on the weekend.

To verify the convergence of the CD algorithm, I also monitor the residual error $||\mathbf{D} - \mathbf{WH}||^2$ (i.e., the first term in equation (6.2)) over different training iterations. Fig. 6.6 shows the convergence plots of $||\mathbf{D} - \mathbf{WH}||^2$ for different datasets. Here, I zoom into the first 30 training iterations (out of a total of 200 iterations as per Section 6.4.3) in order to see more clearly the convergence of $||\mathbf{D} - \mathbf{WH}||^2$. Indeed, $||\mathbf{D} - \mathbf{WH}||^2$ converges rapidly within 10 iterations and no longer decreases substantially afterwards. This shows that the CD algorithm offers an efficient method for training NMF.

Fig. 6.7 illustrates the time series of the average speed along the clusters of road segments every 5-minute interval on a typical weekday in Pittsburgh. My NMF method has clustered the road segments into different types, each having different throughput and daily speed distribution. For example, for clusters 2, 3, and 4, I can see clearly the rush hour effects observed earlier in Fig. 6.3 to different levels. These clusters mostly contain road segments leading to (and away from) the business areas in the downtown. The other clusters with slower speeds contain mostly small segments in the residential areas, or those that are in the business areas but do not lead to the residential areas.

Fig. 6.8 presents a heatmap visualization of the temporal cluster mapping derived from the column-wise normalized matrix $\mathbf{H}$ on a typical weekday in Pittsburgh. The result shows clear temporal patterns of the traffic speed in the city, whereby the probabilistic

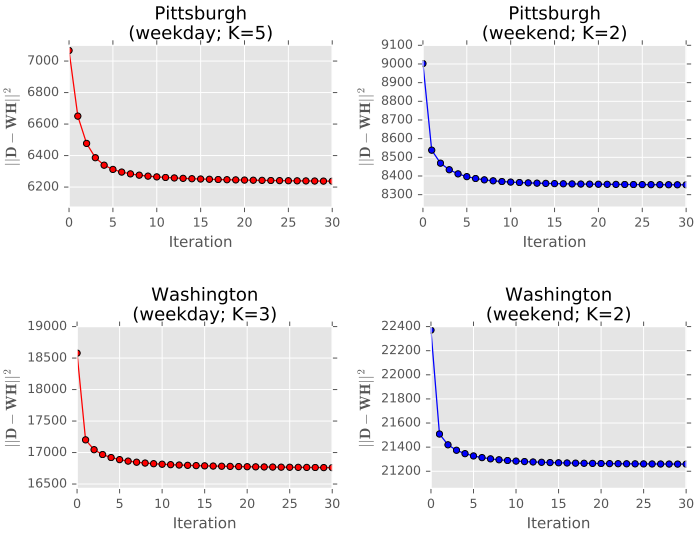

FIGURE 6.6: Convergence of the coordinate-descent training in NMF using the best number of clusters K^* for each dataset.

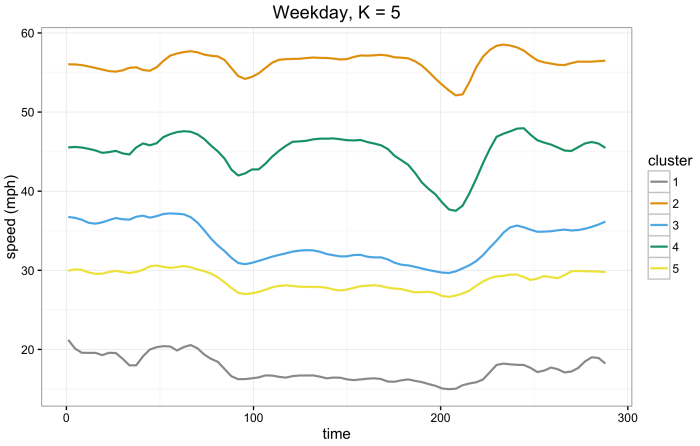

FIGURE 6.7: Time series of the average speed along clustered road segments for weekday data in Pittsburgh. Horizontal axis shows the 5-minute intervals.

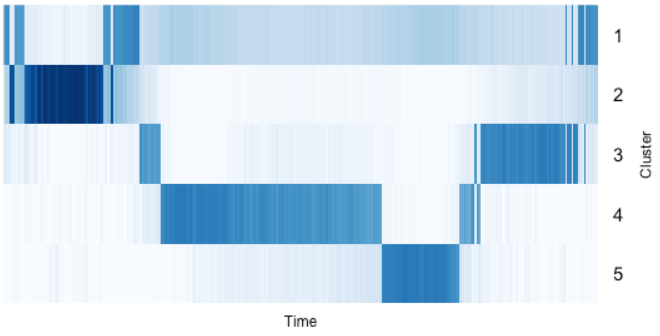

FIGURE 6.8: Heat map of the column-wise normalized matrix $\mathbf{H}$ visualizing the temporal clustering for weekday in Pittsburgh. Bolder shades $\rightarrow 1$ and lighter $\rightarrow 0$.

TABLE 6.4: Runtime statistics (in seconds) of NMF-based localization for Pittsburgh (PGH) and Washington (WAS) on weekday (WD) and weekend (WE).

City (day)	Mean	Median	Stdev
PGH (WD)	0.3137	0.2676	0.1243
PGH (WE)	0.1889	0.1621	0.1043
WAS (WD)	0.2597	0.2050	0.1071
WAS (WE)	0.1395	0.1146	0.0490

assignment of the temporal clusters is sparse. That is, at a given time step, only a few clusters (darker shades) have substantially higher probability value than the rest (lighter shades). In this case, I can identify rush hours by looking at rapidly changing cluster assignments that occur within a fairly short period of time.

Also, because of the temporal sparsity problem mentioned in Section 6.6.1, each $\mathbf{D}_t$ of each dataset has a significant number of missing values. NMF solves this problem by imputing those missing values while imposing non-negativity and sparsity constraints.

6.6.5 Results

Table 6.4 shows the summary statistics of the NMF-based localization runtime. I can see that, on average, NMF-based localization is sufficiently fast for most real-time applications (much less than 1 second) for all datasets.

Fig. 6.9 shows the prediction evaluation results of the six GP models listed in Table 6.3 for both Pittsburgh (PGH) and Washington (WAS) across the three evaluation metrics (MAE, MAPE, and RMSE) averaged over all the trials on both test weekday (WD) and weekend (WE). For Pittsburgh (top), it can be seen that global GPs without side information always have the highest error rates. Grid-based local GPs perform better than global GPs; however, the predictions with the lowest errors come from NMF-based local GPs. Having side information always improve prediction accuracies with weekdays having stronger effects than weekends. Side information has the strongest effects on global GPs, which is not surprising given its largely diffuse training set. All the three metrics display consistent observations with MAPE having the highest variance. My pairwise Wilcoxon tests between global GPs and NMF-based local GPs (with/without side information) and between NMF-based local GPs and grid-based local GPs (with/without side information) are all significant at the 5% level, except for LGP^+ and LGR^+ for weekday data evaluated using MAPE. It can be argued that NMF-based local GPs with

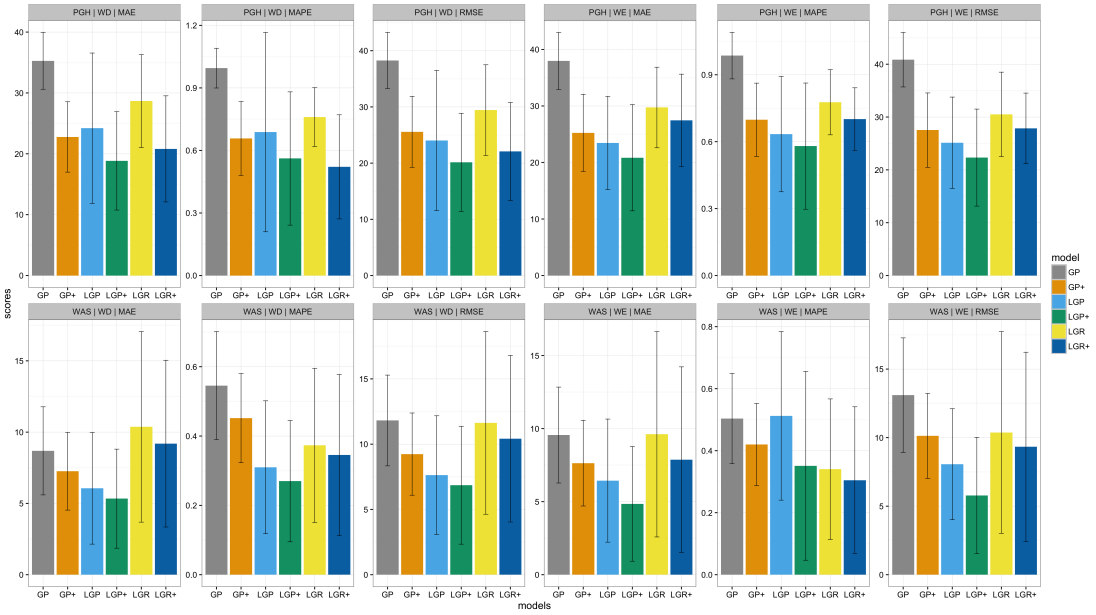

FIGURE 6.9: Evaluation of speed prediction across the 6 models using the metrics: MAE, MAPE, and RMSE. Datasets evaluated are: Pittsburgh (PGH) and Washington (WAS) on weekday (WD) and weekend (WE).

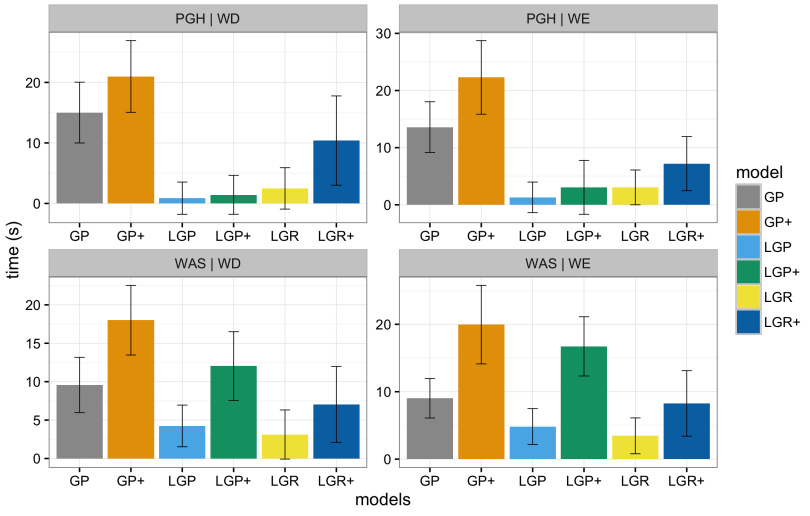

FIGURE 6.10: Evaluation of the runtime performances across 6 models for the cities of Pittsburgh (PGH) and Washington (WAS) on weekday (WD) and weekend (WE).

side information is the best-performing model overall. This demonstrates the effects of learning from a smaller, but *more relevant* local subsets of training data [92].

For Washington, similar observations can be seen in Fig 6.9 (bottom). Global GPs without side information almost always have the highest error rates. Grid-based local GPs yield high variances and, at the same time, perform much worse than those in

Pittsburgh (when compared to global GPs). This showcases the inability of simple spatial grid partitioning to adequately model more complex traffic patterns in a completely different urban setting. Having side information invariably reduces error rates for all the models. Similar pairwise Wilcoxon tests were performed, all of which are significant at the 5% level, except for the three pairs: GP vs. LGP, GP^+ vs. LGP^+, and LGP^+ vs. LGR^+ for weekend data evaluated using MAPE due to high variances. It can thus be concluded that NMF-based local GPs with side information is the best-performing model for weekday data. It is, however, inconclusive for weekend data.

Fig. 6.10 shows the evaluation of runtime performances for all the models. For Pittsburgh (top), NMF-based local GPs significantly outperform global GPs by more than 10 folds (i.e., NMF-based local GPs are more than 10 times faster) for weekday, with and without side information. Higher K^* significantly reduces the runtime of local GPs as evidenced by shorter runtime on the weekday compared to that on the weekend. Apart from that, I see a similar pattern for weekend: both local GPs significantly outperform global GPs in terms of runtime, and NMF-based local GPs are more than 6 times faster. Having side information invariably improves prediction accuracies, but also increases runtime for all models. This is particularly true for grid-based local GPs, which suggests that the chosen set of side information induces more complex correlation structure (hence, parameter estimates) for GP learning . All Wilcoxon pairwise tests are statistically significant at the 5% level.

For Washington, Fig. 6.10 (bottom) shows similar observations: local GPs are faster than global GPs and having side information increases runtimes. What is interesting, however, is the observations that NMF-based local GPs with side information have significantly higher runtimes than grid-based local GPs. This might be due to the need of $LGP^+$to model more complex local subsets that results from non-uniform partitioning of training data than LGR^+. I further discuss this observation in the following section. All Wilcoxon pairwise tests are significant at the 5% level.

6.6.6 Discussion

For all datasets, global GPs incur high runtimes and have low prediction accuracies, which render them impractical for real-time applications. Local GPs thus become viable solutions to real-time traffic prediction with significantly lower runtime costs, with and

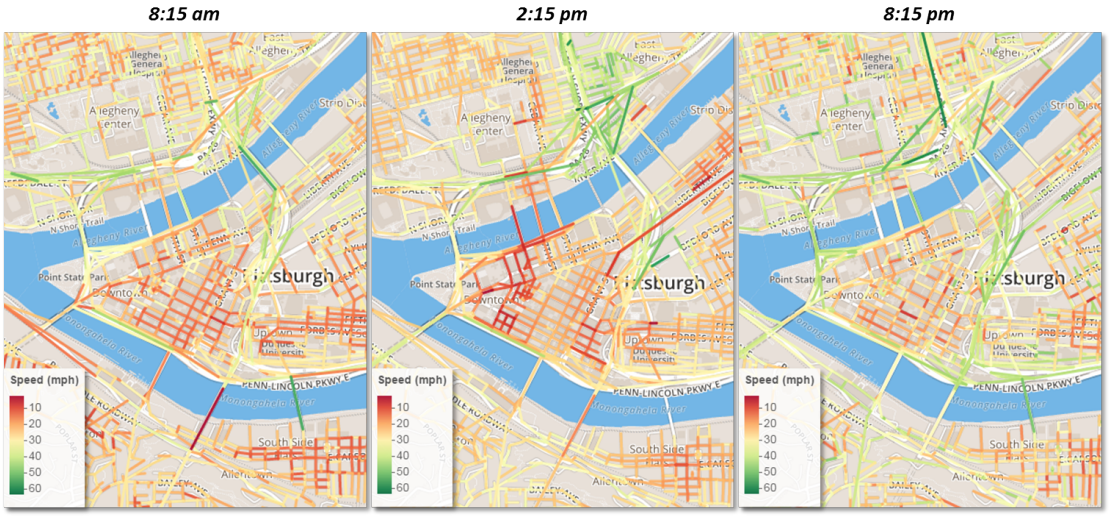

FIGURE 6.11: Visualization of the spatiotemporal inference of traffic speed in Pittsburgh on the test weekday (August 28, 2014) using LGP$^+$. Training data obtained using the sliding window method from 5% of the road network and at three time points: 8 a.m., 2 p.m., and 8 p.m. Test time for each is a 3-step ahead prediction.

without side information. Local GPs with side information can give more accurate predictions but at increased time costs, and thus are more suitable for longer-horizon applications. On the other hand, local GPs without side information are more suitable for shorter-horizon applications, where decisions are to be made fast.

In most cases, NMF-based local GPs predict significantly better than grid-based local GPs, as shown in Fig. 6.9. I have also seen that, for the same set of side information features, different localization methods can result in significantly different runtimes for training local GPs. This is due to my uniform (and uninformed) selection of the same set of side information listed in Table 6.2 for both cities. Different cities induce different traffic phenomena and optimization problems. It is unreasonable that the same set of side information is able to model those distinct phenomena equally effectively. Discriminatory feature selection should have been exercised. Feature selection is an entire different issue and often relies on domain knowledge and is out of scope of this work.

In practice, one needs to trade off between model expressiveness (i.e., side information) and efficiency depending on one's sensitivity to accuracy and time. How to select side information also matters. In this respect, it is important to consider the most relevant side information (and the smallest subset of such) to the phenomenon being modeled in order to maximize its benefits. Such knowledge also belongs to the domain expert.

Fig. 6.11 visualizes the spatiotemporal inferences of traffic speed on *the entire* road network of Pittsburgh (zoomed into the downtown area) on the test weekday at three test times: 8:15 a.m., 2:15 p.m., and 8:15 p.m. NMF-based local GPs with side information were used to make the inferences. The training sets were derived using the sliding window method at time $t \in \{8, 14, 20\}$ hours. Each test time is a 3-step ahead prediction. At each test time, the observed speeds cover 5% of the whole network (while prediction makes for the entire of it). Fig. 6.11 shows clearly the morning rush hour effect at 8:15 a.m., where the main roads leading to the downtown and other business areas become highly congested (with lower speed distribution). At 2:15 p.m., congestion becomes more localized to the business areas because of office hours, while the main roads have become visibly more cleared of traffic. At 8:15 p.m., traffic on the whole gets visibly faster with main roads leading to and from the business areas having apparently much faster flows, and congested areas have now become more localized to the nightlife areas.

6.7 Conclusion

This chapter addresses an important and typical problem in urban computing: real-time traffic speed modeling and prediction. To this end, I propose the novel idea of localizing spatiotemporal Gaussian processes (GPs) using non-negative matrix factorization (NMF). In addition, I make use of the expressiveness of GP kernel functions to model traffic speed through directed links of a road network and incorporate side information via additive kernel. Extensive empirical studies using real-world traffic data collected in diverse geospatial settings have demonstrated the efficacy of my proposed approach, in terms of both computational efficiency and prediction accuracy, against the baseline global and local GPs. I also show that a tradeoff exists between model expressiveness and runtime performance when side information is taken into account. It is therefore important to consider the most relevant side information for that matter.

Chapter 7

Incident Prediction for Law Enforcement Resource Optimization

7.1 Introduction

In today's world of heightened security concerns, there is an ever increasing pressure on law enforcement agencies around the world to efficiently deploy resources and timely respond to emergent incidents. Such pressure is further aggravated by the urbanization trend across the world [41], resulting in manpower crunches on law enforcement agencies trying to meet the rising demand in large and densely populated urban areas.

With the availability of spatiotemporal data that provides fine-grained details of the incidents (e.g., precise time and location of occurrence, as well as the police response to the incidents), it is now possible to make high-precision predictions of future occurrences of the incidents using advanced machine learning models. Applications of such capabilities include predictive policing and adaptive patrolling [80]. Furthermore, such models should be able to incorporate the rich set of socioeconomic and geopolitical features underlying those incidents to learn and generate the distributions of incidents under diverse what-if scenarios. This proves indispensable to an effective design of any data-driven staffing models for the allocation of law enforcement resources. I call such problem the "resource planning model" for law enforcement, which is introduced in Appendix A.

Briefly speaking, the problem calls for an optimal allocation of law enforcement resources over space and time that satisfies a certain quality of service (QoS) constraint.

This chapter tackles the incident prediction problem, which is to predict the number of incidents of a given type[1] that would occur at a query location and time. The main motivation is to test the robustness of the resource planning model (described in Appendix A) in face of uncertainties (i.e., randomness of the spatiotemporal process underlying the incidents) and changes in policy (e.g., merging of base locations' boundaries due to reduced resource requirements). I look at the incidents from a distributional point of view, where I discretize the continuous space into grid squares and the continuous time into intervals. I then count the number of incidents that occurred within each discrete grid square and time interval. I finally use the spatiotemporal Gaussian process (GP) introduced in the previous chapter to model and predict such count variables. Thus, the main technical challenge here is the incorporation of non-spatiotemporal features (i.e., "side information") into the GP kernel function for incident generation [30].

7.2 Problem Statement

Suppose the city's map can be divided into finite grid squares and, similarly, the continuous timeline can be hashed into finite intervals, my goal is to model the number of incidents that occurs within each grid and at each interval. That is, let $|S|$ denote the number of spatial grids and $|T|$ the number of temporal intervals. For each type of incident, I wish to have a *distribution* of the *count* within each combination of $|S| \times |T|$. I call each of such combinations a "bin" as the process of discretizing the spatial and temporal dimensions is essentially spatiotemporal data binning. In other words, given a query tuple (x, y, t), where x, y and t represent the longitude, latitude and timestamp, respectively, I first need to hash it into a bin i that has features $\mathbf{f}_i$. Let (x_i, y_i) denote the centroid coordinates of the bin and t_i the interval index. Given the tuple $(x_i, y_i, t_i, \mathbf{f}_i)$, I then wish to predict (or generate) the number of incidents that happens in i. For simplicity, within each bin, the incidents are assumed to have uniform distribution and the granularity of such spatiotemporal binning is a given parameter.

[1] Each corresponds to a certain urgency class.

TABLE 7.1: Summary of additional notations used in this chapter.

Notation	Description
$S, \|S\|$	Spatial dimension and number of spatial grids, respectively
$T, \|T\|$	Temporal dimension and number of time intervals, respectively
δ, τ	Parameters specifying the granularity of the spatial and temporal dimension
x, y, t	Longitude, latitude and timestamp, respectively
x_i, y_i, t_i	Centroid coordinates of bin i and interval index, respectively
$\mathbf{f}_i$	Feature vector of bin i

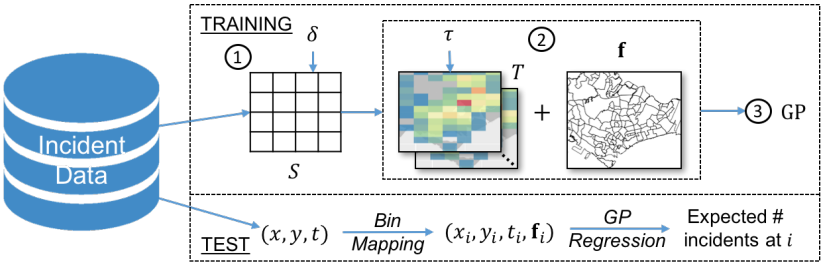

FIGURE 7.1: The Gaussian process (GP) framework for incident prediction.

Table 7.1 summarizes the additional notations used in this chapter. It is worth emphasizing that the problem discussed and the solution proposed in this chapter do not consider complex factors such as demographic, socio-economic and socio-political features that could have shaped the macro-trend of crime patterns over the longer-term periods (e.g., years). I'd rather look at crimes (and its incidents) as discrete events that could be predicted purely based on the correlations between spatiotemporal and geo-political features (e.g., division and sector boundaries) in the short-term periods. The distribution of such events is expected to fluctuate in the shorter window of prediction (e.g., days, weeks), but no long-term trend could be predicted or should be inferred from there. Such a complex problem is outside the scope of this thesis.

7.3 Spatiotemporal GP for Incident Prediction

Using the "count" as a response variable, I model the incident generation within each bin as a spatiotemporal process. To this end, I make use of the spatiotemporal Gaussian process (GP) introduced earlier in Section 6.5. Section 7.3.1 introduces the GP framework used for incident prediction, and Section 7.3.2 describes the special kernel function designed for the proposed GP model that incorporates side information.

7.3.1 Solution Framework

Fig. 7.1 illustrates the proposed GP framework for incident prediction. Like all other frameworks in this thesis, it consists of a training and a test phase. Let δ and τ ($0 < \delta, \tau < 1$) be the input parameters specifying the granularity of the spatial grid and time interval, respectively, the training phase consists of the following steps:

Step 1 The spatial dimension S is discretized into uniform grid squares using the parameter δ, which specifies the number of spatial divisions (i.e., rows/columns) per axis. Assuming each axis is of unit length, the number of divisions is $\frac{1}{\delta}$ each;

Step 2 Likewise, the temporal dimension T is discretized into intervals using the parameter τ. Each incident is then binned into one of the $|S| \times |T| = \frac{1}{\delta^2 \tau}$ bins, and I count the number of incidents in each bin. I treat such count as a response variable coupled with the feature vector $\mathbf{f}$[2] of the bin as "side information";

Step 3 Finally, I use a spatiotemporal GP to learn the distribution of the incidents over space and time taking into account their side information $\mathbf{f}$. To this end, I use the centroid coordinates (longitude and latitude) of the square grids to represent the spatial features of the random variables.

In the test phase, my goal is to predict the number of incidents that occur within a square grid and during a certain time interval. Given a query tuple (x, y, t), I first hash it into bin i to produce (x_i, y_i, t_i). I call this "bin mapping". Let $\mathbf{f}_i$ be the feature vector (i.e., side information) of bin i, given the tuple $(x_i, y_i, t_i, \mathbf{f}_i)$, I then simply perform a GP regression (refer to Section 6.5.1) on the learned model in the training phase to compute both the expected number of incidents and its variance.

The main difference between this GP framework for incident prediction and the GP framework for traffic speed prediction discussed in the previous chapter is that, in this solution, the incidents are predicted in an "offline" manner. That is, the model is trained and its parameters are stored (on disk, e.g.) first and then when an inference query is made, the corresponding model parameters are retrieved to make prediction. This is in contrast to the previous GP model where (local) training data are retrieved right after the inference query is made and which is used to trained a local GP model on the fly in order to make prediction in real-time – that is, in an "online" fashion.

[2]E.g., artificial boundaries such as base locations or peak/off-peak hours classification.

TABLE 7.2: Features used to learn the GP model for incident prediction.

Feature	Description
`location`	Longitude and latitude coordinates of the incident
`hours`	The integer hours of the incident's occurrence time (0–23)
`is_weekend`	Binary variable whether the incident occurs on the weekend
`neighborhood`	Categorical variable specifying the incident's neighborhood
`sector`	Categorical variable specifying the incident's sector

7.3.2 Kernel Function for Incident Distribution

Let i and j be two separate spatiotemporal "bins", according to Section 6.5.2, the kernel function between i and j can be written as:

$$k((x_i, y_i, t_i), (x_j, y_j, t_j)) = k_s((x_i, y_i), (x_j, y_j))k_t(t_i, t_j), \tag{7.1}$$

where k_s and k_t are the spatial and temporal kernel function, respectively.

I can further incorporation side information $\mathbf{f}$ using the additive kernel feature [27]:

$$\begin{aligned} &k((x_i, y_i, t_i, \mathbf{f}_i), (x_j, x_j, t_j, \mathbf{f}_j)) \\ &= k((x_i, y_i, t_i), (x_j, y_j, t_j)) + \sum_f k(\mathbf{f}_i, \mathbf{f}_j), \end{aligned} \tag{7.2}$$

$\forall$ feature $f \in \mathbf{f}$. A typical kernel function for the side information is the linear kernel.

7.4 Experiments

7.4.1 Dataset

Refer to Sect. 3.2.3 for the dataset description.

For the experiments described in this chapter, I only use the incidents occurred in the last 12 weeks of the period. Table 7.2 summarizes the features of the incidents used for learning the model. These features are selected based on regression analysis[3], which is not discussed here to save space. Non-binary categorical features are modeled using one-hot encoding (i.e., transforming them into binary dummy variables). It is not hard to see that `is_weekend`, `neighborhood` and `sector` are the side information.

[3]Choosing those significant at the 5% level.

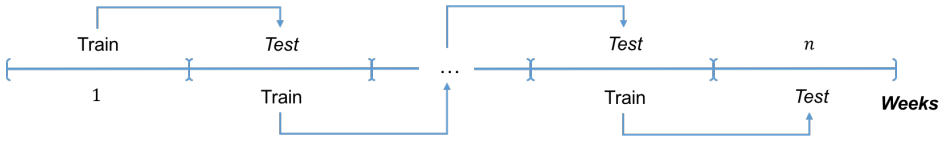

FIGURE 7.2: Experimental design to validate the GP framework for incident prediction. n is the number of weeks ($n = 12$ in my experiments).

7.4.2 Experimental Design

For simplicity, I classify the incidents into two types: urgent and non-urgent, based on its given priority class. By definition, urgent incidents have higher urgency class than non-urgent ones, i.e., they have shorter maximum response time threshold. I then model two subsets of data separately: (1) urgent incidents only, and (2) both urgent and non-urgent incidents (i.e., all the incidents). About $\frac{1}{3}$ of all the incidents are urgent. For each subset, I design the following experiment to evaluate the model.

Because of the time series nature of the incidents, I adopt the "sliding window" approach to sequential cross-validation by alternating between training and testing on a weekly basis for each of the 12 weeks. That is, let w ($1 \leq w < 12$) be a week during the period, then all the incidents occur in w (and in w only) are used for training, and all those in $w+1$ are used for testing. In the next cycle, $w+1$ becomes the training week (without considering the data in w or before), and $w+2$ is the test week. This process repeats until the last week of the 12 is the test week. Each training week only learns from the data of the *true* distribution (and not the generated incidents). Each test week compares the *generated* incidents with the true distribution using the proposed evaluation metrics. Such metrics are then collected for each of the test weeks and aggregated for final model comparison and significance tests. Fig. 7.2 illustrates my design of experiments to evaluate the GP framework for incident prediction.

Finally, I set $\delta = 0.01$ and $\tau = \frac{1}{24}$ (hours) in my experiments. That is, the spatial dimension is divided into a grid of 100×100 squares, and the temporal dimension is divided into intervals of 1 hour each (indistinguishable for each day). The total number of bins is thus 240,000 (with a large number of them empty, i.e., having no incidents).

7.4.3 Evaluation

The following models are used as the baselines for comparison with the proposed GP:

- Linear regression (LM);
- Random forest (RF) regression;
- Support vector machine (SVM) regression with RBF kernel – refer to Eqn. (6.9);
- Gradient boosting regression (GBR).

All these models use the features listed in Table 7.2. For the baselines, I use the off-the-shelf implementation of the `Scikit-learn` machine learning library [79] and perform the same experiments as described above to evaluate them. For all the models, I use the RMSE, MAE, and MAPE metrics to evaluate them – refer to Eqn. (6.17) – (6.19). In essence, I am measuring the differences between the expected number of incidents (predicted by the GP or baselines) and the real number of incidents per bin. I then average the metrics over all the test weeks and report their means and variances.

7.4.4 Results

Fig. 7.3 shows the results of my experiments for both subsets of data: "Both" (top – all the incidents) and "Urgent" (bottom – urgent incidents only). For all the incidents, the figure shows that GP is the best performing model (lowest mean errors) for all the metrics (particularly MAPE). Pairwise t-tests between GP and the baselines are significant at the 5% level. For urgent incidents, GP still performs competitively (i.e., as good as or better than) compared with the baselines. Except for GBR, GP performs significantly better than the others. It, however, performs as good as GBR as the figure shows (i.e., not significant at the 5% level for the three metrics). This is likely due to the different tactics and respond strategies deployed for urgent incidents. Richer set of features are perhaps more useful in modeling them. This also probably explains why SVM performs particularly badly for urgent incidents (with MAPE approaching 0.40).

Notice that in both cases (urgent and both), I don't observe a "big jump" in performances of GP (if any), but rather a modest (but significant) improvement. This is to be expected as the phenomenon under model is rather complex and the set of features used is quite limited. On the other hand, employing more features would make the model more complex but risk overfitting. My main purpose here is to demonstrate that, under the same set of features, the proposed GP model renders a competitive performance.

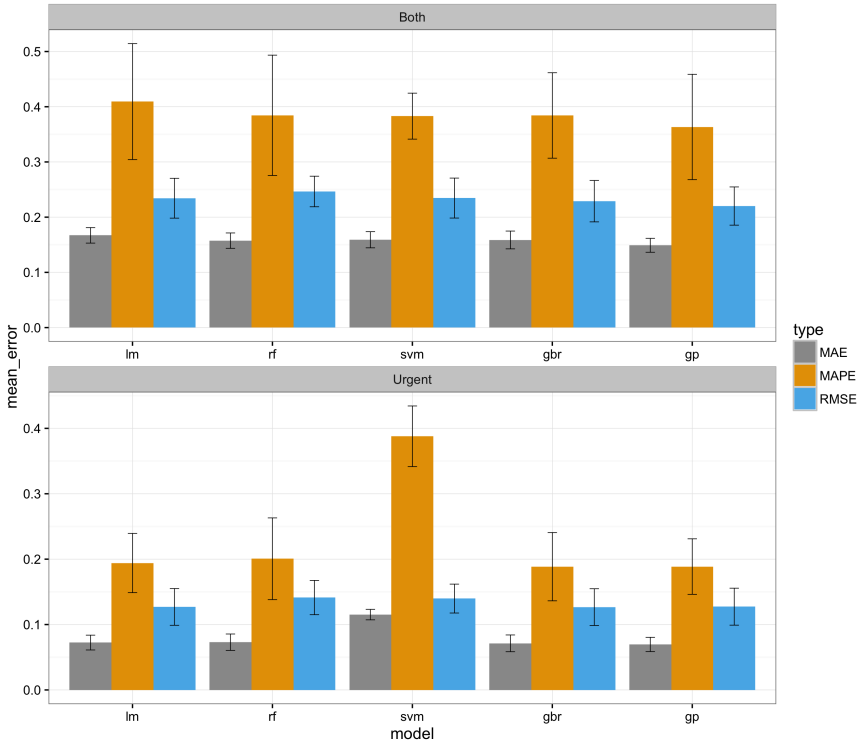

FIGURE 7.3: Mean cross-validation results for incident prediction models for all incidents (top) and urgent incidents only (bottom).

7.4.5 Discussion

For general incident prediction purposes, I conclude that the proposed GP is the best-performing model. Not only because of its prediction accuracy, but also because of its ability to generate incident distribution in face of changing policies (encoded as "side information"). For example, two neighboring sectors might be merged into one and adopt the policing policy of either one. Such ability to generate distributions proves indispensable to test the robustness of the resource planning model and compute $\hat{\alpha}$ as shown in Fig. A.1. However, it is not discussed here because it falls outside the scope of the thesis, which focuses on modeling spatiotemporal phenomena in urban settings.

Finally, Fig. 7.4 visualizes the spatial distribution of the all incidents for the whole last week (week 12) of the evaluation period. The top panel shows the predicted distribution (learned from week 11) and the bottom one shows the true distribution. The figure clearly shows that my GP model can quite correctly predict the distributional pattern over space of the incidents, except for a few minor discrepancies with the ground truth.

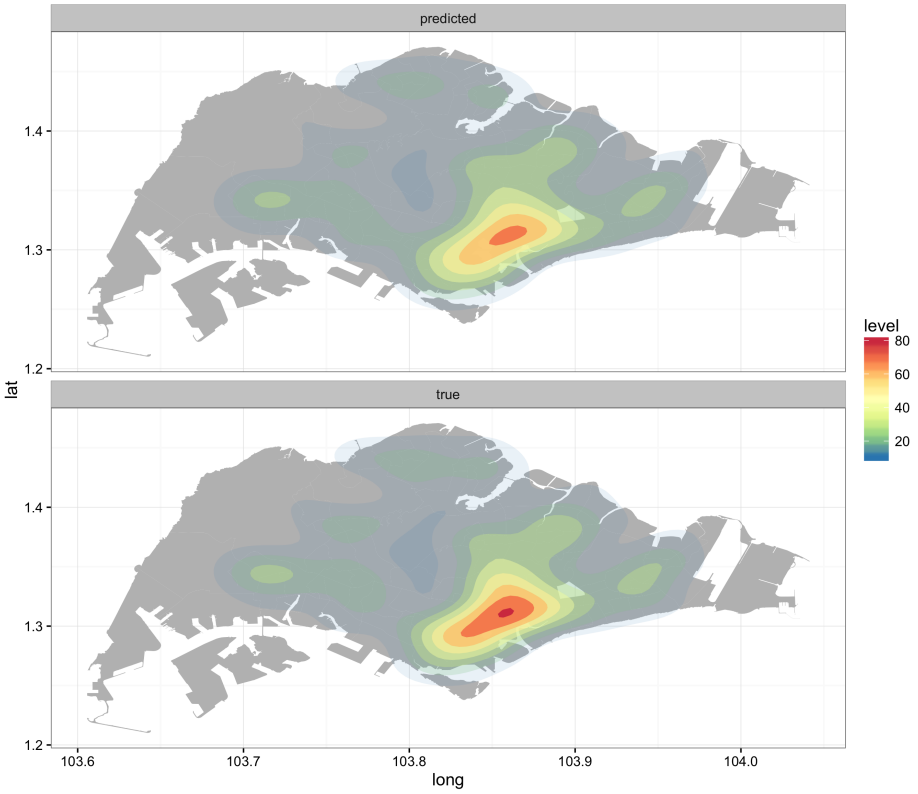

FIGURE 7.4: Heat maps visualization incident distribution (for all the incidents) over one test week (week 12) between the predicted (top) and true (bottom) distribution.

7.5 Conclusion

In this chapter, I focus on one an important problem for law enforcement resource optimization: incident prediction. To this end, I propose a GP framework that leverages on what has been developed in the previous chapter to model the spatiotemporal distribution of the incidents taking into account side information such as temporal classification and spatial boundaries. I perform extensive experiments using real-world incident data to validate the framework. My experimental results indicate the superior performance of the framework and its utility in testing the robustness of the resource planning model. Overall, I believe that this work renders an essential stepping stone to an overall data-driven solution framework for law enforcement agencies (refer to Appendix A) to efficiently and effectively respond to incidents and plan for future needs.

Chapter 8

Conclusion

8.1 Summary

This thesis has taken the reader through a journey of solving some of the most salient problems in urban computing: from predicting how people move about in cities under certain constraints in space and time, to predicting traffic phenomenon and the occurrences of crime incidents in urban areas. As impressive as that sounds, it has only scratched the surface of urban computing, which is still a young and fast growing field riding on the explosive growth of big data, machine learning and rapid urbanization across the world. Through solving the specific problems addressed in this thesis, a common pattern of problem solving emerges that could be abstracted and integrated into a general solution framework. Such a framework is, therefore, highly extensible to other problems of similar nature, and has serendipitously become the purpose of this thesis.

The integrated framework combines machine learning methods in creative ways to solve urban problems using spatiotemporal data. Spatiotemporal data possess certain modeling challenges, most important of which are the violation of the i.i.d. assumption due to the inherent nature of spacetime processes and the scalability issue of big data in general. The proposed framework overcomes these challenges by a three-step process: (1) clustering, (2) environment modeling, and (3) machine learning modeling.

In clustering, the training set is split into finite subsets (or clusters), each modeling a homogeneous behavior of the phenomenon. Doing so also models the heterogeneity of the data and reduces the complexity of model training. In Chapters 4–5, trajectory

clustering models the heterogeneous population of theme park visitors. In Chapter 6, spatiotemporal clustering helps reduce the training set into local subsets, which speeds up training significantly. In environment modeling, the input data and the features representing the built environment are somehow combined together to more accurately model the underlying phenomenon. In Chapters 4–5, those are the frames of references under which agents make their decisions. In Chapters 6–7, those are the spatial features representing the road networks and urban areas. It is in the latter case[1] that the i.i.d. assumption becomes a problem and when kernel functions come to the rescue. Finally, a machine learning model is trained based on the input data and the modeled environment. In Chapters 4–5, it is revealed preference and reinforcement learning model, respectively. Whereas, Chapters 6–7 use the generative Gaussian process (GP) model.

The framework is then "instantiated" to solve three specific instances of spatiotemporal problem in urban environments: human mobility prediction (Chapters 4–5), traffic speed prediction (Chapter 6) and incident prediction (Chapter 7). By solving the problems studied in this thesis, two subtextual themes emerge:

1. **Rationality.** Under what conditions do people make optimal decisions?
2. **Generalizability.** How much data is needed to train a *good enough* model?

The first theme emerges from solving the decision problems in Chapters 4–5. In both chapters, the prediction problems are solved by modeling the decision-making process from the agent's point of view. In Chapter 4, I deal with an optimal bundle problem, in which a knapsack problem lends itself as a natural solution. Indeed, it has been shown that the knapsack decision model outperforms the baselines in various settings. Recall that the time budget given is rather generous (i.e., 10 hours), given the number of items (attractions) to choose is only 4 out of 16. It is thus reasonable that most people make optimal decisions in a "knapsack"-style fashion. On the other hand, it is non-trivial how to apply the classic knapsack problem in this new spatial setting, where cost evaluations are dynamic. To this end, HMM is used to model the agent's "frame of reference" when making decisions upon knowing its "initial intention". Such given initial intention explores the relationship between the amount of information required for an agent to make an optimal decision or, in other words, for the proposed model

[1]Generally, generative models such as GPs and most Bayesian models (e.g., naive Bayes) make the distributional assumption of i.i.d. [71, 83].

to make an accurate prediction. I have shown through extensive experiments that the model is rather robust to the gradual reduction in that given information, where its prediction accuracies do slightly decrease but still outperform the baselines.

In Chapter 5, the visitor's actual trajectory is to be predicted, which naturally calls for sequential decision modeling. MDP-based decision models are thus proposed, together with greedy heuristic ones. Through extensive experiments, I have shown the existence of two types of visitors, one that the MDP model predicts better, and one in which the greedy heuristic is the better choice. It turns out that the first type also has substantially more time budget than the second, almost 2 hours on average. This explains why the first type makes more optimal decision than the second, whose greedy heuristic is mathematically suboptimal. In other words, more time is needed to make better decisions. As simple as that sounds, this was not obvious in the first place. Hence, in both chapters, time (budget) is the common thread and whose role is crucial to the optimality of decision making. In Chapter 4, it does not seem to adversely affect the decisions, since the given budget is quite generous compared to the task. However, in Chapter 5, when the choice set is such bigger, time becomes the discriminating factor that decides when an agent makes a more (or less) optimal decision.

The second theme emerges from the prediction problems in Chapters 6–7. In both chapters, GPs are used to solve the predictive (of traffic speed) and the generative (of crime incidents) problem. In Chapter 6, where traffic speed distribution over road networks are to be predicted, I have shown how the complexity of the problem can be significantly reduced by clustering the training set into "localized" subsets, each is used to train a "local" GP. This only works due to the relevancy of the clustered data points and their "side information", which effectively (and efficiently) predicts the traffic query at hand. This essentially says that the "more" doesn't necessarily mean the "better". What matters is the clever selection of relevant training data to what is being predicted. On the other hand, the more may actually mean the better when it comes to the expressiveness of the kernel function. The chapter has shown that a more expressive kernel function that incorporates more features often makes better prediction, but it also takes more time to train. Therefore, the question of how much data is good enough really boils down to the sensitivity of the modeler to the prediction accuracy and runtime performance, which can be a natural tradeoff to each other.

In Chapter 7, the task is to predict (or rather generate) the distribution of crime incidents over urban areas. The second theme is explored in a slightly different way. While in the previous chapter, it is the ability to make inferences based on a small subset of training data, in this chapter, it is the ability to generalize what the model has seen in the training data to what does not yet exist, i.e., to generate incidents based on a set of alternative scenarios specified by the modeler. Hence, the concept of generalizability is explored from two difference perspectives using the same model: one emphasizes on the using as little data as possible by introducing local GPs, and the other on generating mockup scenarios as realistically as possible using a global GP. In either case, feature selection plays an important role that could not be understated, even though it was not elaborately described in the chapters due to limited scope.

Both themes explored in this thesis are in fact the direct result of the two types of spatiotemporal data used. As explained earlier in Chapter 1, they are the trajectory-based type and the sensor-based type. While for the trajectory-based type, the phenomenon is better predicted by modeling it from the agent's point of view using a discriminative model, for the sensor-based type, it is better predicted by modeling it more holistically using a generative probabilistic model that takes into consideration the correlation structure of other "nearby" sensors (i.e., data points) in space and time.

8.2 Future Directions

8.2.1 Follow-up Work

Chapter 5 is considered to have finished the work left in Chapter 4 by predicting an actual trajectory produced by an agent (rather than an unordered set). The work left by Chapter 5 are the followings. For the "early bird" agents, it would be interesting to know if a more sophisticated sequential decision model such as the adaptive knapsack with stochastic rewards [45] may be better. For the myopic "latecomers", it raises the question of how to more effectively model them, since the proposed greedy heuristic is rather naive. An exemplar work along this line is due to Sobel and Wei [94].

For the prediction problems in Chapters 6–7, it might help to model the phenomena considered (i.e., traffic speed and crime incidents) even more holistically by fusing heterogeneous sources of data (e.g., social media and real-time crowd-sourced data such as news reports) with the original spatiotemporal data. This is one important problem in urban computing as pointed out by Zheng et al. [112]. Exemplar works along this line such as [18, 20, 111] have shown to significantly improve predictions.

Specifically, the work in Chapter 6 potentially paves the way to useful applications in autonomous vehicle routing thanks to its abilities to make high-precision and adaptive predictions of traffic flows with efficiency. In a not-so-distant future, urban mobility and transportation systems will be revolutionized by the advancements of autonomous and connected vehicles, which allow for driverless vehicular control and enable effective information dissemination among vehicles. Adaptive vehicle routing will then become one of the most viable technologies to achieve safer and more reliable autonomy. Indeed, by processing and learning real-time traffic information from probe vehicles and social media (e.g., Twitter, Waze), vehicles can be provenly routed safely and efficiently in a non-myopic way. An early exemplar work in this domain is due to Liu et al. [65].

8.2.2 Towards Urban Reasoning

Urban reasoning is "the ability to help urban planners fine-tune their plans using an AI model and urban sensed data" according to Assem et al. [4]. Or simply put, having technologies to predict and detect urban phenomena across a city is interesting, but what is deeper, and would make more impact, is the ability to understand the reasons for the detected patterns. Hence, urban reasoning extends the vision of urban computing to provide insights about the reasons underlying the major challenges that cities face [4].

In the context of the proposed framework, it is the ability to derive valid scientific conclusions from spatiotemporal data such as the valid measures of association between variables observed in space and time and causal and ecological inferences (i.e., drawing conclusions about individuals from aggregate-level data). Causal inference with observational spatiotemporal data is particularly challenging for the same reason that spatiotemporal statistics is hard. That is, the i.i.d. assumption does not hold and the data cannot be analyzed as a random sample. An illustrating work in this domain is due to Flaxman et al. [30], in which the authors were not only able to predict crime

incidents in the Chicago metropolitan area "far into the future", but also explain for the discovered trends and patterns. This is achieved using expressive spectral mixture covariance kernels capable of learning intricate structure in large datasets.

While kernel methods have been always a favorite one-stop solution to most geospatial problems, generalizing and incorporating it into a general solution framework that is capable of deriving valid insights and reasons is a non-trivial task. To this end, many approaches to causal inference have relied on statistical tests of independence between variables such as: Fisher z-score, Pearson correlation, and more recently the Hilbert-Schmidt Independence Criterion (HSIC) [39]. In fact, the entire framework of graphical models for causal inference proposed by Pearl [78] relies critically on the assumption about d-separation in graphs. Therefore, testing these assumptions with observational spatiotemporal data requires applying a valid conditional independence test.

The above tests are prone to report spuriously high correlations when used on non-i.i.d. data due to the underlying autocorrelated structure [31]. I propose to extend the capabilities of the framework to incorporate causal reasoning by having a component to test the conditional independence relationships before the machine learning modeling step. Specifically, given a tuple of variables (X, Y, Z), we wish to test whether X and Y are independent of each other given Z, denoted as $X \perp\!\!\!\perp Y|Z$. Such a test can be robustly performed by reducing the question about conditional independence with non-i.i.d. data to the question about unconditional independence with i.i.d. data, which can be readily answered using the HSIC. Following the framework proposed by Flaxman et al. [31] and let f be a spacetime process, the following steps can be performed to test $X \perp\!\!\!\perp Y|Z$:

1. Pre-whiten each variable to eliminate its dependence on f and obtain the residuals: r_X, r_Y and r_Z;
2. Obtain the residuals ϵ_{XZ} and ϵ_{YZ} by performing GP regressions of r_X and r_Y on r_Z, respectively;
3. Use the HSIC to test for the independence $\epsilon_{XZ} \perp\!\!\!\perp \epsilon_{YZ}$.

After having established valid causal relations using the above tests of conditional independence, a wide range of machine learning models can be used to model such relationships, including GPs [30, 31] and graphical models (e.g., Bayesian nets) [78].

Appendix A

A Data-driven Solution Framework for Law Enforcement Resource Optimization

A.1 Introduction

Traditionally, the staffing and allocation model of law enforcement is accomplished by simple statistical means based on aggregate historical demands (i.e., the number of incidents) [103]. Given the uncertain and transient nature of such demand across space and time, such crude approach yields solution that often contains "slacks" at different space/time while shortages in others [69]. With today's availability of big spatiotemporal data that provides fine-grained details of the incidents, it is now possible to design adaptive and data-driven staffing models for the allocation of law enforcement resources that are sensitive to both demand patterns and operational timing constraints.

From the service industry's perspective, there are two major resource planning problems: (1) Deciding the staffing levels, that is the number of agents required on duty at each base location and time slot that satisfies a desired quality of service (QoS) constraint; and (2) Shift scheduling, that is to translate the staffing levels into practical work shifts. Staffing level optimization is important in reducing manpower cost and improving the service quality. Refer to [24], [53] and [17] for successful applications in service delivery

systems and call centers. In this thesis, we are concerned with the first problem of staffing level optimization in the context of law enforcement manpower planning.

More precisely, I was involved in a team project that studies the problem of optimizing the staffing level of law enforcing agents (i.e., police officers) across base locations and time periods throughout a day using a data-driven approach. Our goal is to design high-fidelity allocation strategies such as to achieve the QoS for each class of incidents that maximizes resource savings over the current practice. In order to understand the real-world context of the work presented in Chapter 7 as well as its significance, I discuss the key ideas and the overall solution framework in this appendix.

A.2 Problem Statement

Formally, the law enforcement staffing and allocation problem is concerned with deciding the staffing levels across different base locations $\mathcal{L}$ in order to meet the response time requirement Δ of a given set of incidents $\mathcal{R}$ within a given risk level α. In this chapter, staffing levels refer to the number of agents (or cars) needed at different base locations at different time periods of the day. There are two variants of the problem:

- Deterministic resource optimization;
- Stochastic resource optimization.

The former is described by the tuple: $< \mathcal{R}, \mathcal{L}, \mathcal{T}, \Delta, \alpha >$, where $\mathcal{R}$ is a set of incidents, and each incident $r \in \mathcal{R}$ is a tuple $< l, d, c, t, s >$, where l is the location, d is the demand for the resources (e.g., the number of cars), c is the class representing the urgency level, t is the time of occurrence, and s is the service time. Let the set of base locations for law enforcement agents be $\mathcal{L}$ and $T_{l,l',t}$ be the travel time (in minutes) from location l to location l' at time t. In practice, upon an emergency call, the operator would first identify and assign a certain urgency class to the incident. *Each urgency class has a certain maximum response time threshold, and satisfying which is the key metric for the QoS.* Let Δ be the maximum response time vector of all the urgency classes. Intuitively, more urgent incidents should be responded faster than less. That is, $\Delta_c < \Delta_{c'}$, where c is a class with higher urgency than c'. The QoS constraint allows at most α fraction of

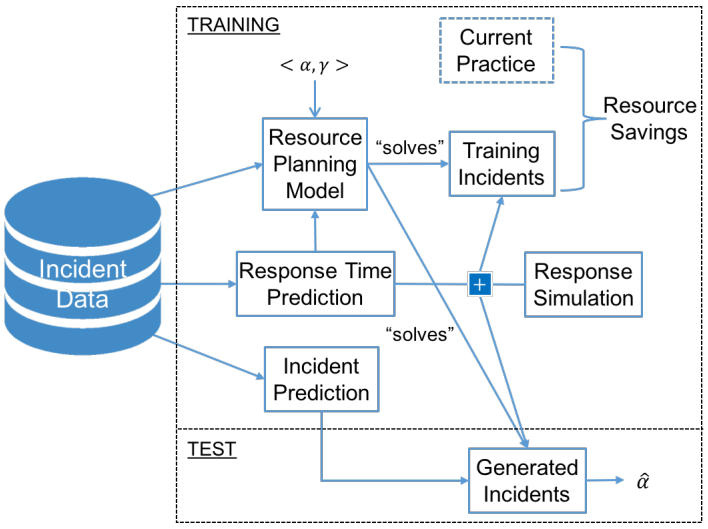

FIGURE A.1: The data-driven framework for law enforcement resource optimization with quality of service (QoS) guarantees.

the incidents within a planning horizon to "fail", i.e., whose response times exceed the maximum threshold Δ for the incidents in $\mathcal{R}$.

In the stochastic variant, we allow the QoS constraint to be violated over multiple periods in the planning horizon thereby giving rise to a chance constraint. Let α be a risk value for the QoS constraint such that it represents the planner's **risk attitude**. Let γ be a risk value for the chance constraints that allow the QoS constraint to be violated at most γ fraction of the incidents. For example, a chance constraint with a risk value γ for a 24-hour planning stipulates that the fraction of failed incidents is no more than α for $(1-\gamma)$ fraction of the days in a month. Given $0 \leq \alpha, \gamma < 1$, our goal is to decide the optimal staffing level and allocate the agents across all base locations such that the total resource requirements are minimal while satisfying the QoS constraint. That is, the probability that all the incidents are responded within the maximum threshold is $\geq 1-\alpha$ and the probability that all the chance constraints are violated is $\leq \gamma$.

A.3 Solution Overview

The data-driven solution framework for law enforcement resource optimization with QoS constraint is illustrated in Fig. A.1. As shown in the figure, the framework consists of two phases: training and test – both make use of the provided incident data.

In the **training phase**, given the input QoS parameters α and γ, we propose a mixed integer linear programming (MILP) model to do optimal resource planning with guaranteed QoS. In short, the optimal allocation produced by the model stipulates the precise number of resources (e.g., police cars) at each base location during the defined time interval[1]. Such allocation is then used to "solve" the incidents in the training data, i.e., to simulate the responses to the incidents given the allocation. Simultaneously, we also train two machine learning models for response time and incident prediction:

- The response time prediction model (presented in the next section) is to be used as an input function for both the resource planning model and the test phase later on. In particular, in the training phase, it is used partially[2] to simulate responses to the (training) incidents given the computed resource allocation.
- The incident prediction model (discussed in Chapter 7) learns the incident distribution from the training data and generates incidents under diverse scenarios to test the model's robustness in the test phase.

Finally, we compute the resource savings (supposedly ≥ 0) induced by the optimal allocation compared with the current (suboptimal) practice.

In the **test phase**, the optimal resource allocation learned in the training phase is then deployed to solve the generated incidents (together with the response time prediction function). A new fraction $\hat{\alpha}$ of the incidents that satisfy the QoS constraint is then computed and compared with the original α to evaluate the model's robustness.

A.4 Response Time Prediction

This section describes the machine learning model for response time prediction between the base location and the incident location in order to estimate the agents' response time. This is an important component of the data-driven framework for law enforcement resource optimization depicted in Fig. A.1. We first assume that the agent always starts from some centroid location of the sector. This is due to the complete lack of information about the agent's origin location from the provided data. We can then learn a regression

[1] Granularity of the time interval is also an input to the model in this sense.
[2] Together with the Response Simulation model, which is not discussed here.

TABLE A.1: Features used to learn the predictive model for the response time.

Feature	Description
`traffic_travel_time`	The time-dependent travel time computed by Google Maps API
`mean_travel_time`	Hourly average travel time from sector to sector captured by data
`is_urgent`	Binary classification whether the incident is urgent or non-urgent
`is_cross_dispatch`	Whether the responding car comes from a different local center
`num_cars`	The number of cars dispatched to respond to the incident
`hours`	The integer hours of the incident's occurrence time (0–23)

model that predicts the response time from the sector centroid to the incident location using the derived features. Table A.1 summarizes these features.

In this regression model, the true response time captured by the data (i.e., the duration from when the agent's car is dispatched to its arrival time at the scene) is the response variable. The features in Table A.1 were derived from a combination of both regression analysis and random forest feature importance. `traffic_travel_time` is the estimated travel time from the agent's sector centroid location to the incident location computed by Google Maps API at the time the agent was dispatched. `mean_travel_time` is the hourly average travel time from the agent's sector to the incident's sector computed from the historical data. `is_urgent` is a binary classification whether the incident is urgent or not. `is_cross_dispatch` is a binary variable indicating whether the dispatching car comes from a different local center. This typically happens when the resources at the incident's local center are being deployed and unavailable, thus resources from another (neighboring) center are called for. `num_cars` is the number of cars dispatched. `hours` is the integer hours (0–23) of the incident's timestamp.

The following models are evaluated: random forest (RF), linear regression (LM), support vector machine (SVM), and gradient boosting regression (GBR). All these models use the features listed in Table A.1. We additionally evaluate a naive "baseline" model that uses the `mean_travel_time` feature as the predicted response time for a given test incident. For SVM, the RBF kernel is used. For GBR, we use the efficient implementation in the XGBoost package [21]. We use both MAE and RMSE to evaluate the models. We perform 10-fold cross-validation (CV) and take the mean errors across the folds. The results are shown in Fig. A.2 with the mean error rates and the variances over 10 folds.

Fig. A.2 shows that the model that performs the best overall (by both MAE and RMSE) is the GBR model with average MAE well below 4 minutes. Unsurprisingly, the baseline model performs the worst (since it uses only one feature). GBR is a powerful ensemble learning method that produces a predictive model in the form of an ensemble

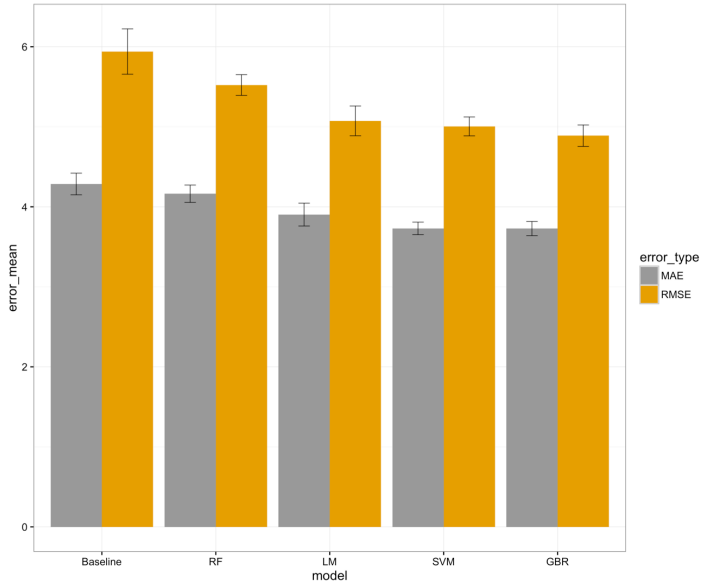

FIGURE A.2: Evaluation of models for response time prediction using 10-fold CV.

of week regression trees. It has been shown to be robust against overfitting (hence, suitable for highly skewed and long-tailed data such as response time) in many machine learning contests including the Netflix prize [10]. Also, noteworthy is that SVM is just slightly worse than GBR. However, it is not as scalable as GBR (which uses the efficient parallel implementation of XGBoost) to train big data. Therefore, we choose GBR as our predictive model for response time estimation in our MILP model.

Bibliography

[1] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In B. Schlkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, Cambridge, MA, USA, 2007. MIT Press.

[2] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, page 1. ACM, 2004.

[3] Sidney N. Afriat. The construction of utility functions from expenditure data. *International Economic Review*, 8(1):67–77, 1967.

[4] Haytham Assem, Lei Xu, Teodora Sandra Buda, and Declan O'Sullivan. Spatio-temporal clustering approach for detecting functional regions in cities. In *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*, pages 370–377. IEEE, 2016.

[5] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[6] Jean Bacon, Andrei Iu Bejan, Alastair R. Beresford, David Evans, Richard J. Gibbens, and Ken Moody. *Using real-time road traffic data to evaluate congestion.* Springer, 2011.

[7] Maria-Florina Balcan, Amit Daniely, Ruta Mehta, Ruth Urner, and Vijay V Vazirani. Learning economic parameters from revealed preferences. In *Web and Internet Economics*, pages 338–353. Springer, 2014.

[8] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[9] Eyal Beigman and Rakesh Vohra. Learning from revealed preference. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 36–42. ACM, 2006.

[10] Robert M. Bell and Yehuda Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[11] R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(4):679–684, April 1957.

[12] Paul E. Black. Levenshtein distance. *Algorithms and Theory of Computation Handbook*, 1999.

[13] Stephen P. Borgatti. Centrality and network flow. *Social Networks*, 27(1):55–71, 2005.

[14] Marco Bottero, Bruno Dalla Chiara, and Francesco Paolo Deflorio. Wireless sensor networks for traffic monitoring in a logistic centre. *Transportation Research Part C: Emerging Technologies*, 26:99–124, 2013.

[15] Yanshuai Cao, Marcus A. Brubaker, David Fleet, and Aaron Hertzmann. Efficient optimization for sparse Gaussian process regression. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 26, pages 1097–1105, 2013.

[16] Pablo Samuel Castro, Daqing Zhang, and Shijian Li. Urban traffic modelling and prediction using large scale taxi GPS traces. In *Pervasive Computing*, pages 57–72. Springer, 2012.

[17] Mehmet Tolga Cezik and Pierre L'Ecuyer. Staffing multiskill call centers via linear programming and simulation. *Management Science*, 54(2):310–323, 2008.

[18] Chao Chen. *Understanding social and community dynamics from taxi GPS data.* PhD thesis, Evry, Institut National des Télécommunications, 2014.

[19] Hao Chen, Hesham A. Rakha, and Shereef Sadek. Real-time freeway traffic state prediction: A particle filter approach. In *Proceedings of the 14th International*

Conference on Intelligent Transportation Systems (ITSC), pages 626–631. IEEE, 2011.

[20] Jie Chen, Kian Hsiang Low, Yujian Yao, and Patrick Jaillet. Gaussian process decentralized data fusion and active sensing for spatiotemporal traffic modeling and prediction in mobility-on-demand systems. *IEEE Transactions on Automation Science and Engineering*, 12(3):901–921, 2015.

[21] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[22] Michael Chui, Markus Löffler, and Roger Roberts. The Internet of Things. *McKinsey Quarterly*, 2(2010):1–9, 2010.

[23] A. Cichocki and A. H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E92-A:708–721, 2009.

[24] Yixin Diao and Aliza Heching. Staffing optimization in complex service delivery systems. In *Proceedings of the 7th International Conference on Network and Service Management*, pages 1–9. IEEE, 2011.

[25] Chris Ding, Xiaofeng He, and Horst D. Simon. On the equivalence of non-negative matrix factorization and spectral clustering. In *Proceedings of the SIAM International Conference on Data Mining*, 2005.

[26] N. R. Draper and H. Smith. *Applied Regression Analysis*. Wiley-Interscience, 1998.

[27] David K. Duvenaud, Hannes Nickisch, and Carl E. Rasmussen. Additive Gaussian processes. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 226–234, 2011.

[28] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[29] Nuno Ferreira, Jorge Poco, Huy T. Vo, Juliana Freire, and Cláudio T. Silva. Visual exploration of big spatiotemporal urban data: A study of New York City taxi trips.

IEEE Transactions on Visualization and Computer Graphics, 19(12):2149–2158, 2013.

[30] Seth Flaxman, Andrew Gordon Wilson, Daniel B. Neill, Hannes Nickisch, and Alexander J. Smola. Fast Kronecker inference in Gaussian processes with non-Gaussian likelihoods. In *Proceedings of the ICML*, pages 607–616, 2015.

[31] Seth R. Flaxman, Daniel B. Neill, and Alexander J. Smola. Gaussian processes for independence tests with non-i.i.d. data in causal inference. *ACM TIST*, 7(2):22, 2016.

[32] G. David Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[33] Chris Fraley and Adrian E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002.

[34] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[35] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Next place prediction using mobility Markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, page 3. ACM, 2012.

[36] Huiji Gao, Jiliang Tang, and Huan Liu. Mobile location prediction in spatio-temporal context. In *Nokia Mobile Data Challenge Workshop*. Citeseer, 2012.

[37] Gerd Gigerenzer, Reinhard Selten, et al. Rethinking rationality. *Bounded rationality: The adaptive toolbox*, pages 1–12, 2001.

[38] Adam Greenfield and Mark Shepard. *Urban computing and its discontents*. Architectural League of New York, 2007.

[39] Arthur Gretton, Kenji Fukumizu, Choon Hui Teo, Le Song, Bernhard Schölkopf, Alexander J. Smola, et al. A kernel statistical test of independence. In *NIPS*, volume 20, pages 585–592, 2007.

[40] Valerie Guralnik and Karen Zita Haigh. Learning models of human behaviour with sequential patterns. In *Proceedings of the AAAI Workshop on Automation as Caregiver*, pages 24–30, 2002.

[41] Gerhard K. Heilig. World urbanization prospects: The 2011 revision. *United Nations, Department of Economic and Social Affairs (DESA), Population Division, Population Estimates and Projections Section, New York*, 2012.

[42] Petter Holme. Congestion and centrality in traffic flow on complex networks. *Advances in Complex Systems*, 6(02):163–176, 2003.

[43] De-An Huang, Amir-massoud Farahmand, Kris M Kitani, and J. Andrew Bagnell. Approximate maxent inverse optimal control and its application for mental simulation of human interactions. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

[44] Tsuyoshi Idé and Sei Kato. Travel time prediction using Gaussian process regression: A trajectory-based approach. In *Proceedings of the SIAM International Conference on Data Mining*, pages 1185–1196, 2009.

[45] Taylan Ilhan, Seyed M. R. Iravani, and Mark S. Daskin. The adaptive knapsack problem with stochastic rewards. *Operations Research*, 59(1):242–248, 2011.

[46] Shan Jiang, Joseph Ferreira, and Marta C. González. Clustering daily patterns of human activities in the city. *Data Mining and Knowledge Discovery*, 25(3):478–510, 2012.

[47] Rico Jonschkowski, Sebastian Höfer, and Oliver Brock. Patterns for learning with side information. *arXiv preprint arXiv:1511.06429*, 2015.

[48] Mohamed Amine Kafi, Yacine Challal, Djamel Djenouri, Messaoud Doudou, Abdelmadjid Bouabdallah, and Nadjib Badache. A study of wireless sensor networks for urban traffic monitoring: Applications and architectures. *Procedia Computer Science*, 19:617–626, 2013.

[49] Yiannis Kamarianakis and Poulicos Prastacos. Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches. *Transportation Research Record: Journal of the Transportation Research Board*, 1857(1):74–84, 2003.

[50] Slava Kisilevich, Florian Mansmann, Mirco Nanni, and Salvatore Rinzivillo. *Spatio-temporal clustering*. Springer, 2010.

[51] John Krumm and Eric Horvitz. Predestination: Inferring destinations from partial trajectories. In *Proceedings of the 2006 UbiComp*, pages 243–260. Springer, 2006.

[52] Martin Kulldorff. A spatial scan statistic. *Communications in Statistics – Theory and Methods*, 26(6):1481–1496, 1997.

[53] Akshat Kumar, Sudhanshu Shekhar Singh, Pranav Gupta, and Gyana R. Parija. Near-optimal nonmyopic contact center planning using dual decomposition. In *Proceedings of the ICAPS*, 2014.

[54] Sébastien Lahaie. Kernel methods for revealed preference analysis. In *Proceedings of the ECAI*, pages 439–444, 2010.

[55] Truc Viet Le, Siyuan Liu, and Hoong Chuin Lau. Reinforcement learning framework for modeling spatial sequential decisions under uncertainty. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*, pages 1449–1450. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[56] Truc Viet Le, Siyuan Liu, Hoong Chuin Lau, and Ramayya Krishnan. A quantitative analysis of decision process in social groups using human trajectories. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems*, pages 1425–1426. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[57] Truc Viet Le, Siyuan Liu, Hoong Chuin Lau, and Ramayya Krishnan. Predicting bundles of spatial locations from learning revealed preference data. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1121–1129. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[58] Truc Viet Le and Minh Thap Nguyen. An empirical analysis of a network of expertise. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1387–1394. ACM, 2013.

[59] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[60] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 593–604. ACM, 2007.

[61] Xiaolei Li, Jiawei Han, Jae-Gil Lee, and Hector Gonzalez. Traffic density-based discovery of hot routes in road networks. In *Advances in Spatial and Temporal Databases*, pages 441–459. Springer, 2007.

[62] Siyuan Liu, Ce Liu, Qiong Luo, Lionel M. Ni, and Ramayya Krishnan. Calibrating large scale vehicle trajectory data. In *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*, pages 222–231. IEEE, 2012.

[63] Siyuan Liu, Yunhuai Liu, Lionel M. Ni, Jianping Fan, and Minglu Li. Towards mobility-based clustering. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 919–928. ACM, 2010.

[64] Siyuan Liu, Qiang Qu, and Shuhui Wang. Rationality analytics from trajectories. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):10, 2015.

[65] Siyuan Liu, Yisong Yue, and Ramayya Krishnan. Adaptive collective routing using Gaussian process dynamic congestion models. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 704–712, 2013.

[66] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1010–1018. ACM, 2011.

[67] Jaakko Luttinen and Alexander Ilin. Efficient Gaussian process inference for short-scale spatio-temporal modeling. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 741–750, 2012.

[68] Haiping Ma, Huanhuan Cao, Qiang Yang, Enhong Chen, and Jilei Tian. A habit mining approach for discovering similar mobile users. In *Proceedings of the 21st International Conference on World Wide Web*, pages 231–240. ACM, 2012.

[69] Nick Malleson and Martin A. Andresen. Spatio-temporal crime hotspots and the ambient population. *Crime Science*, 4(1):1–8, 2015.

[70] Wesley Mathew, Ruben Raposo, and Bruno Martins. Predicting future locations with hidden Markov models. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 911–918. ACM, 2012.

[71] Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

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

[73] Ayan Mukhopadhyay, Chao Zhang, Yevgeniy Vorobeychik, Milind Tambe, Kenneth Pence, and Paul Speer. Optimal allocation of police patrol resources using a continuous-time crime model. In *Proceedings of the 7th International Conference on Decision and Game Theory for Security*, pages 139–158. Springer, 2016.

[74] Marion Neumann, Kristian Kersting, Zhao Xu, and Daniel Schulz. Stacked Gaussian process learning. In *Proceedings of the IEEE International Conference on Data Mining*, pages 387–396, 2009.

[75] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.

[76] Duy Nguyen-Tuong, Jan R. Peters, and Matthias Seeger. Local Gaussian process regression for real time online model learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1193–1200, 2009.

[77] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[78] Judea Pearl. Causality: Models, reasoning and inference. *Econometric Theory*, 19(675-685):46, 2003.

[79] 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. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[80] Walt L. Perry. *Predictive policing: The role of crime forecasting in law enforcement operations*. Rand Corporation, 2013.

[81] Jorge Poco, Harish Doraiswamy, Huy Vo, João L. D. Comba, Juliana Freire, Cláudio Silva, et al. Exploring traffic dynamics in urban environments using vector-valued functions. *Computer Graphics Forum*, 34(3):161–170, 2015.

[82] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[83] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

[84] Constantin A. Rothkopf and Christos Dimitrakakis. Preference elicitation and inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 34–48. Springer, 2011.

[85] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 101–103, New York, NY, USA, 1998. ACM.

[86] Adam Sadilek and John Krumm. Far out: Predicting long-term human mobility. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2012.

[87] Paul A. Samuelson. Consumption theory in terms of revealed preference. *Economica*, pages 243–253, 1948.

[88] Jagan Sankaranarayanan, Hanan Samet, and Houman Alborzi. Path oracles for spatial networks. *Proceedings of the International Conference on Very Large Databases*, 2(1):1210–1221, 2009.

[89] Ralf-Peter Schäfer, Kai-Uwe Thiessenhusen, Elmar Brockfeld, and Peter Wagner. A traffic information system by means of real-time floating-car data. In *ITS World Congress 2002*, 2002.

[90] Amnon Shashua, Ron Zass, and Tamir Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. In *Proceedings of the European Conference on Computer Vision*, pages 595–608, 2006.

[91] Konsta Sirvio and Jaakko Hollmén. Spatio-temporal road condition forecasting with Markov chains and artificial neural networks. In *Hybrid Artificial Intelligence Systems*, pages 204–211. Springer, 2008.

[92] Edward Snelson and Zoubin Ghahramani. Local and global sparse Gaussian process approximations. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 524–531, 2007.

[93] John Snow. *On the mode of communication of cholera*. John Churchill, 1855.

[94] Matthew J. Sobel and Wei Wei. Myopic solutions of homogeneous sequential decision processes. *Operations Research*, 58(4-part-2):1235–1246, 2010.

[95] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 374–383, 2006.

[96] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[97] Michael Szenberg, Lall B. Ramrattan, and Aron A. Gottesman. *Samuelsonian economics and the twenty-first century*. Oxford University Press, 2006.

[98] Waldo R. Tobler. A computer movie simulating urban growth in the Detroit region. *Economic Geography*, pages 234–240, 1970.

[99] Yibing Wang and Markos Papageorgiou. Real-time freeway traffic state estimation based on extended Kalman filter: A general approach. *Transportation Research Part B: Methodological*, 39(2):141–167, 2005.

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

[101] Wilhelmina Adriana Maria Weijermars. *Analysis of urban traffic patterns using clustering*. University of Twente, 2007.

[102] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[103] Jeremy M. Wilson and Alexander Weiss. A performance-based approach to police staffing and allocation. *U.S. Department of Justice Office of Community Oriented Policing Services*, 2012.

[104] Yuanchang Xie, Kaiguang Zhao, Ying Sun, and Dawei Chen. Gaussian processes for short-term traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, 2165:69–78, 2010.

[105] Kai Yu and Wei Chu. Gaussian process models for link analysis and transfer learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1657–1664, 2008.

[106] Morteza Zadimoghaddam and Aaron Roth. Efficiently learning from revealed preference. In *Internet and Network Economics*, pages 114–127. Springer, 2012.

[107] Chao Zhang, Manish Jain, Ripple Goyal, Arunesh Sinha, and Milind Tambe. Learning, predicting and planning against crime: Demonstration based on real urban crime data. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1911–1912. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[108] Chao Zhang, Albert Xin Jiang, Martin Short, P Jeffrey Brantingham, and Milind Tambe. Towards a game theoretic approach for defending against crime diffusion. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent systems*, pages 1355–1356. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[109] Daqing Zhang, Lin Sun, Bin Li, Chao Chen, Gang Pan, Shijian Li, and Zhaohui Wu. Understanding taxi service strategies from taxi GPS traces. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):123–135, 2015.

[110] Yunlong Zhang and Yuanchang Xie. Forecasting of short-term freeway volume with ν-support vector machines. *Transportation Research Record: Journal of the Transportation Research Board*, 2024:92–99, 2008.

[111] Yu Zheng. Methodologies for cross-domain data fusion: An overview. *IEEE Transactions on Big Data*, 1(1):16–34, 2015.

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

[113] 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.

[114] Shao Zhifei and Er Meng Joo. A review of inverse reinforcement learning theory and recent advances. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293–311, June 2012.

[115] Andy Diwen Zhu, Hui Ma, Xiaokui Xiao, Siqiang Luo, Youze Tang, and Shuigeng Zhou. Shortest path and distance queries on road networks: Towards bridging theory and practice. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 857–868, 2013.

[116] Brian D. Ziebart, Anind K. Dey, and J. Andrew Bagnell. Probabilistic pointing target prediction via inverse optimal control. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, pages 1–10, New York, NY, USA, February 2012. ACM.

[117] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.

[118] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Human behavior modeling with maximum entropy inverse optimal control. In *AAAI Spring Symposium: Human Behavior Modeling*, page 92, 2009.