基于海量时空交通数据的学习与预测

（申请清华大f å f 博ë f M论‡ ）

ù { U M ¤ É á o v b
f Ñ ¡ —: Ñ f € /
v ‹
ü Y N ú © Y ˆ

Œ t m

Learning and Prediction over Massive Spatio-Temporal Traffic Data

Dissertation Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Doctor of Philosophy

in

Computer Science and Technology

by

Dong Wang

Dissertation Supervisor : Assitant Professor Jian Li

June, 2017

s 于学Mº 文• (授C 的ô

, º Œh † ã 清华大f s Ý 留 • 用f M 论‡ „ Ä š ，即：

清华大f 拥 在著\ C 法Ä š 范围…f M 论‡ „ • 用C ，v -

包括：（1）ò 获f M„ 研究生必须按f 校Ä š 提¤ f M 论‡ ，f 校可å 采用影印) 印或v Ö 复6 手µ Ý X 研究生 ¤ „ f M 论‡ ；（2）: Y f ŒÑ 研î„ ，f 校可å l 开„ f M 论‡ \ : 资™在图f 馆 资™¤ 等场所› 校… 生阅读，或在校园Q › 校… 生浏È 部 …[1]

, º Ý 证遵ˆ 述Ä š

ÝÝ „Æ„ 论‡ ãã Æ ” ” ˆ遵ˆddÄšš ）

\ 签 ：_____ ü 签 ：_____

å ：_____ å ：_____

X •

随@Ï 济„ Ø速发U，城 º 口持í ž • 城 基础设½Œ¯ ƒ ¹ 量限，这Ù ºì „ å 8 ú L & e † 诸多 ¿，$ v / 在¤ 通拥5 Œ打车难„ î 题 $:% 重 N ¤ 通H ‡ Øý 源消 Œ¯ ƒ 污Ó 等î 题_ « ø ç 引发，这› ò Ï 成:† 城 进 e 发U „ ö 颈

d ö，随@GPSL e 设备Œû ¨ 端” 用« ºì • 泛• 用， Í p 据海量§ 生，包括车辆轨迹，用户L * Œ在¿ 叫车p 据等等 这› 在城 - 6 集0 „ 海量p 据包+ † s Ž 城 „ • 贵á o 这: ú 立ô 好„ z ý ¤ 通û ß（ITS）[1]提› † ° „ : ，Î Ï 轻¤ 通拥5，提Ø ºì 在å 8 通ä - „ 生活质量 ‹ 如，根据¤ 通a ö „ 预测ŒÙ š 路径„ L v ö ô 0 计，ITS可å Ù ºì 推荐ô 好„ 路¿ 图，Î Ä 避拥^ „ 道路 ö，通过预测通ä 求，ITS可å 提M 派ú 车辆e s a › ，Ï ú ß 车空载造成„ ý 源消在，论‡ - ，我ì @重• 用深¦ f ` Œ: 器f ` 算法基Ž ö 空¤ 通p 据ã ³ ITS- „ * 重• 预测î 题

我ì 研究„ 第* î 题/ ž ö 预测在¿ 叫车„ › 我ì 基Ž * ° 颖„ 深¦ f ` Ó „ 提ú * 端0 端„ 框¶ DeepSD 我ì „ ¹ 法只 • 量手å y 征，可ê ¨ Î 历史订单- 发° 复B „ › 模式 d 外，我ì „ 框¶ Ø¦ 灵活Œ可扩U，预测¾ 确（2016t 滴滴算法大赛- 获得† 第2 ）

我ì 研究„ 第Œ* î 题/ ž ö ¤ 通路µ 预测 我ì 提ú † Í 预测城 道路¤ 通¶ µ „ 集成¤ 通¶ µ 预测û ß (ETCPS) 我ì 在¤ 通路µ ö ô • - Â 测0 †$* H „ø s ' ，v 基Ž Â 测提ú †$* 模型 我ì 通过¾ 心集成这 $* 模型e 取得¾ Æ预测Ó œ， û ß ¹ 扩U 0 大Ä 模p 据

我ì 研究„ 第 * î 题/ ù Ž Ù š 路径Œø ” „ ú 发ö ô，0 计司: L v Œ路径所 ö ô 我ì 提ú † * 端0 端框¶ DeepTTE，基Ž 深¦ 递归^ Ï Q Ü ù t a 道路ú 模预测 我ì 虑† ö 空ø s Œ › 可ý 影Í L v ö ô „ 因，取得¾ 确预测Ó œ（2017t DataCastleú L ö ô 预测竞赛- 获得第3 ）

s . Í 深¦ f ` ；: 器f ` ；ö 空¤ 通p 据

Abstract

The rapid economic development enables a continuing expansion of both urban population and volume of vehicles. However, the limitation of urban infrastructure and environment capacity causes inconvenience in people's daily commutes. Such inconvenience manifests in many ways, especially in traffic congestion and difficulty in getting cabs. It leads to low traffic efficiency, high energy consumption and environmental pollution, which have become bottlenecks of urban development.

At the same time, the wide usage of GPS embedded devices and mobile apps has produced a variety of massive data (e.g., vehicle mobility, traffic patterns, and online car-hailing data). The massive data collected in urban spaces contains valuable information about a city. Analyzing such data with machine learning and deep learning methods brings new opportunities for building a better intelligent transportation system (ITS)[1] to alleviate the traffic congestion and improve the human life quality in daily commute. The ITS recommends better route plans to people to avoid the congested roads, based on the prediction of traffic conditions and travel time estimation of given path. Meanwhile, by forecasting the commute demands, the ITS can dispatch the taxis to balance the supply-demand in advance and reduce the gas consumptions of no-load taxis. In this thesis, we focus on solving three important prediction problems in the ITS using spatio-temporal traffic data.

The first problem we study in this thesis is to predict the real-time car-hailing supply-demand. This is one of the most important component of an effective scheduling system. We present an end-to-end framework called *Deep Supply-Demand (DeepSD)* using a novel deep neural network structure. Our approach can automatically discover complicated supply-demand patterns from the car-hailing service data while only requires a minimal amount hand-crafted features. Moreover, our framework is highly flexible and extendable to utilize multiple data sources. Our model has achieved competitive prediction result (No.2 among 1648 teams in Di-tech Algorithm Competition

2016).

The second problem we study in this thesis is the real-time prediction of the traffic condition. For this problem, we propose an *Ensemble based Traffic Condition Prediction System (ETCPS)* for predicting the traffic conditions of any roads in a city, based on the current and historical GPS data collected from floating vehicles. We observe two useful correlations in the traffic condition time series, and propose two different models called *Predictive Regression Tree (PR-Tree)* and *Spatial Temporal Probabilistic Graphical Model (STPGM)* based on the two observations. Our best quality prediction is achieved by a careful ensemble of the two models. Our system provides high-quality prediction and can easily scale to very large datasets.

The third problem we study in this thesis is estimating the travel time estimation for the given path. We propose an end-to-end framework for Travel Time Estimation, called *DeepTTE*. Our model estimates the travel time of the whole path directly, based on deep recurrent neural networks. In our model, we consider the spatial and temporal dependency in the path as well as various factors which may affect the travel time such as the driver's habit, the day of the week, etc. Our model has achieved competitive prediction result (No.3 among 1578 teams in Travel Time Estimation Competition in DataCastle 2017).

Key words: Deep Learning; Machine Learning; Spatio-temporal traffic data

目 U

, 1 章 Introduction 1
1.1 Background and Challenges 1
1.2 Supply Demand Prediction for Online Car-hailing Services 4
1.3 Traffic Condition Prediction 5
1.4 Travel Time Estimation 5
1.5 Organization 6
, 2 章 Preliminary 7
2.1 Tree-based Methods 7
2.1.1 Decision Tree 7
2.1.2 Random Forest 8
2.1.3 Gradient Boosting 8
2.2 Deep Learning 9
2.2.1 Basic Structure 9
2.2.2 Embedding Method 11
2.2.3 Recurrent Neural Network (RNN) 13
2.2.3.1 Long Short-Term Memory (LSTM) 13
2.2.4 Residual Network 14
, 3 章 Related work 16
3.1 Supply Demand Prediction for Online Car-hailing Services 16
3.1.1 Taxi Route Recommendation 16
3.1.2 Taxi Demand Prediction 16
3.2 Traffic Condition Prediction 17
3.3 Travel Time Estimation 19
3.3.1 Road Segment-Based Travel Time Estimation 19
3.3.2 Path-Based Travel Time Estimation 19
3.4 Deep Learning in Spatio-temporal Data 20

and temporal correlation pattern at the same time is not easy.

- The patterns in spatio-temporal data always vary dynamically due to different geographic locations and time intervals. For example, in the morning the car-hailing demand tends to surge in the residential areas whereas in the evening the demand usually tends to surge in the business areas. Furthermore, the supply-demand patterns under different days of a week can be extremely different. Prior work usually builds several sub-models for different days of the week[8–11]. Such implementations, on one hand, make the model tedious, on the other hand, each sub-model only utilizes a small part of data which may suffer from the lack of training data.
- Moreover, the spatial temporal data usually contains multiple attributes. For example, in the online car-hailing supply-demand problem, order data contains attributes such as the timestamp, passenger ID, start location, destination etc, as well as several "environment" factors, such as the traffic condition, weather condition etc. These attributes together provide a wealth of information for prediction. However, it is nontrivial how to use all the attributes in a unified model. Currently, the most standard approach is to come up with many "hand-crafted" features (i.e., feature engineering), and fit them into an off-the-shelf learning algorithm such as logistic regression or random forest. However, feature engineering typically requires substantial human efforts and there is little general principle how this should be done.
- Finally, the spatial temporal data is usually quite massive. For example, in this thesis, the trajectory data we use in the experiment part of the travel time estimation problem (Chapter A.5) contains 9,653,822 trajectories and 1.4 billion GPS records in total. To deal with such massive data efficiently, we usually need to take advantages of big data platforms, such as Hadoop, Spark. Moreover, compared with the traditional machine learning methods, deep learning techniques show great potential for mining massive data. However, to the best of our knowledge, there is no standard deep learning model to deal with such massive, noisy, and multi-attribute spatio-temporal data we mentioned above.

In this thesis, we focus on solving three important prediction problems in the ITS using carefully devised machine learning and deep learning models to address the above challenges. We present the concrete problems and our contributions in the following sections.

1.2 Supply Demand Prediction for Online Car-hailing Services

In Chapter A.3, we study the problem of predicting the real-time car-hailing supply-demand, which is one of the most important component of an effective scheduling system. Our objective is to predict the gap between the car-hailing supply and demand in a certain area in the next few minutes. Based on the prediction, it is possible to balance the supply-demands in advance by dispatching the cars and dynamically adjusting the price. After observing the data, we find that the car-hailing supply-demand varies dynamically due to different geographic locations and time intervals. Furthermore, the supply-demand patterns under different days of a week can be extremely different. It is difficult to predict such heterogeneous data.

Contributions: We present an end-to-end framework called *Deep Supply-Demand (DeepSD)* using a novel deep neural network structure. Our approach can automatically discover complicated supply-demand patterns from the car-hailing service data while only requires a minimal amount hand-crafted features. Moreover, our framework is highly flexible and scalable. Based on our framework, it is very easy to utilize multiple data sources (e.g., car-hailing orders, weather and traffic data) to achieve a high accuracy. We conduct extensive experimental evaluations, which show that our framework provides more accurate prediction results than the existing methods. Our experimental results also show that embedding method can "cluster" the areas with similar supply-demand patterns, and enable different areas to share historical records, which improves data utilization and prediction accuracy.

1.3 Traffic Condition Prediction

In Chapter A.4, we study the problem of predicting the traffic condition of each road after a few minutes, when the current and historical traffic conditions of the road network are given. Ubiquitous location based services enable us to collect a large volume of traffic data from GPS-embedded devices. Our prediction is based on such GPS data. Despite there exist several researches and products for traffic prediction based on the GPS data, most of them only focused on the arterial roads and did not consider the urban roads. After observing the data, we find that by transforming the traffic condition time series into two different forms of time series (expectation-reality gap and first order difference of traffic condition series), the new time series reveal very strong autocorrelations. We hope these observations can provide useful insight in further study of the travel condition prediction problem and related problems.

Contributions: We propose an *Ensemble based Traffic Condition Prediction System (ETCPS)* for predicting the traffic conditions of any roads in a city based on the current and historical GPS data collected from floating vehicles. We have observed two useful correlations in the traffic condition time series, which are the bases of our design. In order to exploit these two correlations for prediction, we propose two different models called *Predictive Regression Tree (PR-Tree)* and *Spatial Temporal Probabilistic Graphical Model (STPGM)*. Our best quality prediction is achieved by a careful ensemble of the two models. Our system provides high-quality prediction and can easily scale to very large datasets.

1.4 Travel Time Estimation

In Chapter A.5, we study the problem of travel time estimation for a given path, driver and start time. Although the problem has been widely studied in the past years, providing an accurate travel time is still a challenging problem. Prior work usually focuses on estimating the travel times of individual road or sub-paths, and then summing up these estimated travel times. However, such approach leads to an inaccurate estimation, since the travel time is not only affected by the traffic condition, but also

affected by the number of road intersections or traffic lights in the path, and the estimation errors for individual road may accumulate. Furthermore, the travel time of a given path for a specific driver is also affected by the driving style of the driver.

Contributions: We propose an end-to-end framework for Travel Time Estimation called *DeepTTE*. Our model estimates the travel time of the whole path directly, based on deep recurrent neural networks, which can easily process variable-length GPS trajecto. In our model, we consider the spatial and temporal dependency in the path as well as various factors which may affect the travel time such as the driver's habit, the day of the week etc. We conduct extensive experiment on a large scale dataset. The experiment result shows that our model significantly outperforms the other existing methods. To the best of our knowledge, this is the first time that the trajectory is taken as a whole to estimate the travel time for arbitrary origins and destinations.

1.5 Organization

The rest of the thesis is organized as follows: In Chapter 1, we provide an introduction to this research work giving the background, challenges and the main contributions of this thesis. In Chapter 2, we provide some background on the machine learning and deep learning methods used in the rest of this thesis. We cover topics such as tree-based models (including decision tree, random forest and gradient boosting), and deep learning (including basic structure, embedding method, recurrent neural network, and residual network). In Chapter A.2, we present a brief overview of the related work. Applications of deep learning techniques in spatio-temporal data are also included. In Chapter A.3, we present an end-to-end framework called DeepSD using a novel deep neural network structure to predict the supply-demand for online car-hailing services. In Chapter A.4, we introduce an Ensemble based Traffic Condition Prediction System (ETCPS) for predicting the traffic conditions of any roads in a city. In Chapter A.5, we present an end-to-end framework for travel time estimation called DeepTTE based on deep recurrent neural networks. Our conclusion and future courses of action are followed in Chapter A.6.

, 2章 Preliminary

In this chapter, we review some preliminary knowledge that is necessary for latter chapters. We introduce some useful models that we use in the thesis, including the tree based models, and deep learning models.

2.1 Tree-based Methods

Tree-based methods are widely used in spatio-temporal data analysis. As we mentioned in the Chapter 1, the spatio-temporal data is usually highly noisy and irregular. The tree-based methods such as random forest and gradient boosting can be easily used to deal with such non-structured data and obtain a reasonable performance. In this section, we briefly introduce the decision tree method, random forest and gradient boosting.

2.1.1 Decision Tree

The decision tree method is a non-parametric supervised learning method used for classification and regression[12]. The goal of decision tree method is to build a model that can determine the best decisions. It predicts the value (class) of a target variable (class) by learning simple decision rules inferred from the data features.

To be more concrete, we assume the data is associated with n features. Each time, we select one feature and partition data into small chunks according to the value of the feature based on the certain criterions. Each chunk represents a node. The feature and split-point are chosen to achieve the best fit. We continue partition these chunks into smaller chunks, until some stopping rule is applied. The common criterions are Gini Index (CART method)[13], Information divergence (ID3 method[12]), Information Gain Ratio (C4.5 method[14]).

2.1.2 Random Forest

Random forest[15] is an ensemble of classification or regression trees. These trees are generated through changing the training set and feature set using the same strategy as bagging. The new training sets are created by re-sampling n times from the original data set. The new feature sets are created by re-sampling m times from the original feature set. Prediction is made by aggregating (majority vote for classification or averaging for regression) the predictions of the ensemble. Random forest generally exhibits a substantial performance improvement over the single tree classifier such as CART and C4.5.

Random forest increases the classification or regression accuracy by decreasing the variance of the classification or regression errors. In another word, it taps on the instability of a classifier or regression. "Instability" of a classifier or regression means that a small change in the training samples may result in comparatively great changes in accuracy[13].

2.1.3 Gradient Boosting

Gradient boosting[16–18] is also an ensemble of classification or regression trees. For a given data set, it builds M models. Each model may be very simple (like decision tree), which we call it weak learner. As a kind of boosting methods, gradient boosting builds the model in a forward stage-wise manner, which is quite different from bagging methods (such as random forest). In each stage, it introduces a weak learner to compensate the "shortcomings" of existing weak learners.

To be more concrete, we define the training data as (x_i, y_i), where $i \in [1, N]$. The loss in using $F(x)$ to predict y on the training data is defined as $L(f) = \sum_{i=1}^{N} L(y_i, f(x_i))$ where $L(f)$ is a derivable loss function. The goal is to minimize $L(f)$ with respect to f, where here $f(x)$ is constrained to be a sum of trees. Forward stage-wise boosting is a greedy strategy. In each stage $1 \leq m \leq M$ of gradient boosting, we assume that there is some imperfect model f_m. The gradient boosting constructs a new weak learner T_m that can maximally reduces the loss $L(f) = \sum_{i=1}^{N} L(y_i, f_m(x_i) + T(x_i))$. Then, the new model is defined as $f_{m+1}(x) = f_m(x) + T_m(x)$. To construct the weak learner $T_m(x)$, we

calculate the negative gradient vector $\partial L(y_i, f(x_i))/\partial f(x_i)$ and fit such vector to obtain $T_m(x)$[13].

2.2 Deep Learning

The deep learning method[19,20] becomes the hottest topics in machine learning area since 2006 and it has already been successfully applied in computer vision, speech recognition, natural language processing etc. The traditional machine learning methods usually require a large quantity of carefully hand-crafted features. However, feature engineering typically requires substantial human efforts and there is little general principle how this should be done. Alternatively, the deep learning methods learn to represent the data by combining the simple features into more sophisticated features with its deep architecture automatically. Thus, the deep learning methods are capable of achieving more accurate results with less human effort.

Currently, most of deep learning architectures focus on the image[21,22], text[23,24] and speech data[25–27]. Compared with such data, the spatio-temporal traffic data has both spatial and temporal patterns and it usually contains multiple attributes. Moreover, we usually have to handle city-level data which is in a very large scale. Due to the characteristic of spatio-temporal traffic data, few work utilizes deep learning methods in spatio-temporal traffic data analysis and prediction. We carefully devise several deep architectures to solve some classical problems using such data (Chapter A.3, A.5)[28]. Our results demonstrate the strength of deep learning in prediction over massive spatio-temporal traffic data. In this section, we first illustrate the basic structure in deep learning and then continue by introducing several useful deep learning architectures in spatio-temporal traffic data learning and prediction.

2.2.1 Basic Structure

A deep learning model is a neural network which consists of many *layers* of *non-linear* information processing stages and *hierarchical architectures*[20,29,30]. Such structure benefits from joint learning of representations with increased levels of ab-

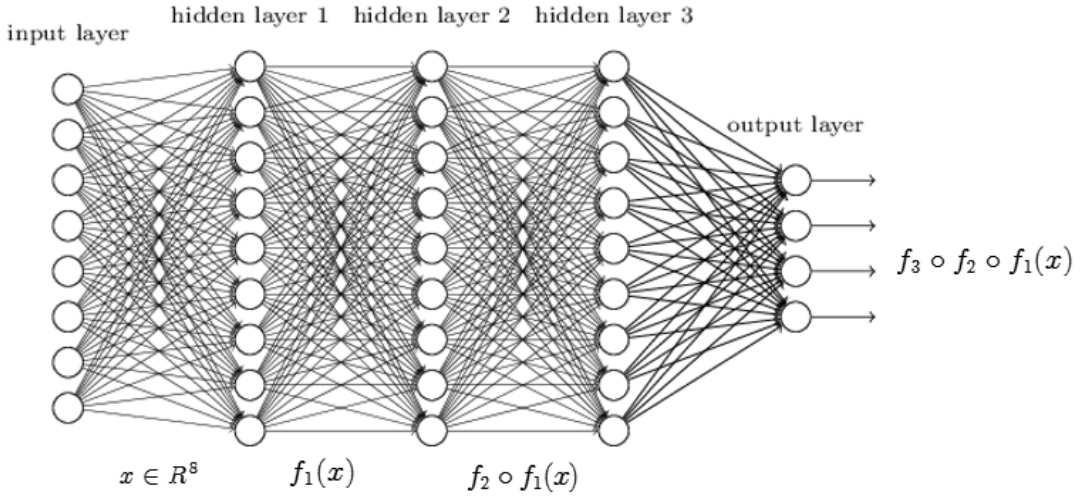

Figure 2.1 Illustration of a neural network (picture from[32])

straction and classification/regression. Formally, a layer can be viewed as a non-linear mapping $f(.)$. For the input data x, the layer maps it into $f(x;\theta)$ where θ is the parameter of this layer. We show a simple example in Fig. 2.1. We use $f_i(.;\theta_i)$ to represent the i-th layer in the model. The lower layers extract the simple features from the original data and the deeper layers combine them into more complicated features[31]. The final output can be represented as the compound function $\hat{y} = f_1 \circ \ldots \circ f_n(x)$. The loss function is denoted as $E(y,\hat{y})$, where y is the label of input x.

We stress that all the parameters are initialized randomly. To train the model, we first calculate the loss function E and update the parameters in the last layer θ_n with corresponding gradient $\frac{\partial E}{\partial \theta_n}$. We then calculate the gradient of θ_{n-1} by the chain rule

$$\frac{\partial E}{\partial \theta_{n-1}} = \frac{\partial E}{\partial \theta_n} \cdot \frac{\partial \theta_n}{\partial f_{n-1}} \cdot \frac{\partial f_{n-1}}{\partial \theta_{n-1}}$$

and update the corresponding parameters accordingly. We call such method *back-propagation*. Moreover, to enhance the learning ability of deep learning models , several improved architectures were devised, such as Convolutional Neural Network (CNN)[33,33], Recurrent Neural Network (RNN)[34] and Residual Neural Network (ResNet)[35]. We introduce three useful architectures in spatio-temporal traffic data analysis in Section 2.2.2 to Section 2.2.4.

2.2.2 Embedding Method

Embedding method is a feature learning technique[36,37]. It is widely used in deep learning models, especially in natural language processing (NLP) tasks[36,38,39]. It is a parameterized function which maps the categorical values to the real numbers.

Specifically, neural networks treat every input as a real value. A simple way to transform categorical values to real numbers is *one-hot representation*[40]. For example, suppose the value of a categorical feature is 3 and the corresponding vocabulary size (highest possible value) is 5. Then, its one-hot representation is $(0, 0, 1, 0, 0)$. However, using such representation can be computationally expensive when the vocabulary size is huge. Moreover, such representation does not capture the similarity between different categories.

The embedding method overcomes such issues by mapping each categorical value into a low-dimensional space (relative to the vocabulary size). For example, the categorical value with one-hot representation equal to $(0, 0, 1, 0, 0)$ can be represented as the form of $(0.2, 1.4, 0.5)$. Formally, for each categorical feature, we build an embedding layer with parameter matrix $W \in R^{I \times O}$. Here I is the vocabulary size of input categorical value and O is the dimension of the output space (which we refer to as the embedding space). For a specific categorical value $i \in [I]$, we use $\mathsf{onehot}(i) \in R^{1 \times I}$ to denote its one-hot representation. Then, its embedded vector $\mathsf{embed}(i) \in R^{1 \times O}$ is equal to $\mathsf{onehot}(i)$ multiply the matrix W, i.e., the i-th row of matrix W. We usually have that $O \ll I$. Thus, even the vocabulary size is very large, we can still handle these categorical values efficiently.

Furthermore, an important property of embedding method is that the categorical values with similar semantic meaning are usually very close in the embedding space[41]. Here we give an example of word embedding, which embeds words into real vector, $W : words \to \mathbb{R}^n$. As shown in the figure 2.2, we visualize the embedded words with t-SNE, a sophisticated technique for visualizing high-dimensional data. The words with similar semantic meaning are very close, such as the words representing numbers are embedded in the left, and the words representing jobs are embedded in the right region. Another important property word embeddings exhibit is that analo-

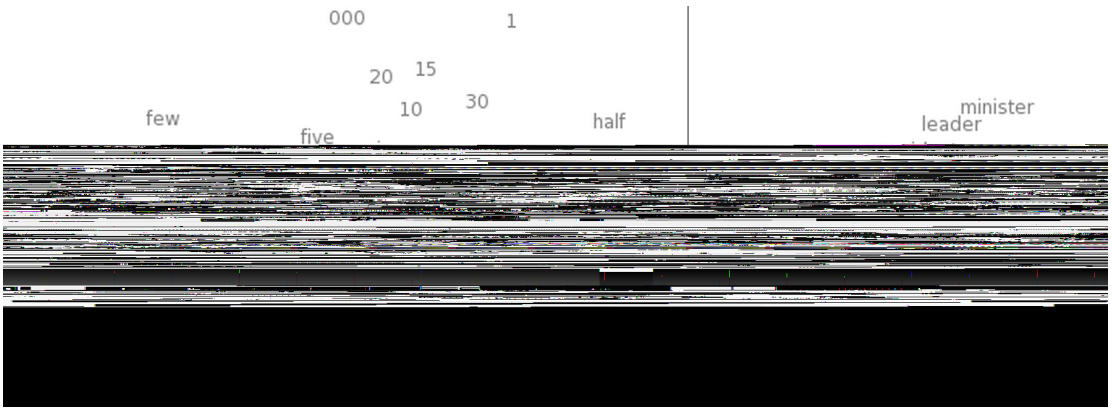

Figure 2.2 T-SNE Visualizations of Word Embeddings. (picture from[41])

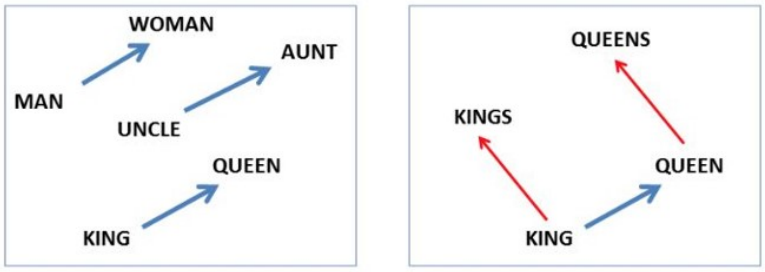

Figure 2.3 Male-female Difference Vector (picture from[42])

gies between words seem to be encoded. For example, the male-female difference vector between “man” and “woman” seems almost the same as that between “king” and “queen”, as shown in Fig 2.3.

In general, these properties are considered as side effects. The neural network does not explicitly map the categorical values with similar semantic meanings to the a similar position in the vector space. These properties more or less popped out of the optimization process. It seems that neural networks can learn better ways to represent data, automatically.

In our problem of supply demand prediction for online car-hailing services in Chapter A.3, we find that if two different areas share similar supply-demand patterns, then their area IDs are close in the embedding space. See Section 4.5 for the details. We stress that the parameter matrix W in the embedding layer is optimized with other parameters in the network. We do not train the Embedding Layers separately.

2.2.3 Recurrent Neural Network (RNN)

Recurrent neural network (RNN) is an artificial neural network which contains an internal state and a directed cycle between units[34]. It is suitable for capturing the temporal dependency and has been used successfully in sequential learning such as the natural language processing, speech recognition, etc[38,43,44]. Especially, an internal state can be viewed as the "memory" of previous time steps. When RNN calculates a new internal state, it captures the temporal dependency with all the previous input sequences. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. Formally, RNN takes a sequence $\{x_t\}$ as input. In each iteration, RNN calculate the "hidden state" (memory)

$$h_t = \sigma(x_t W_x + h_{t-1} W_h + b)$$

where $W_x$fiW_h is the parameters to be learned.

However, vanilla RNN failed in processing long sequences due to vanishing gradient and exploding gradient problems[45]. To overcome such issue, Long Short-Term Memory was developed[45].

2.2.3.1 Long Short-Term Memory (LSTM)

Long Short Term Memory network (LSTM) is a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber[45], and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems, and are now widely used.

An LSTM contains several LSTM units. Each LSTM unit (See Fig. 2.4 for an illustration) contains a memory cell and three gates which are used to control the flow of information in/out of their memory. Mathematically, given the input vector $x = \{x_0, x_1, \ldots, x_n\}$, and denoting the output as $y = \{y_0, y_1, \ldots, y_n\}$, the expected output (the internal states of LSTM) are updated as follows:

$$i_t = \sigma(W_{xi} x_t + W_{yi} y_{t-1} + W_{ci} c_{t-1} + b_i),$$
$$f_t = \sigma(W_{xf} x_t + W_{yf} y_{t-1} + W_{cf} c_{t-1} + b_f),$$

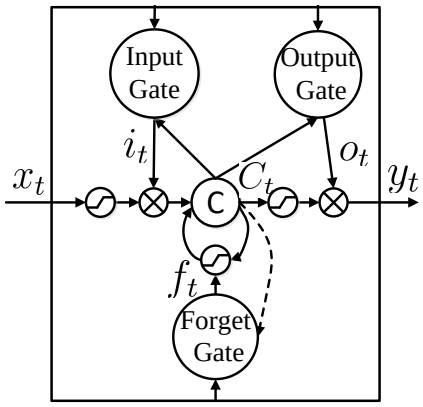

Figure 2.4 LSTM

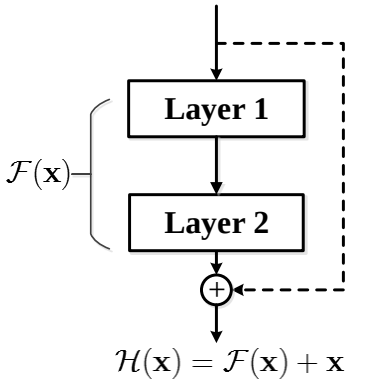

Figure 2.5 Residual Network

$$c_t = f_t \otimes c_{t-1} + i_t \otimes tanh(W_{xc}x_t + W_{yc}y_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{yo}y_{t-1} + W_{co}c_{t-1} + b_o),$$
$$y_t = o_t \otimes tanh(c_t)$$

where σ denotes the logistic sigmoid function and $\otimes$ denotes element-wise multiplication.

2.2.4 Residual Network

Many non-trivial tasks have greatly benefited from very deep neural networks, which reveals that network depth is of crucial importance[46–48]. However, an obstacle to train a very deep model is the gradient *vanishing/exploding* problem, i.e., the gradient vanishes or explodes after passing through several layers during the backpropagation[20,49]. To overcome such issue in deep neural networks, He et al.[35] proposed a new network architecture called the residual network (ResNet). ResNet makes it possible (easier) to train a very deep convolutional neural network successfully.

The residual learning adds the *shortcut connections* (dashed line in Fig. 2.5) and *direct connections* (solid line in Fig. 2.5) between different layers. Thus, the input vector can be directly passed to the following layers though the shortcut connections. For example, in Fig. 2.5, we use $\mathbf{x}$ to denote the input vector and $\mathcal{H}(\mathbf{x})$ to denote the desired mapping after two stacked layers. In the residual network, instead of learning

the mapping function $\mathcal{H}(\mathbf{x})$ directly, we learn the residual mapping $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$ and broadcast $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ to the following layers. It has been shown that optimizing the residual mapping is much easier than optimizing the original mapping[35], which is the key to the success of deep residual network.

, 3 章 Related work

In this chapter, we review the existing related work for the three problems that we study in this thesis respectively. There is a large body of literature on learning and prediction over massive spatio-temporal data and we only mention a few closely related ones. At the end of this chapter, we review the existing work which study the prediction with spatio-temporal data using deep learning.

3.1 Supply Demand Prediction for Online Car-hailing Services

3.1.1 Taxi Route Recommendation

The taxi route recommendation aims to predict the best routes for drivers in order to maximize their utilization. Yuan et al.[9] presented an algorithm to suggest the taxi drivers with locations towards which he/she is most likely to pick up a passenger soon. They used a Poisson model to predict the probability of picking up a passenger for each parking place. In their work, the pick-up locations are fixed in advance. Our work aims to predict the supply-demand gap in every area. Wang et al.[50] investigated the problem of recommending a cluster of roads to the taxi drivers. They used a single hidden layer neural network with carefully selected hand-crafted features. Our work uses a deep neural network with little hand-crafted features. Ge et al.[51] provided a cost-efficient route recommendation algorithm which can recommend a sequence of pick-up locations. They learnt the knowledge from the historical data of the most successful drivers to improve the taxi driver utilization of remaining ones. However, such problem setting is much different from ours.

3.1.2 Taxi Demand Prediction

The taxi demand prediction studies the problem of forecasting the demands in every pick up location. Moreira-Matias et al.[52] combined the Poisson Model and AutoRegressive Moving Average (ARMA) model to predict the demand in each taxi

stand. Again, they only considered the demands in several fixed locations. Moreover, in their work they treated the data in each taxi stand separately. Such implementation suffers from the lack of training data. In a recent work, Chiang et al.[10] proposed a generative model, called Grid-based Gaussian Mixture Model, for modeling spatio-temporal taxi bookings. Their approach was able to predict the demand of taxis in any time interval for each area in the city. Nevertheless, on one hand, they treated the orders in weekdays and weekends separately. On the other hand, in their approach, the total amount of taxi bookings was decided by a Poisson model in advance. When the real-time taxi demand changed rapidly, their approach may lead to a large prediction error.

We stress that prior work only studied the demand prediction but ignored the supply. In the real applications such as taxi route recommendation, taxi dispatching etc, it is important to predict the equilibrium of the supply-demand. Moreover, none of these work studied incorporating the environment data such as the weather or traffic conditions to enhance the prediction accuracy. In Di-tech Prediction Competition 2016①, the champion team proposed an accurate algorithm based on Gradient Boosting Descent algorithm ② and 1534 carefully hand-crafted features. Besides the basic features (such as the area ID, day of week, the previous gap and corresponding statistics), they also considered very detailed features such as average/standard deviation of waiting time, calling time of different passengers during different time intervals, the ratio of car-hailing supply/demand of different areas etc. Despite the proposed model achieves a remarkable performance, designing such features are highly non-trivial and requires enormous amount of human effort.

3.2 Traffic Condition Prediction

Most of prior works use the probabilistic models to predict the traffic conditions. Hunter et al.[53] formulated the traffic condition prediction in the arterial network to a maximum likelihood problem and estimated the travel time distributions based on the

① http://research.xiaojukeji.com/competition/main.action?competitionId=DiTech2016&&locale=en
② https://github.com/Microsoft/LightGBM/

observed route travel times. Yeon et al.[54] estimated traffic conditions on a freeway using Discrete Time Markov Chains (DTMC). However these works assumed that the travel times on different road segments are independent without considering the correlation between the traffic conditions on different roads which may lead to incorrect prediction in the urban area[55].

To capture the correlations between road segments, Hofleitner et al.[56] formulated the transitions between states among adjacent road segments as a dynamic Bayesian network model and predicted the traffic conditions by an EM approach. However, it did not consider the efficiency on the large scale data.Yuan et al.[9] built a landmark graph based on the trajectories of taxis, where each node (entitled a landmark) indicates a road segment each edge indicates the aggregation of taxis' commutes between two landmarks. They formulated the correlations and estimated the edge travel time distributions based on the landmark graph. However, as the landmarks are selected from the top-k frequently traversed road segments, many of road segments with sparse records can not be predicted.

The most related work with our model was proposed by Yang et al.[57]. They proposed an algorithm called STHMM which is a spatio temporal hidden markov model. They further presented an effective method to deal with the sparsity in the data. However, they did not consider the heterogeneity of transition patterns in different time intervals. In our experiment section (Section 5.8), we show that our model outperform STHMM in both the efficiency and accuracy. We stress that Chu et al.[58] considered the transition patterns in different time intervals and proposed a time-vary dynamic network. However their goal is to reveal the causal structure in a ring road system which differs from ours.

Furthermore, we stress two recent related works[5,59]. Wang et al.[59] presented an efficient algorithm to estimate the travel time of any path, based on sparse trajectories generated by taxi in recent time slots and in history, by using the tensor decomposition. Instead of predicting the traffic conditions, they studied the estimation of travel time for given travel paths in the current time slot. Asghari et al.[5] estimated the travel time distributions based on the historical sensor data. As their work studied the algorithm to

find the most reliable route for the travel planning, it has a related but different scope.

3.3 Travel Time Estimation

3.3.1 Road Segment-Based Travel Time Estimation

Estimating travel time has been studied extensively[60–62]. However, these works estimated the travel time of individual road segment without considering the correlations between the roads. Yuan et al.[57] used a spatial-temporal Hidden Markov Model to formalize the relationships among the adjacent roads. Wang et al.[63] improved this work through an ensemble model based on two observed useful correlations in the traffic condition time series. Wang et al.[64] proposed an error-feedback recurrent Convolutional neural network called eRCNN for estimating the traffic speed on each individual road. These studies considered the correlation between different roads. However, they focused on accurately estimating the travel time or speed of individual road segment. The travel time of a path is affected by various factors, such as the number of road intersections and the traffic lights in the path. Simply summing up the travel time of the road segments in the path does not lead to an accurate result[65].

3.3.2 Path-Based Travel Time Estimation

Rahmani et al.[66] estimated the travel time of a path based on the historical data of the path. However, the historical average based model may lead to a poor accuracy. Moreover, as new queried path may be not included in the historical data, it suffers from the data sparse problem. Yuan et al.[9] built a landmark graph based on the historical trajectories of taxis, where each landmark represents a single road. They estimate the travel time distribution of a path based on the landmark graph. However, as the landmarks are selected from the top-k frequently traversed road, the roads with few traveled records can not be estimated accurately. Furthermore, Wang et al.[59] estimated the travel time of the path, based on the sub-trajectories in the historical data. They used the tensor decomposition to complete the unseen sub-trajectory and such method enhance the accuracy effectively. Nevertheless, it still suffers from the data sparsity

problem since there are many sub-trajectories which were visited by very few drivers. Dai et al.[67] proposed a new paradigm for path cost distribution estimation. Given a departure time and a query path, they showed how to select an optimal set of weights with associated sub-trajectories that cover the query path and compute the cost distribution of the query path using the joint distribution. As they focused on estimating the uncertainty of travel cost, it has a related but different scope. In a very recent travel time estimation competition ①, the champion team used a series of standard machine learning models such as the random forest, the multi-layer perceptron, the LASSO etc., as the base estimators. They use Gradient Boosting method to combine the estimation results of different estimators and use the combined result as the final result. However, we stress that in practice, devising many machine learning models is very tedious and hard to maintain. Instead, we only use a single end-to-end framework.

3.4 Deep Learning in Spatio-temporal Data

Recently, the deep learning techniques demonstrate the strength on spatio-temporal data mining problems. An increasing number of researchers studied applying the deep learning technique to prediction problems[68–71]. However, few work studied the prediction with spatio-temporal data using deep learning. Lv et al.[72] studied predicting the traffic flow with deep neural networks. They adopted a *stack autoencoder* to train the network layer by layer greedily. They showed that the deep model is more accurate comparing with the baseline methods. Zhang et al.[73] designed a novel architecture called *DeepST* to predict the crowd flow. Their model learned the spatio-temporal patterns by a sequence of convolutional neural networks. They proposed improved *DeepST* to *ST-ResNet* in[74] by using residual learning to construct a much deeper networks, and proposing a parametric-matrix-based fusion mechanism for modeling both spatial and temporal dependencies. Song et al.[75] built an intelligent system called DeepTransport, for simulating the human mobility and transportation mode at a citywide level. Dong et al.[76] studied characterizing the driving style of differen-

① The competition information and data can be found in https://github.com/DeepTTE/DeepTTE

t drivers by a stacked recurrent neural network. Ma et al.[77] proposes a CNN-based method that learns traffic as images and predicts large-scale traffic speed. To the best of our knowledge, applying the deep learning technique to enhance car-hailing supply-demand prediction accuracy has not been studied so far, and no prior work studies estimating the travel time of the whole path based on the deep learning approach.

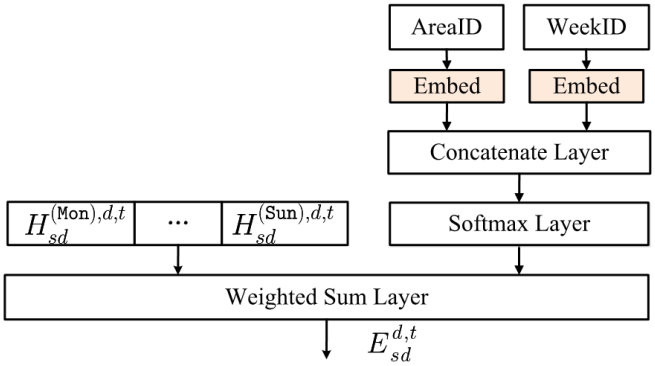

Figure 4.7 Historical supply-demand vector H_{sd}

the real-time supply-demand vector $V_{sd}^{d,t+10}$. If we can estimate $V_{sd}^{d,t+10}$ accurately, we can easily predict the currently supply-demand gap.

In our model, we use the empirical estimations $E_{sd}^{d,t}$, $E_{sd}^{d,t+10}$ and the real-time supply-demand vector $V_{sd}^{d,t}$ to estimate $V_{sd}^{d,t+10}$. We first use the Fully-Connection Layers to project these three vectors onto the same low-dimensional space (in our experiment we fix the dimensionality to be 16). We denote the projected vectors as $\mathsf{Proj}(V_{sd}^{d,t})$, $\mathsf{Proj}(E_{sd}^{d,t})$ and $\mathsf{Proj}(E_{sd}^{d,t+10})$. Instead of estimating $V_{sd}^{d,t+10}$ directly, we estimate the projection of $V_{sd}^{d,t+10}$. We denote the estimated projection as $\hat{\mathsf{Proj}}(V_{sd}^{d,t+10})$ and we have that,

$$\hat{\mathsf{Proj}}(V_{sd}^{d,t+10}) = \mathsf{Proj}(V_{sd}^{d,t}) - \mathsf{Proj}(E_{sd}^{d,t}) + \mathsf{Proj}(E_{sd}^{d,t+10}).$$

Finally, we concatenate $\mathsf{Proj}(V_{sd}^{d,t})$, $\mathsf{Proj}(E_{sd}^{d,t})$, $\mathsf{Proj}(E_{sd}^{d,t+10})$, $\hat{\mathsf{Proj}}(V_{sd}^{d,t+10})$, with a Concatenate Layer and pass it through two Fully-Connected layers FC_{64} and FC_{32}. We use the output of FC_{32} as the output of the extended supply-demand block. See Fig. 4.8 for an illustration.

We explain the reason that we estimate $V_{sd}^{d,t+10}$ in such way. The vector $\mathsf{Proj}(V_{sd}^{d,t}) - \mathsf{Proj}(E_{sd}^{d,t})$ indicates how the real-time supply-demand of $[t-L, t)$ deviates from its empirical estimation. We thus estimate $\mathsf{Proj}(V_{sd}^{d,t+10})$ by adding such deviation to the projection of empirical estimation $\mathsf{Proj}(E_{sd}^{d,t+10})$. Moreover, the projection operation on one hand reduce the dimension of each supply-demand vector from $2L$ to 16. On the other hand, we find that using the projection operation in our experiment makes our

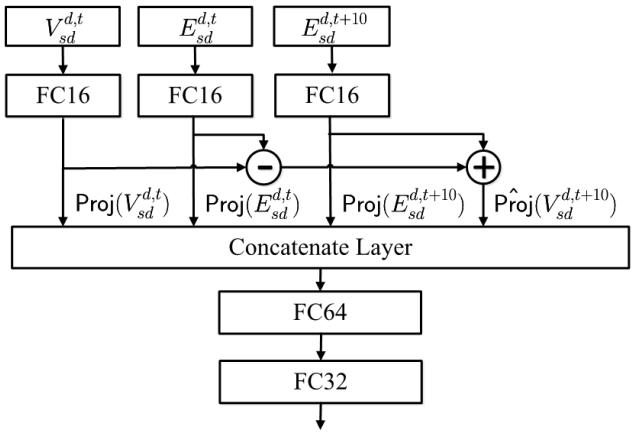

Figure 4.8 Extended Supply-demand Block

model more stable.

4.4.2 Last Call Block and Waiting Time Block

In this section, we present two additional blocks, called the last call block and the waiting time block. Note that the order data contains multiple attributes. However, when calculating the supply-demand vector, we did not consider the attribute $o.pid$. Thus, the supply-demand vector $V_{sd}^{d,t}$ does not contain any "passenger information". From $V_{sd}^{d,t}$, we can not answer the questions such as "how many unique passengers did not get the rides in the last 5 minutes" or "how many passengers waited for more than 3 minutes" etc. However, we find that the passenger information is also very important to supply-demand gap prediction. For example, if many passengers failed on calling the rides or waited for a long time, it reflects that the current demand exceeds the supply significantly which can lead to a large supply-demand gap in the next few minutes. We use the last call block and the waiting time block to provide the passenger information. Both of these two blocks have the same structure as the extended supply-demand block. In another word, we just replace the real-time supply-demand vector $V_{sd}^{d,t}$ in the extended supply-demand vector with the *real-time last call vector* and *real-time waiting time vector*.

For the last call block, we define the *last call vector* as follows.

Definition 6 (Real-time last call vector) For a specific area a at timeslot t in the d-th day, we first pick out the *last call orders* in $[t - L, t)$ for all passengers, (i.e. for a specific passenger *pid*, we only keep the last order sent by *pid*), and denote the order set as ${S_L}^{d,t}$. Then, the real-time last call vector $V_{lc}^{d,t}$ is defined as a $2L$-dimensional vector. We denote the first L dimensions as $V_{A_{lc}}^{d,t}$. For the ℓ-th dimension of $V_{A_{lc}}^{d,t}$, we have that

$$V_{A_{lc}}^{d,t}(\ell) = |\{pid \mid \exists o \in {S_L}^{d,t} \; s.t. \; o \text{ is valid} \wedge o.pid = pid \\ \wedge \; o.ts = t - \ell\}|$$

$V_{Alc}(\ell)^{d,t}$ describes the amount of passengers whose last call is at $t - \ell$ and she/he successfully got the ride. Similarly, we define $V_{B_{lc}}^{d,t}$ which corresponds to the passengers who did not get the rides. □

We explain the reason that we define the real-time last call vector. In our data, we find that if a passenger failed on calling a ride, she/he is likely to send the car-hailing request again in the next few minutes. Especially, the last calls near timeslot t are highly relevant to the supply-demand gap in $[t, t + 10)$.

Based on $V_{lc}^{d,t}$, we can further obtain the empirical last call vector $E_{lc}^{d,t}$ with the same way as we obtain $E_{sd}^{d,t}$. We thus construct the extended real-time last call block with the same structure as the extended supply-demand block.

For the waiting time block, we define the *real-time waiting time vector* $V_{wt}^{d,t} \in R^{2L}$ in the same way as we defining $V_{sd}^{d,t}$ and $V_{lc}^{d,t}$.

Definition 7 (Real-time waiting time vector) For a specific area a at timeslot t in the d-th day, we define the real-time waiting time vector as $V_{wt}^{d,t}$. The ℓ-th dimension in the first part $V_{A_{wt}}^{d,t}$ (first L dimensions) is the total number of passengers who waited for ℓ minutes (from her/his first call in $[t - L, t)$ to the last call) and did get the rides at last. Similarly, we define the second part $V_{B_{wt}}^{d,t}$ which corresponds to the wait time of passengers who did not get the ride.

We thus construct the extended waiting time block with the same structure of the extended supply-demand block.

Figure 4.9 Incorporating extra feature

Finally, we connect the supply-demand block, the last call block and the waiting time block through residual learning, as shown in Fig. 4.6. These three blocks together form the extended order part in the advanced model. We use the extended order part to replace the original order part and we thus obtain the advanced version of DeepSD.

4.4.3 Extendability

Finally, in this section we present the extendability of our model. In real applications, it is very common to incorporate new extra attributes or data sources into the previous model. For example, imagine that we have already trained a model based on the order data and the weather data. Now we obtained the traffic data and we want to incorporate such data to enhance the prediction accuracy. Typically, we have to discard the already trained parameters and re-train the model from beginning. However, our model makes a good use of the already trained parameters. In our model, such scenario corresponds to that we have trained a model with the order block and the weather block.

segment r_i in the t-th interval is defined as the average speed of all the GPS records observed in this road segment during the t-th interval, denoted as o_t^i. Note that for some road segments, there may not exist any GPS record in the t-th interval and thus we can not define the corresponding traffic condition. We explain how we deal with such case in Section 5.8 . Currently, we simply assume o_t^i is well-defined for all i and t. Moreover, we use $\text{Org}^i = \{o_1^i, \ldots, o_T^i\}$ to denote the traffic condition time series of road segment r_i.

Expected Traffic Condition Note that the traffic conditions usually have the "daily pattern". For example, a road segment is usually in a jam during 6:00-9:00 each day whereas from 9:00 to 11:00 it is usually light. For the t-th interval, we define $t \bmod M$ as its *daily index*, i.e., it is the $t \bmod M$-th interval in its corresponding day. For example, if we set $M = 96$, then the 226-th interval represents the time period from 8 : 30-8 : 45 in the third day and its daily index is 226 mod 96 = 34. Let $A_t^i = \{o_{t'}^i | t' \equiv t \bmod M\}$ be the set of traffic conditions observed in road segment r_i during the $t \bmod M$-th interval for all days. For example, in Table 5.2, the 34-th interval is a time period from 8 : 30 to 8 : 45 on the first day. Then, A_{34}^i is the set of traffic conditions of the road segment r_i in all days from 8 : 30 to 8 : 45. We call the mean of A_t^i the *expected traffic condition* of r_i in time interval t, denoted as $a_t^i = \sum_{a \in A_t^i} a / |A_t^i|$. Essentially, the expected traffic condition a_t^i indicates the value that traffic conditions are usually around, in the $t \bmod M$-th interval of a day. We use $\text{Avg}^\text{i} = \{a_1^i, \ldots, a_T^i\}$ to denote the expected traffic condition time series of the road segment r_i. Note that Avg^i is a periodic series and once we have the training data, a_t^i is always available for all $t \in Z$.

Problem Definition Given the historical traffic conditions before time interval T, $\text{Org}^\text{i} = \{o_1^i, \ldots, o_T^i\}$ for all i, our goal is to predict the traffic condition on the $T + 1$-th interval o_{T+1}^i or even longer for each road segment r_i. For convenience, for any t, we use p_t to denote the predicted traffic condition in the time interval t.

Figure 5.3 o_t and o_{t+1} Figure 5.4 g_t and g_{t+1} Figure 5.5 ACF of Di (Org)

5.3 Useful Observations

Most of prior works predict the future tra c conditions directly based on the tra c condition time series. However, it is dicult to extract the patterns in the tra c condition time series Org. We nd that by transforming the Org into two di erent forms of time series, the new time series reveal very strong autocorrelations. We hope these observations can provide useful insight in further study of the travel condition prediction problem and related problems.

Expectation-reality gap The tra c condition time series of the same road segment in each day usually exhibits strong periodic pattern which we refer to as the "daily pattern". We eliminate the daily pattern from the tra c condition series by subtracting the corresponding expected tra c condition from each of the tra c conditions. Speci cally, we set $g_t^i = o_t^i - a_t^i$ and we thus obtain a new series $Gap = \{g_t^i | t = 1, \ldots, T\}$. Intuitively, if $g_t < 0$, it means that the tra c condition in the time interval is more congested than usual. We nd that there exists a strong correlation between g_{t+1} and g_t. Fig. 5.3 and Fig. 5.4 show the scatter diagram of (o_t, o_{t+1}) and (g_t, g_{t+1}) of a speci c road segment respectively. As we can see, by transforming the tra c condition series Org^i to the gap series Gap^i, we essentially extract the "pattern" of the tra c condition series.

First order di erence of tra c condition series We use $\Delta_t^i = o_t^i - o_{t-1}^i$ to represent the rst order di erence of tra c condition series, denoted as $\Delta(Org)$. We use ACF (Auto Correlation Function) to analyze the autocorrelation in the time series of

we thus obtain two time series Org and Avg as defined in Section 5.2. The details are presented in the experiment part (Section 5.8). In Section 5.3, we illustrate two useful observations. Next, in Section 5.5, we use a regression tree based model called PR-Tree to predict the future traffic conditions based on our observed correlations. We further adopt a probabilistic graphical model called STPGM in Section 5.6 which captures both our observations and the correlations between the road segments. Finally, we combine two models in the ensemble stage as shown in Section 5.7. We show that combining two different models enhances the accuracy of the prediction in Section 5.8.

5.5 Predicting The Traffic Condition with PR-Tree

In this section , we define a regression tree based model called PR-Tree to predict the traffic condition of each road segment individually. We first describe the structure of PR-Tree in detail and how we predict the traffic condition on this tree in Section 5.5.1. Then in Section 5.5.2, we present the training algorithm of PR-Tree.

5.5.1 Description of PR-Tree

Recall that the time series Gap shows a strong autocorrelation as we claimed in Section 5.3. We can thus approximate g_{t+1} by an estimation $\hat{g}_{t+1}$ based on g_t and predict the traffic condition in the $t+1$-th interval by $p_{t+1} = a_{t+1} + \hat{g}_{t+1}$ (the expected traffic condition a_{t+1} is always available as we claimed in Section 5.3). From Fig. 5.4, it is reasonable to set $\hat{g}_{t+1} = \theta \cdot g_t$ since the scatter diagram shows a nearly linear correlation. However, we find that the ratio g_{t+1}/g_t varies when g_t takes different values. For example, if g_t is close to -10, g_{t+1} is usually around 1.2 times g_t whereas if g_t is close to -8, g_{t+1} is usually around 1.4 times g_t. Motivated by this, instead of estimating g_{t+1} by $\theta \cdot g_t$, we use a proper function $R(g_t)$ and estimate g_{t+1} by $g_t \cdot R(g_t)$.

Structure To learn a proper function R, we propose a regression tree based model called PR-Tree. Specifically, PR-Tree splits the input space into several subspaces. Each subspace is associated with an output parameter θ. Given the input g_t, we find the subspace corresponding to g_t and return the corresponding θ as $R(g_t)$. Formally, each

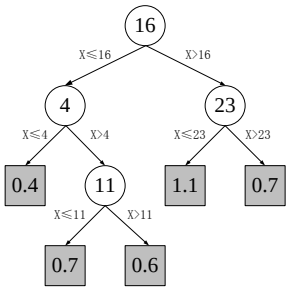

Figure 5.7 An example of PR-Tree

inner node of PR-tree has a splitting value and each leaf node has an output parameter θ. To find the corresponding subspace of g_t, we search on PR-Tree as follows. Initially, the current node is the root of PR-Tree. If g_t is less than or equal to the splitting value of the current node, we search the left child recursively. Otherwise, we search the right child. We perform such search until it reaches a leaf node and returns the corresponding θ on the leaf node as $R(g_t)$. For simplicity, we use R to represent the corresponding PR-Tree.

We show an example of a PR-Tree in Figure 5.7. The PR-Tree contains four inner nodes (the splitting value of these nodes are $\{4, 11, 16, 23\}$), and five leaf nodes (their values are $\{0.4, 0.7, 0.6, 1.1, 0.7\}$). We take $g_t = 5$ as the input. As the splitting value of the root node is 16 and $g_t \leq 16$, we search its left child recursively and finally reach a leaf node with output parameter $\theta = 0.7$.

Prediction To predict the traffic condition in the time interval $t + 1$, we simply set $\hat{g}_{t+1} = R(g_t) \cdot g_t$ and predict o_{t+1} by $p_{t+1} = a_{t+1} + R(g_t)$. Fig. 5.7 shows an example. Given the current traffic condition $o_t = 45$, assuming the expected traffic condition on t and $t + 1$ are $a_t = 40$, $a_{t+1} = 43$, we get $g_t = o_t - a_t = 5$. By taking g_t as the input of PR-Tree, we get $R(g_t) = 0.7$. Then, we estimate o_{t+1} by $a_{t+1} + R(g_t) \cdot g_t = 46.5$.

5.5.2 Training PR-Trees

First, we present the objective for training PR-Trees. Recall that we predict o_{t+1} as $p_{t+1} = a_{t+1} + R(g_t) \cdot g_t$. Given the training set $\text{Org}^{\text{i}} = \{o_1^i, \ldots, o_T^i\}$, our goal is to minimize the squared error $\sum_{t\in[1,T)} (p_{t+1} - o_{t+1})^2$. Equivalently, we need to find an

Algorithm 1 PR-Tree Splitting (Split)

Require: Node *root*, Training sequence TR, cross validation sequence CV

Ensure: Update the PR-Tree.

$e_{TR} = f(TR, \mathsf{out}(TR))$
$e_{min} = \infty$
for $i = 1, \ldots, |TR| - 1$ **do**
    $TR_l \leftarrow$ first i elements in TR
    $TR_r \leftarrow T \backslash TR_l$
    if $f(TR_l, \mathsf{out}(TR_l)) + f(TR_r, \mathsf{out}(TR_r)) < e_{min}$ **then**
        $e_{min} = f(TR_l, \mathsf{out}(TR_l)) + f(TR_r, \mathsf{out}(TR_r))$
        $TR_l^* = TR_l$, $TR_r^* = T \backslash TR_l^*$ ▷ update the best TR_l
    end if
end for
If $e_{min} > e_S - \gamma$ **return**
$root.lc \leftarrow$ a new node corresponds to TR_l^* ▷ split *root*
$root.rc \leftarrow$ a new node corresponds to TR_r^* ▷ split *root*
if $best_{CV} > Q(CV)$ **then** ▷ qualify the splitted PR-Tree
    $best_{CV} = Q(CV)$ ▷ update the global best value
    $\mathsf{Split}(root.lc, TR_l^*, CV)$, $\mathsf{Split}(root.rc, TR_r^*, CV)$
    set the splitting value of *root* as $max_{s \in TR_l^*}\ s.u$ ▷ inner node
else
    $root.lc$ = None, $root.rc$ = None
    set the output value of *root* as $\mathsf{out}(TR)$ ▷ leaf node
    return
end if

optimal PR-Tree (function R^*) that

$$R^* = \underset{R}{\operatorname{argmin}} \sum_{t \in [1,T)} (g_{t+1} - R(g_t) \cdot g_t)^2 \tag{5-1}$$

Our training algorithm is slightly different from the standard regression tree training algorithm. To train the PR-Tree, given the time series Gap $= \{g_1^i, \ldots, g_T^i\}$, we

construct another sequence $S = \{(u, v)|u = g_t, v = g_{t+1}, \forall t = [1, T)\}$. Each element $s \in S$ indicates a pair of values (g_t, g_{t+1}). We use $s.u$ to denote the first value in pair s and $s.v$ to denote its second value. We sort S by increasing order of $s.u$. For any subsequence $S_x \subset S$ and any PR-Tree R, we define the cost of S_x as $Q(S_x) = \sum_{s \in S_x} (s.v - R(s.u) \cdot s.u)^2$. which represents the squared error if we use PR-Tree R to fit the set S_x.

Our training algorithm works as follows. During the training phase, each node corresponds to a subsequence of $S_x \subset S$. For a specific node, if it is an inner node, we use S_l, S_r to denote the corresponding subsequences of its left child and its right child respectively. Then, its splitting value is $\max_{s \in S_l} s.u$. Otherwise, it is a leaf node. We define $f(S_x, \alpha) = \sum_{s \in S_x} (s.v - \alpha \cdot s.u)^2$. The output θ of this leaf node is $\text{argmin}_\alpha \ f(S_x, \alpha)$, denoted as $\mathsf{out}(S_x)$.

Initially, we have a singleton tree. There is the only one node which corresponds to S. We split the PR-Tree recursively. For each node, there is a best splitter S_l^*, i.e.,

$$S_l^* = \underset{S_l}{\text{argmin}} \{f(S_l, \mathsf{out}(S_l)) + \ f(S \backslash S_l, \mathsf{out}(S \backslash S_l)\}.$$

We enumerate the first i elements of S_x as S_l ($S_r = S \backslash S_l$) to search the best splitter S_l^* (line 3 to line 10 in Algorithm 1). Note that since S is sorted and $f(S_l, \alpha)$ is the sum of quadratic terms which is still quadratic. To obtain the best splitter S_l^*, we can maintain the coefficients of $f(S_l, \alpha)$ and the minimum of the quadratic term can be calculated in $O(1)$ time. Each time when we enumerate a new subsequence, we only need to update the coefficients. Thus, we can obtain the best splitter in $O(|S|)$ time efficiently. We denote $S_r^* = S_x \backslash S_l^*$. If $f(S_l^*, \mathsf{out}(S_l^*)) + f(S_r^*, \mathsf{out}(S_r^*)) < f(S_x, \mathsf{out}(S_x)) - \gamma$, we split the current node into two child nodes with subsequences S_l^* and S_r^* respectively where γ is a threshold to be specified (line 12 to line 13). Otherwise, we terminate the recursion (line 11).

The readers may notice that such splitting procedure may cause a serious overfitting problem, i.e., the PR-Tree keeps splitting until each node only contains a very short subsequence. To remedy this issue and reduce the generalization error, we split S into two parts, the training part TR and the cross validation part CV. We use TR to

train PR-Tree, each time when a node is split, we qualify the current PR-Tree on the cross-validation set CV and check whether if $Q(CV)$ decreases (line 14). If the qualification on CV does not decrease, we undo the splitting operation (line19 to line21) and terminate the recursion. Otherwise, we continue the splitting operation (line 15 and line 17) and split its children nodes recursively (line 16). See Algorithm 1 for the pseudo code.

5.6 Predicting Traffic Condition with STPGM

Despite that the PR-Tree performs well in most of our data (which we show in Section 5.8), it does not consider the correlations between the road segments. Some roads are easily affected by its neighbors, the congestions of its neighbors usually lead to the congestion of itself in the next few time intervals. For such roads, PR-Tree does not perform well. Motivated by this, we propose a probabilistic graphical model called STPGM which is used in combination with the PR-Tree in our system.

We first construct a *spatial temporal probabilistic graph (STPG)* G_p which corresponds to a road network G. If a vehicle can travel from the road segment r_i to the road segment r_j (or from r_j to r_i) directly, we say that r_i and r_j are adjacent. We construct a vertex v_i in G_p which corresponds to a road segment r_i in G. We add an edge between v_i and v_j if and only if the road segments r_i and r_j are adjacent. For a specific v_i, we use $Neib(v_i)$ to denote all the adjacent vertices of v_i. Intuitively, the adjacent road segments affect each other much more significantly than the other road segments. Thus, each edge in G_p represents a "strong effectiveness" in the road network.

5.6.1 States of STPGM

We first discretize the traffic conditions into different states. Recall that as we claimed in Section A.4.1, the traffic conditions and the transition patterns are very different not only at different road segments, but also at different time intervals. However, for a specific road segment, we find that the traffic conditions and transition patterns are usually similar for the time intervals with the same daily index. For example, if the traffic is congested in $8:00$, it usually stays congested in next several time intervals.

However, if the traffic is congested in 10 : 00, the traffic becomes light in the next few minutes with a large probability. Motivated by this, we consider different time intervals separately and use the same state sets for the time intervals with the same daily index.

For a specific road segment r_i, instead of clustering all of its traffic conditions in series Org^{i} (which are widely used in the prior works[9,56,57,88]), we consider the traffic conditions under different daily index separately. Formally, we consider a specific daily index $l \in [M]$. Recall that $A_l^i = \{o_t^i | t \equiv l \bmod M\}$. We cluster the traffic condition set A_l^i into k clusters with K-Mediods where k is a parameter to be specified (see Section 5.8 for details). For example, if the daily index l corresponds to 8 : 30-8 : 45 in a day, then we cluster the traffic conditions for all days during 8 : 30-8 : 45. We use the center $c_{x,l}^i$ of each cluster to represent a state, and denote the set of the centers as $C_l^i = \{c_{1,l}^i, \ldots, c_{k,l}^i\}$. The state of the traffic condition in the time interval t is represented by its nearest center in $C_{[t \bmod M]}^i$, denoted as s_t^i. We show an example of a random selected road segment r_i where $C_{25}^i = \{44, 48, 52, 58\}$ and $C_{74}^i = \{15, 25, 32, 38\}$ (km/h). The time interval 25 corresponds to 6 : 00-6 : 15 where the traffic is usually light and the time interval 74 corresponds to 18 : 30-18 : 45 where the traffic is usually heavy.

5.6.2 Parameter Learning

We predict the traffic condition of a specific vertex (corresponds to a road segment) v_i based on the historical traffic conditions of itself and its neighbors. We assume that the traffic condition of v_i in the time interval $t + 1$ is only related with the traffic conditions of v_i and $Neib(v_i)$ in the time interval t.

Formally, consider a vertex v_i. Let $\{v_i\} \cup Neib(v_i) = \{v_{i_1}, \ldots, v_{i_n}\}$ and the corresponding states in time interval t are $\{c_{x_i,t}^i, c_{x_{i_1},t}^{i_1}, c_{x_{i_2},t}^{i_2}, \ldots, c_{x_{i_n},t}^{i_n}\}$. Our goal is to learn the transition probability for all the possible states in $C_{(t+1) \bmod M}^i$, i.e.,

$$\begin{aligned} & P(s_{t+1}^i = c_{x_i,t+1}^i | s_t^{i_1} = c_{x_{i_1},t}^{i_1}, s_t^{i_2} = c_{x_{i_2},t}^{i_2}, \ldots, s_t^{i_n} = c_{x_{i_n},t}^{i_n}) \\ = & \frac{P(s_{t+1}^i = c_{i,t+1}^{x_i}, s_t^{i_1} = c_{x_{i_1},t}^{i_1}, \ldots, s_t^{i_n} = c_{x_{i_n},t}^{i_n})}{P(s_t^{i_1} = c_{x_{i_1},t}^{i_1}, \ldots, s_t^{i_n} = c_{x_{i_n},t}^{i_n})} \end{aligned} \tag{5-2}$$

For the prediction, it is unnecessary to compute the the denominator, which we show in Section 5.6.3. As for the numerator, the state space in Equation 5-2 explodes exponentially whereas the training data is relatively limited. It is not sufficient to estimate the numerator precisely. Thus, we approximate the numerator of Equation 5-2 by

$$P(s^i_{t+1} = c^{x_i}_{i,t+1}) \prod_{j=1}^{n} P(s^t_{i_j} = c^{x_{i_j}}_{i_j,t} | s^i_{t+1} = c^i_{x_i,t+1}) \tag{5-3}$$

where $P(s^t_{i_j} = c^{x_{i_j}}_{i_j,t} | s^i_{t+1} = c^i_{x_i,t+1})$ indicates that given the observed state in the time interval $t+1$, the probability that the previous state of v_{i_j} is $c^{x_{i_j}}_{i_j,t}$.

We define the indicator function $I(s^i_t, c^i_{x,t})$ which indicates that whether the state of the road segment r_i in the time interval t equals $c^i_{x,t}$. We use $N = \sum_{t' \equiv t \bmod M} I(s^i_{t'}, c^i_{x,t})$ to represent the total days that the state of the road segment r_i in the $t \bmod M$-th interval of each day is $c^i_{x,t}$. Then, we calculate the probability $P(s^i_t = c^i_{x,t})$ by the frequency $P(s^i_t = c^i_{x,t}) = N/D$. Similarly, for the term $P(s^t_{i_j} = c^{x_{i_j}}_{i_j,t} | s^i_{t+1} = c^i_{x_i,t+1})$, we have

$$P(s^t_{i_j} = c^{x_{i_j}}_{i_j,t} | s^i_{t+1} = c^i_{x_i,t+1}) = \frac{\sum_{t' \equiv t \bmod M} (I(s^i_{t'+1}, c^i_{x_i,t+1}) \cdot I(s^{i_j}_{t'}, c^{i_j}_{x_{i_j},t}))}{\sum_{t' \equiv t \bmod M} I(s^{i_j}_{t'+1}, c^{i_j}_{x_{i_j},t+1})}. \tag{5-4}$$

Thus, we get the approximation of the numerator of Equation 5-2.

5.6.3 Prediction

Suppose the traffic conditions of the road network in time interval t are observed. We first construct the states for each road segment r_i. To predict the traffic condition of a road segment r_i, after obtaining the states of v_i and $Neib(v_i)$ in the time interval t, we use Equation 5-2 to infer the probability of each state for v_i in the time interval $t+1$. Then, we select the state with the largest probability as the predicted state and the corresponding cluster center as the predicted traffic condition. Note that as the denominator of Equation 5-2 is a constant value when the states of v_i and $Neib(v_i)$ in the time interval t are given, it is actually unnecessary to compute this denominator.

5.7 Model Extensions

Ensemble We find that in the experiment, the performances of PR-Tree and STPGM differ in different roads. Some roads are rarely affected by their neighbors, such as the arterial roads. For such roads, PR-Tree outperforms STGPM. However, as PR-Tree does not consider the correlations of the roads, STPGM performs better than PR-Tree for the roads which are highly affected by its neighbors, especially the roads that only few GPS records are observed. Our prediction for traffic condition in the $t+1$-th interval is a linear combination of the previous traffic condition o_t^i, the prediction obtained by PR-Tree and STPGM. The weights of the linear combination is obtained by linear regression. We show that in the experiment, by combining the models, our system achieves a higher accuracy for the prediction.

Alternate of the input series In fact, both the PR-Tree and STPGM are the models which capture the correlations in a time series. Recall that in the PR-Tree model, we use the time series Gap as the input. In STPGM, we use the traffic condition time series Org as the input. Essentially, we can use the any time series related with the traffic as the input of both models and predict the traffic condition in a proper way. For example, if we use the Org as the input of a PR-Tree, we actually try to approximate o_{t+1}^i by $o_t^i \cdot \theta(o_t^i)$ and we predict the traffic condition directly use $\theta(o_t^i)$. Similarly, we can use the Gap as the input of STPGM. Besides the proposed two series, we can also use the first order difference of Org (i.e., Diff(Org) as defined in Section 5.3) as our input or the traffic conditions filtered with Kalman filtering. The details are presented in Section 5.8.

5.8 Experiments

In this section, we evaluate the effectiveness and efficiency of the proposed models.

5.8.1 Experiment Setting

Data Set In all experiments, we use the real dataset which consists of GPS records collected from 12,000 taxis from November 1st to December 31st in 2012 ① .The GPS data are map matched[89] [90] to road network ② of Beijing. During our experiment, all the driver id were anonymized by recoding. We evaluate our algorithms on the data of November and December respectively. For each month, we divide the data set into the training set (1st - 24th), and the test set (25th - the last day). We distinguish two cases in our experiments: the standard case and the sparse case. For the standard case, we select 10812 road segments which contains more than 140 GPS records per day in average. In the sparse case, we select 101672 road segments in which the GPS records occurred in more than 10 time intervals per day in average. In all experiments, we focus on the time period from 6 : 00 to 24 : 00 in each day since there are only few GPS records observed during 00 : 00 to 6 : 00.

Measurement We evaluate the performances of our models on the test data set by Mean Absolute Error (MAE), Mean Relative Error (MRE) and Mean Squared Error (MSE), i.e., MAE $= \frac{1}{|E|}\sum_{i=1}^{|E|}\sum_{t=1}^{T}|p_t^i - o_t^i|$, MRE $= \frac{1}{|E|}\sum_{i=1}^{|E|}\sum_{t=1}^{T}|p_t^i - o_t^i|/o_t^i$, MSE $= \frac{1}{|E|}\sum_{i=1}^{|E|}\sum_{t=1}^{T}(p_t^i - o_t^i)^2$. Recall that we evaluate our algorithms on the datasets of November and December respectively. For convenience, for each model, we use the mean of the errors on the two months as the final error. All the experiments are implemented parallelly with Python 2.7 and run on a service on Open Stack (Intel Xeon E312 CPU of 16 cores with 2.1GHz for each core and 32GB memory on Ubuntu 14.04LTS operate system).

5.8.2 Pre-processing

Data Cleaning In the data cleaning phase, we eliminate the GPS records for taxis which slow down or even stop for picking or attracting passengers. We distinguish two cases of such records. One is *boarding*, i.e., the passengers get on or get off the taxi. The other is *balling*, i.e., the taxis slow down or stop to attract guests who need

① This data can be downloaded in http://www.datatang.com/data/45888
② This data can be downloaded in http://www.datatang.com/data/45422

taxis. For the boarding state, the speed of the taxi usually varies sharply in a short time. Therefore, once we detect such sharp variation of the speed, we eliminate such GPS records. To handle the balling state, for a specific road, we check the speeds of all taxis in this road in a specific time t. If the speeds of most taxis are relatively high, only few of the taxis are driving at a very low speed , we think such taxis are on the balling state and we eliminate the corresponding GPS records.

Deal with Sparsity Recall that as we claimed in Section 5.2, some road segments may not contain any GPS record during the time interval t for some $t \in [T]$. Thus, the corresponding traffic condition o_t^i is not defined. To solve this issue, for the road segment r_i, if the GPS record set observed in the time interval t is not empty, we define $\bar{o}_t^i$ as the average speed of the GPS records in the t-th interval. Otherwise, we have $\bar{o}_t^i = -1$. Let $\overline{A_t^i} = \{\bar{o}_{t'}^i | t' \equiv t \bmod M \wedge \bar{o}_{t'}^i \neq -1\}$ indicate the traffic conditions during the t mod M-th interval in each day. We define $\bar{a}_t^i$ as the mean of $\overline{A_t^i}$ and the series Bias $= \{b_t = \bar{o}_t^i - \bar{a}_t^i | \forall \bar{o}_t^i \neq -1\}$. Then, for each pair of adjacent elements in Bias, we perform the linear interpolation to obtain the undefined b_i. For example, if $Bias = \{b_1 = 3, b_4 = 4.5, b_7 = 10.5\}$, we obtain a series $\{b_1 = 3, b_2 = 3.5, b_3 = 4, b_4 = 4.5, b_5 = 6.5, b_6 = 8.5, b_7 = 10.5\}$ after performing linear interpolation. Finally, we have that the traffic condition o_t^i is obtained by $o_t^i = \bar{a}_t^i + b_t$.

5.8.3 Performance Evaluation

Performances of different models We present the evaluations of our models. We first compare our model with the baseline Avg, i.e., predict the traffic condition o_t^i by its expected value a_t^i. Furthermore, in the recent work, Yang et al.[57] proposed STHMM for traffic condition prediction which is based on a spatial temporal hidden markov model. We compare STHMM with our models as well.

The results are shown in Fig. 5.8(a) and Fig. 5.8(b). As we can see, the baseline (Avg) performs worst in both cases. Despite that STHMM outperforms Avg in both cases, both of our models PR-Tree and STPGM perform better than STHMM in our data set. Moreover, in the standard case, PR-Tree performs better than STPGM as shown in Fig. 5.8(a) whereas in the sparse case STPGM performs better. By combining

(a) All models (standard) (b) All models (sparse) (c) PR-Tree (standard)

(d) STPGM (standard) (e) STPGM (sparse) (f) Predict longer

Figure 5.8 Performance Analysis.

PR-Tree and STPGM, our system ETCPS achieves the best performance in both two cases.

Verifying the Observed Patterns Recall that as we claimed in Section 5.7, any time series related with tra c can be taken as the input of both PR-Tree and STPGM, and predict the tra c condition in the proper way. To illustrate the e ects of the observations which we proposed in Section 5.3, we design four di erent experiments with di erent time series and evaluate each experiment on PR-Tree and STPGM respectively. The rst two time series are Org and Gap = Org Avg, as we used in Section 5.5 and Section 5.6. Then, we use the rst order di erence of Org as the input time series, denoted as Di(Org). Thet-th element in Di (Org) is $o_{t+1} - o_t$. Furthermore, since the raw GPS records usually contain the noise such as the GPS drift, we use Kalman ltering to process the tra c condition series Org. We take the rst order di erence of the processed time series as the input as well, denoted as Di(Kal).

We show the experimental results in Fig. 5.8(c), Fig. 5.8(d) and Fig. 5.8(e). Both PR-Tree and STPGM perform badly if we use Org as input directly. However, by using Di (Org) and Gap instead, the performances improve signi cantly which veri es our

on deep recurrent neural networks. The primary contribution of this chapter can be summarized as follows:

- We design an end-to-end framework for Travel Time Estimation, called *DeepTTE*, based on deep recurrent neural networks. We incorporate various factors which may affect the travel time (e.g., the driver, the day of the week, and the time interval, etc.) in a unified model, instead of building several sub-models manually.
- We devise a novel neural network architecture which can easily process variable-length GPS trajectories. Furthermore, by carefully designing the input sequence, our model effectively captures the spatial and temporal dependence in the trajectory simultaneously without much information loss.
- We further extend our model to a *multi-task learning* model by introducing an auxiliary component. The auxiliary component estimates the travel time between each pair of adjacent GPS points which we take as the auxiliary output. We show that the auxiliary component effectively improves the model performance.
- We conduct extensive experiments on a large scale data set which consists of GPS points generated by over 14, 684 taxis collected in one month in Chengdu, China. Our model achieves a high-quality prediction result with the error rate of 12.74% which significantly outperforms several other off-the-shelf machine learning algorithms, as shown in Section 6.4.

This chapter is organized as follows. In Section A.5.2, we formally define our problem and present several preliminaries of our model. In Section A.5.3, we describe our model architecture in detail. We conduct extensive experiments to show the strength of our model in Section 6.4. Finally, we present some related work and conclude this chapter in Section A.2.3 and Section 6.5.

6.2 Problem Definition

Definition 8 (Historical Trajectory) We define a *historical trajectory* T as a sequence of consecutive historical GPS points, i.e., $T = \{p_1, \ldots, p_{|T|}\}$. Each GPS point p_i contains: the latitude ($p_i.lat$), longitude ($p_i.lng$) and the timestamp ($p_i.ts$). Further-

more, for each trajectory we record its corresponding driver ID which we denote as driverID.

We then illustrate the our objective.

Definition 9 (Objective) Given the path S, the corresponding driver ID and the start time, our goal is to estimate the travel time from the source to the destination through S. We assume that the travel path S is specified by the user or generated by the route planing apps. S to a sequence of location points by sampling. Each location is represented as a pair of longitude and latitude.

Remark: In our experiment, we remove the timestamp in the historical trajectories and use such trajectories as the test data. During the training phase, we learn how to estimate the travel time of the given path, based on the historical trajectories as we defined in Definition 10. During the test phase, to make the testing data consistent with the training data, we convert the path S to a sequence of location points by sampling. In this chapter, we do not consider how to optimize the path S.

6.3 Model Architecture

We first describe the architecture of our model as shown in Fig. A.6. Our model consists of four parts: the *attribute* component, the *sequence learning* component, the *residual* component, and the *auxiliary* component.

The *attribute* component processes the attributes of the driver ID, the current day of the week, and the timeslot of the start time. The *sequence learning* component processes the GPS location sequence. The *residual* component utilizes the outputs of the first two components to estimate of the given path. Finally, we introduce an *auxiliary* component which estimates the travel times between consecutive GPS points and helps improve the model performance.

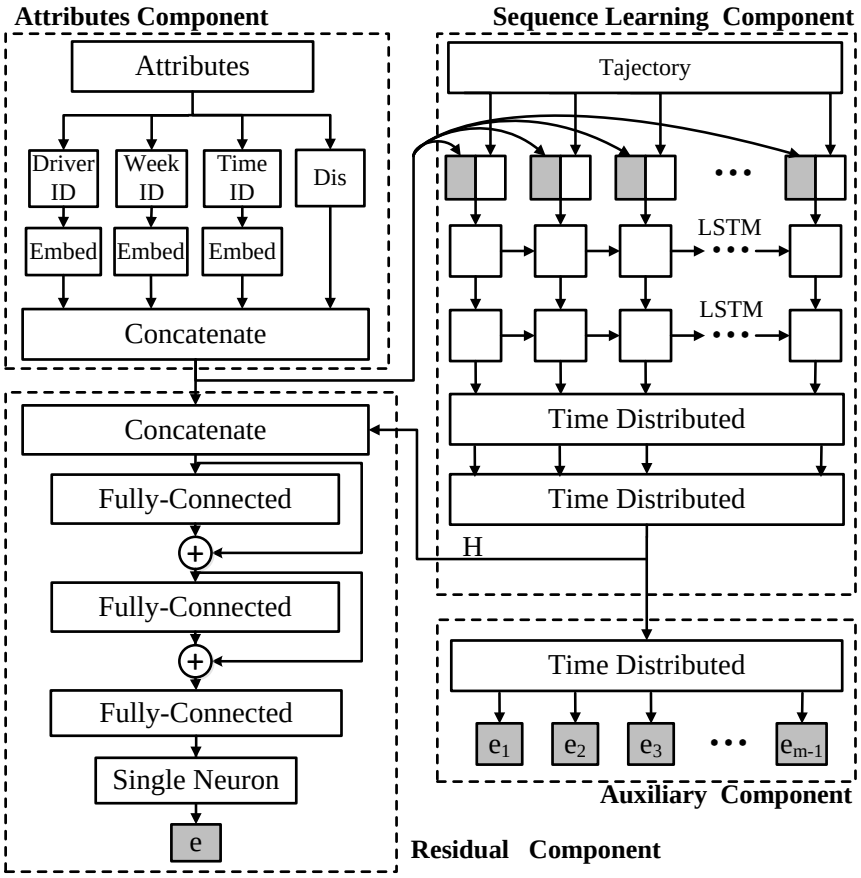

Figure 6.3 Overview of DeepETT.

6.3.1 Attribute Component

In the attribute component, we first process the attributes of the driver ID, the day of the week and the timeslot of travel start① . We use driverID, weekID and timeID to denote these three attributes respectively. Since these attributes are categorical, to feed the attributes into the neural network, we need to transform them into real values. We use the embedding method[38] to process each categorical attribute. The embedding method maps each categorical value $v \in [V]$ to a low-dimensional space $\mathbb{R}^{E\times 1}$ by a matrix $W \in \mathbb{R}^{V\times E}$ (we refer to such space as the embedding space). An important property of embedding method is that the categorical values with similar semantic meaning are usually very close in the embedding space. Thus, the embedding method helps us dis-

① We divide one day into 1440 timeslots.

cover the similarities in the data[38]. In our case, for example, for some specific paths, driving at 7:00 A.M. usually takes similar time with that at 18:00 P.M. since they are both peak hours. Thus, we use the embedding method to reduce the computational cost and help us discover such similarities in the data automatically.

We further consider the attribute of *travel distance*. We denote $\Delta d_{p_a \to p_b}$ as the travel distance from GPS point p_a to p_b, i.e., $\Delta d_{p_a \to p_b} = \sum_{i=a}^{b-1} \mathsf{Dis}(p_i, p_{i+1})$ where Dis is the geographic distance between two GPS points. Then, we concatenate the embedded vectors of driverID, weekID and timeID together with the travel distance $\Delta d_{p_1 \to p_n}$ to form the output of the attribute component, which we denote as *attr*.

6.3.2 Sequence Learning Component

In this part, we demonstrate how the sequence learning component extracts the compressed information from the trajectory using a novel architecture.

Recall that each trajectory T is represented by a sequence of GPS points $\{p_1, \ldots, p_{|T|}\}$. Since different trajectories have different lengths, to handle the sequences with variable lengths, we propose two candidate processing methods.

6.3.2.1 Sampling Method

We randomly sample each trajectory to a fixed length m, as shown in Fig. 6.4. We denote the indices of the sampled points as an ordered list L and the sampled trajectory as T'. Furthermore, to make sure that the source and the destination are included in T', we have that $L_1 = 1$ and $L_m = |T|$. Thus, the sampled trajectory T' can be represented as $\{p_{L_1}, \ldots, p_{L_m}\}$.

Recall that the trajectory contains both the temporal and spatial dependencies. To capture the spatio-temporal dependency, we use two stacked LSTM layers to process the GPS point sequence. A direct way is to take the longitude and the latitude of point p_{L_i} to the LSTM layers at each time step i. However, discovering the relative position and the distance between two GPS points is not easy for neural networks. Instead, for each time step i, we concatenate the coordinates of two consecutive points p_{L_i}, $p_{L_{i+1}}$, the travel distance between p_{L_i} and $p_{L_{i+1}}$ (which we denote as $\Delta d_{p_{L_i} \to p_{L_{i+1}}}$) and the output

Figure 6.4 Sampling Points from Trajectory.

feature vector attr obtained from the attribute component. We use such concatenation $x_i = (lat_i; lon_i; lat_{i+1}; lon_{i+1}; d_{p_{L_i} p_{L_{i+1}}}; attr)$ as the input of LSTM layers.

We pass the input sequence to two stacked LSTM layers and obtain the output sequence $fy_1; y_2; : : : ; y_{m\ 1}g$ Then, we use two stacked time-distributed (fully-connected) layers to map each y_i to a hidden state vector h_i. A time-distributed fully-connected layer takes a sequence of vectors and maps each vector to an output vector using the same mapping function. We nally concatenate the hidden state sequence $fh_1; h_2; : : : ; h_{m\ 1}g$ into a vector H, where H indicates the representation vector of the trajectory.

6.3.2.2 Pooling Method

Alternatively, since the recurrent neural network can process the variable length sequences, we can also use the original trajectory T directly without sampling. In such implementation, instead of using the concatenate layer, we use a pooling layer as our merge layer, i.e., $H = \frac{1}{jTj\ 1} \sum_{i=1}^{jTj\ 1} h_i$ and obtain the representation vector in the

neuron with linear activation to obtain the estimated travel time of the given path.

6.3.4 Auxiliary Component

In fact, the residual component is already available to estimate the travel time of the given path. However, it only considers the "global information"of the whole trajectory but ignores the travel time of sub-trajectories which we refer to as the "local information". Note that during the training phase, since the time stamps of all the GPS points $p.ts$ are available, we can easily infer the local information of the trajectory.

To utilize the local information, a feasible way is to estimate the travel time between each pair of consecutive GPS points and sum up the travel time of all pairs. However, on one hand, the travel time between the consecutive points is usually very small but has a large variance. Estimating such small values is difficult and leads to a very inaccurate result. On the other hand, since we only care the estimation accuracy of the whole trajectory, it is not necessary to accurately estimate the travel time of each individual pair.

Instead, we introduce an auxiliary component to make use of the local information, and we take the estimated travel time between the point pairs as the "auxiliary output". Formally, we denote the travel time of a sub-trajectory $p_a \rightarrow p_b$ as $\Delta t_{p_a \rightarrow p_b}$. For convenience, suppose we adopt the sampling trick in the sequence learning component. Based on the first three components, the auxiliary component receives the hidden state sequence $\{h_1, \ldots, h_{m-1}\}$ from the sequence learning component. Then it maps the hidden state sequence into a real value sequence $\{e_1, e_2, \ldots, e_{m-1}\}$ using a time-distributed fully-connected layer. Each real value e_i corresponds to the estimated travel time $\Delta t_{p_{L_i} \rightarrow p_{L_{i+1}}}$ where L is the indices of sampled points as we defined in Section 6.3.2. We use the estimated travel time sequence $\{e_1, e_2, \ldots, e_{m-1}\}$ as the auxiliary output of our neural network. The auxiliary output is trained together with the estimated travel time of the whole trajectory. See Section 6.3.5 for more details.

6.3.5 Model Training

Our model is trained end-to-end. We use the *mean absolute percentage error (MAPE)* as our objective function. Formally, suppose we adopt the sampling trick in the sequence learning component. Then, the loss function of sequence learning component is defined as

$$\text{loss}_{seq} = |e - \Delta t_{p_1 \to p_{L_m}}| / \Delta t_{p_1 \to p_{L_m}}.$$

For the auxiliary component, the loss function is the average MAPE loss of each time step, i.e.,

$$\text{loss}_{aux} = \frac{1}{m-1} \sum_{i=1}^{m-1} \frac{|e_i - \Delta t_{p_{L_i} \to p_{L_{i+1}}}|}{\Delta t_{p_{L_i} \to p_{L_{i+1}}} + \epsilon}. \tag{6-1}$$

Note that in Eq. (6-1) we use a small factor ϵ to prevent the exploded loss values when $\Delta t_{p_{L_i} \to p_{L_{i+1}}} \to 0$.

The loss function we used during the training phase is defined as the weighted sum of loss_{seq} and loss_{aux}, i.e.,

$$\text{loss} = \text{loss}_{seq} + \alpha \cdot \text{loss}_{aux} \tag{6-2}$$

where α is the weight factor which is specified in the experiment section. Note that the auxiliary loss is just used to improve the accuracy, we still use loss_{seq} to evaluate our model performance.

Thus, we can train our model by the standard backpropagation and gradient descent method.

6.3.5.1 Discussion

Learning to estimate the travel time of the whole trajectory and the travel time sequence of consecutive location pairs at the same time is one of the advantages of using deep learning model. Even though our original problem is to estimate the travel time of the whole path, we convert it to a *multi-task learning* problem, and optimize

multiple objectives with a shared neural network architecture. This technique allows us to fully utilize the information contained in the training data. Multi-task learning has been widely studied in the computer visions and the natural language processing problems[91–93]. It is unclear how to implement such multi-task learning using traditional machine learning techniques such as the random forest, or gradient boosting, in our setting.

6.4 Experiments

In this section, we report our experimental results on the real world dataset. We first describe the experimental setting and the details of dataset in Section 6.4.1. We then compare our model with several baseline methods in Section 6.4.2. In Section 6.4.3 to Section 6.4.5 we present the effects of different components and parameters.

6.4.1 Experiment Setting

6.4.1.1 Data Description

Our dataset consists of 1.4 billion GPS records of 14864 taxis from 2014/08/03 to 2014/08/30 in Chengdu, China①. Each record contains three attributes: the longitude, the latitude and the corresponding time stamp. During our experiment, all the driver id were anonymized by recoding. The total number of trajectories is $9,653,822$. The shortest trajectory contains only 11 GPS location points (2km) and the longest trajectory contains 128 GPS location points (41km).

We use the last 7 days (from 24th to 30th) as the test set and the remaining ones as the training set.

6.4.1.2 Parameter Setting

We present the parameter settings of different components.

Attribute Component: We embed driverID to $\mathbb{R}^{1\times16}$, weekID to $\mathbb{R}^{1\times3}$ and timeID to $\mathbb{R}^{1\times4}$.

① The dataset and the corresponding code can be downloaded at https://github.com/DeepTTE/DeepTTE

Sequence Learning Component: We set the dimension of each LSTM internal state as 128 and the dimensions of two time-distributed fully-connected layers as 128 and 64 respectively. We use leaky relu function (LReLU)[81] as the activation of the time-distributed fully-connected layers. Moreover, we test our model under different sampling rates (i.e., the number of sampled points) when we adopt the sampling trick. See Section 6.4.3 for more details.

Residual Component: We set the dimension of all three residual layers as 128. The activation functions of the residual layers are all leaky relu function.

Furthermore, in the auxiliary loss in Eq.(6-1), we set $\epsilon = 10$. In Eq. (6-2), we set the weight factor α as 3.0. We fix the batch size of our model as 512 and we adopt *Adam*[82] optimizer with learning rate 0.001 to train our model. Our model is trained by 40 epochs.

To evaluate our model, we use 5-fold cross-validation in the training set. For each fold, we select the best model based on the validation. We thus obtain 5 best models. To estimate the travel time on the test set, we use each selected model to obtain an estimation respectively and average the estimations as our final result. The final estimation is evaluated by MAPE as we mentioned in Section 6.3.5.

6.4.1.3 Experiment Environment

Platform: Our model is trained on the server with one GeForce 1080 GPU. We implement our model with Keras 0.8.2 (Theano backend), a widely used Deep Learning Python library.

6.4.2 Performance Comparison

To demonstrate the strength of our model on estimating the travel time, we compare our model with several popular machine learning methods. Since part of the baseline methods cannot process the sequences with variable lengths, to make a fair comparison, we first sample each trajectory to a fixed length of 30. The methods are shown as follow:

Table 6.1 Performance Comparison of Baseline Methods

Model	MAPE
Gradient Boosting	20.32%
MLP-3 layers	16.17%
MLP-5 layers	15.75%
Vanilla RNN	18.85%
DeepTTE	**13.14**%

1. **G**radient Boosting Decision Tree (GBDT): Gradient Boosting Decision Tree is a powerful and widely used ensemble method[13]. To estimate the travel time using GBDT, we concatenate the all the attributes contained in our attribute component and the input sequence in our sequence learning component. We use the concatenated vector as the input of GBDT. In our experiment, we use XGBoost, a widely used GBDT library[78]. The optimal parameters are achieved by the grid search.
2. **Multi-Layer Perceptron (MLP)**: A multi-layer perceptron is a fully-connected neural network with multiple hidden layers. We test our data with a 3-layer MLP (with 3 hidden layers) and a 5-layer MLP respectively. The input vector of MLP is the same as the input of GBDT. The dimension of each hidden layer is fixed as 128 and the activation is leaky relu.
3. **Vanilla RNN**: We further compare our model with a vanilla RNN architecture. Each time step, the vanilla RNN receives the coordinates of p_{L_i} and $p_{L_{i+1}}$ as well as the corresponding travel distance between them. Similarly, we build an attribute component. We concatenate the output the attribute component and RNN, and we pass the concatenation to a 3-layer perceptron to obtain the estimated travel time.

We show the experiment result in Table 6.1. As we can see, the GBDT results in a large error of 20.32%. We stress that although GBDT is a powerful and widely used method, it cannot capture the temporal dependency in the data. Furthermore, GBDT relies on carefully hand-crafted features. However, extracting useful features from the GPS point sequence is not easy. For vanilla RNN, it considers the temporal

Table 6.2 Performance of Different Sampling Capacity

Sampling Capacity	MAPE	Time (per epoch)
DeepTTE-10	15.45%	674s
DeepTTE-30	13.14%	1729s
DeepTTE-70	13.02%	3879s
DeepTTE-100	**12.74**%	5484s
DeepTTE-Var	12.87%	5841s

dependency between GPS location points but it failed to process the long sequence due to the gradient vanishing problem, as we mentioned in Section A.5.1.

6.4.3 Effects on the Sampling Rate

We compare the performances of DeepTTE when we use different sampling rates (i.e., the length m of sampled trajectories as we defined in Section 6.3.2). We use x to denote the lengths of sampled trajectories and DeepTTE-x to denote the corresponding model. We further compare the performance of DeepTTE if we use the pooling trick, which we denote as DeepTTE-Var. The experiment result is shown in Table 6.2.

The result shows that enlarging the sampling rate increases the estimation accuracy of our model. When the sampling rate is 100, our model achieves the best performance of 12.74%. However, using large sampling rate also increases the training time. For DeepTTE-100, it takes about 1.5 hours to train a single epoch. Choosing a proper sampling rate is a trade-off between the speed and accuracy.

DeepTTE-Var achieves the estimation accuracy of 12.85% which is slightly worse than DeepTTE-100. Although it utilizes all the information of the original trajectory T (recall that the max length of T is 128), it causes the information loss when we simply averaging the hidden state by the mean pooling layer.

6.4.4 Effects of the Auxiliary Component

Recall that in our model, we introduce an auxiliary component to estimate the travel time between consecutive GPS points. To verify the effectiveness of the auxil-

Table 6.3 Effects of Attribute Component

Experiment Setting	MAPE
DeepTTE-30	**13.14**%
Eliminate driverID	13.37%
Eliminate weekID	13.58%
Eliminate both	13.59%

iary component, we eliminate the auxiliary component in DeepTTE-30 and train our model under the same condition. The experiment result shows that the estimation error increases from 13.14% to 13.95% dramatically.

Furthermore, if we directly predict the travel time between the consecutive GPS points and use the summation as our estimation, the MAPE is as high as 28.44%.

6.4.5 Effects of the Attribute Component

To show the effects of the attribute component (driver ID, day of the week), we compare the performance of DeepTTE-30 under three different settings. In the first setting, we eliminate the day of the week in the attribute component. In the second setting, we eliminate the driver ID. In the last setting, we eliminate both two attributes and only keep the start time and the travel distance in the attribute component. The experiment result is shown in Table 6.3.

The result shows that eliminating any attribute in our model leads to a reduction of estimation accuracy. As we can see, dropping out the day of the week increases the estimation loss dramatically. Moreover, if we eliminate the driver ID, the MAPE loss increases from 13.14% to 13.37% which verifies that the driving habits affects the travel time estimation, as we mentioned in Section A.5.1.

6.5 Conclusion

In this chapter, we study estimating the travel time of a given path. We propose an end-to-end framework based on deep recurrent neural networks. Our model effectively

captures the spatial and temporal dependencies in the given path at the same time. Furthermore, our model considers various factors which may affect the travel time such as the drivers, the day of the week etc. We conduct extensive experiments on a very large scale real-world dataset. The experimental result shows that our model achieve a high estimation accuracy and outperforms the other off-the-shelf methods significantly.

, 7 章 Conclusion

Currently, the wide usage of location based services and GPS embedded devices have changed people's life style. For example, people use their smart phones to plan their routes, call for car-hailing services, find the trip partners and search the destinations etc. A variety of massive spatial temporal data is generated routinely (e.g., vehicle mobility, traffic patterns, online car-hailing data and geo-tagged check-in data etc). The rapid expansion of both urban population and volume of vehicles has lead to the commute demands in these cities increase sharply. People suffer the traffic congestion and the difficulty in getting cabs. Analyzing the large volume of location based data brings new opportunities for discovering valuable information. It enables the governments to do the traffic analysis and urban planning, which in turn can alleviate the traffic congestion and difficulty in taking cabs. Motivated by this, an increasing number of papers empowered recently.

In this chapter, we will first summarize the contributions of this thesis, and discuss how the frameworks/systems provided in this thesis can be applied to problems outside the scenarios considered. Then, we will talk about the future research directions of the prediction over massive spatio-temporal traffic data.

7.1 Summary of the Thesis

In this thesis, we focus on learning and prediction over the massive spatio-temporal traffic data.Three specific problems are investigated, including car-hail supply demand prediction, traffic condition prediction and travel time estimation. These three problems are quite related, but different. They are the base of building a better intelligent transportation system (ITS) to alleviate the traffic congestion and improve the user experience in daily commute. The ITS recommend people better route plans to avoid the congestions roads, based on the prediction of traffic conditions and travel time estimation of given path. Meanwhile, by forecasting the commute demands, the

ITS can dispatch the taxis to balance the supply-demand in advance and reduce the gas consumptions of no-load taxis.

In the Chapter A.3, we study the problem of predicting the real-time car-hailing supply-demand. We propose an end-to-end framework called *Deep Supply-Demand (DeepSD)*, based on a novel deep neural network structure. Our approach automatically discovers the complicated supply-demand patterns in historical order, weather and traffic data, with minimal amount of hand-crafted features. We presented two versions of DeepSD: a basic version and an advanced version. The basic version simply use the real-time car-hailing order data whereas the advanced version further incorporate the historical car-hailing data with a more complex structure. We conduct extensive experiments on a real-word dataset from Didi. The experimental results show that our model outperforms the existing methods significantly. Furthermore, our model is highly flexible and extendible. We can easily incorporate new data sources or attributes into our model without re-training.

In the Chapter A.4, we study the effective and scalable methods for traffic condition prediction. We propose an Ensemble based Traffic Condition Prediction System (ETCPS) which combines two novel models called Predictive Regression Tree (PR-Tree) and Spatial Temporal Probabilistic Graphical Model (STPGM). Our model is based on two useful observed correlations in the traffic condition data. Our system provides high-quality prediction and can easily scale to very large datasets. We conduct extensive experiments to evaluate our proposed models. The experimental results demonstrate that comparing with the existing methods, ETCPS is more efficient and accurate. In the future, we plan to infer the traffic conditions by incorporating more features from heterogeneous data sources, such as the weather condition, POI information etc. Next, we will focus on the efficient way to deal with road segments which have extremely sparse trajectory records. Furthermore, we plan to try different ensemble methods to combine the different models in order to enhance the performance of the prediction.

In the Chapter A.5, we study estimating the travel time of a given path. We propose an end-to-end framework for Travel Time Estimation called *DeepTTE* based on

deep recurrent neural networks. Our model effectively captures the spatial and temporal dependencies in the given path at the same time. Furthermore, our model considers various factors which may affect the travel time such as the drivers, the day of the week etc. We conduct extensive experiments on a very large scale real-world dataset. The experimental result shows that our model achieve a high estimation accuracy and outperforms the other off-the-shell methods significantly.

Note that, the frameworks/systems provided in this thesis can be easily applied to problems outside the scenarios we considered. For example, in the Chapter A.3, we proposed an end-to-end framework called *Deep Supply-Demand (DeepSD)* to predict the supply-demand for online car-hailing services. In fact, supply-demand prediction is widely studied in many fields, such as the logistics transportation, the commodity retailing, the photovoltaic power generation. To be more concrete, we take the logistics transportation as an example. The logistics companies, such as UPS, FedEX, and EMS, need to maintain networks of warehouses for each city, to pick-up or deliver packages. Each warehouse is responsible for a certain area. However, the number of packages need to be handled varies dynamically due to different warehouses and time intervals. For example, some large warehouses deliver at least $10,000$ packages each day, whereas some warehouses in small-scale only deliver at most 50 packages. Furthermore, the number of packages need to be handled under different time intervals of a day can be extremely different. Obviously, an accurate prediction of the number of packages to be handled for different warehouses and time intervals helps scheduling the employees efficiently, which can greatly improve productivity and save cost. In fact, such problem is quite similar to the supply-demand prediction for online car-hailing services. Currently, we are cooperating with a major logistics transportation company in China. We help them to predict the number of packages to be handled for different warehouses in each time intervals a month in advance. The framework we propose in the Chapter A.3 seems also quite useful in this project.

7.2 Future Directions

There are still a few aspects which can be further optimized. For example, more data sources should be involved in the prediction framework/system. The spatio-temporal traffic data is affected by the complex environments, especially the unexpected events (e.g. extreme weather, traffic accidents, activities, etc.). Event detection based on the search data or the social network data should be added to the prediction system to improve the prediction accuracy in the exceptional situations.

Another interesting avenue would be to explore the feature vectors we extract from trajectories with recurrent neural network. In the chapter A.5, we discuss using the recurrent neural network to represent the whole GPS trajectories as a high dimension feature vector. Although, in that work we use the vector to estimate the travel time of the path, we think the information that the feature vector contains is far more than that. For example, the feature vector may contain the information of driving style of the driver, habit of the trajectory owner, and traffic condition of the roads that the trajectory passed etc. Based on the trajectory feature vector, we can learn the similarity between different trajectories to detect the social relationships between the owners of trajectories.

Finally, in this thesis, we focus on the prediction in the spatio-temporal data. In general, the prediction results provide a reference for decision making. In the next step, we plan to using deep reinforcement learning models to help make decision in the spatio-temporal field. For example, in chapter A.3, we can design a deep reinforcement learning based architecture to decide how to schedule the drivers or recommend a suitable price for dynamic pricing.

Â 文.

[1] Papadimitratos P, De La Fortelle A, Evenssen K, et al. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. IEEE Communications Magazine, 2009, 47(11).

[2] TTI. Texas Transportation Institute: 2015 Urban Mobility Scorecard and Appendices. https://mobility.tamu.edu/ums/report/, 2015.

[3] Zheng W, Lee D H, Shi Q. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. Journal of transportation engineering, 2006, 132(2):114–121.

[4] Kim W, Natarajan S, Chang G L. Empirical analysis and modeling of freeway incident duration. Proceedings of Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on. IEEE, 2008. 453–457.

[5] Asghari M, Emrich T, Demiryurek U, et al. Probabilistic estimation of link travel times in dynamic road networks. Proceedings of Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2015. 47.

[6] Liu W, Zheng Y, Chawla S, et al. Discovering spatio-temporal causal interactions in traffic data streams. Proceedings of Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011. 1010–1018.

[7] Xu J, Deng D, Demiryurek U, et al. Mining the situation: Spatiotemporal traffic prediction with big data. IEEE Journal of Selected Topics in Signal Processing, 2015, 9(4):702–715.

[8] Hu H, Li G, Bao Z, et al. Crowdsourcing-based real-time urban traffic speed estimation: From trends to speeds. Proceedings of 32nd International Conference on Data Engineering. IEEE, 2016. 883–894.

[9] Yuan N J, Zheng Y, Zhang L, et al. T-finder: A recommender system for finding passengers and vacant taxis. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(10):2390–2403.

[10] Chiang M F, Hoang T A, Lim E P. Where are the passengers?: a grid-based gaussian mixture model for taxi bookings. Proceedings of Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2015. 32.

[11] Lichman M, Smyth P. Modeling human location data with mixtures of kernel densities. Proceedings of Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014. 35–44.

[12] Quinlan J R. Induction on decision tree. Machine Learning, 1986, 1(1):81–106.

[13] Friedman J, Hastie T, Tibshirani R. The elements of statistical learning, volume 1. Springer series in statistics Springer, Berlin, 2001.

[14] Salzberg S L. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. Machine Learning, 1994, 16(3):235–240.

[15] Breiman L. Random forests. Machine learning, 2001, 45(1):5–32.

[16] Friedman J H. Stochastic gradient boosting. Computational Statistics & Data Analysis, 2002, 38(4):367–378.

[17] Friedman J H. Greedy function approximation: a gradient boosting machine. Annals of statistics, 2001. 1189–1232.

[18] Breiman L. Arcing the edge. Technical report, 1997.

[19] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press, 2016.

[20] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553):436–444.

[21] Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012. 3642–3649.

[22] Weng J J, Ahuja N, Huang T S. Learning recognition and segmentation of 3-d objects from 2-d images. Proceedings of Proceedings of the 4th International Conference on Computer Vision. IEEE, 1993. 121–128.

[23] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks. Proceedings of Advances in neural information processing systems, 2014. 3104–3112.

[24] Gers F A, Schmidhuber E. LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Transactions on Neural Networks, 2001, 12(6):1333–1340.

[25] Mesnil G, Dauphin Y, Yao K, et al. Using recurrent neural networks for slot filling in spoken language understanding. IEEE/ACM Transactions on Audio, Speech and Language Processing, 2015, 23(3):530–539.

[26] Waibel A, Hanazawa T, Hinton G, et al. Phoneme recognition using time-delay neural networks. IEEE transactions on acoustics, speech, and signal processing, 1989, 37(3):328–339.

[27] Hinton G, Deng L, Yu D, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 2012, 29(6):82–97.

[28] Wang D, Cao W, Li J, et al. DeepSD: Supply-Demand Prediction for Online Car-hailing Services using Deep Neural Networks. Proceedings of 33nd International Conference on Data Engineering. IEEE, 2017. 243–254.

[29] Arel I, Rose D C, Karnowski T P. Deep machine learning-a new frontier in artificial intelligence research [research frontier]. IEEE Computational Intelligence Magazine, 2010, 5(4):13–18.

[30] Bengio Y, et al. Learning deep architectures for AI. Foundations and trends® in Machine Learning, 2009, 2(1):1–127.

[31] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11):2278–2324.

[32] Nielsen M A. Neural Networks and Deep Learning. Determination Press, 2015.

[33] LeCun Y, Bengio Y, et al. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 1995, 3361(10):1995.

[34] Schmidhuber J. Learning complex, extended sequences using the principle of history compression. Neural Computation, 1992, 4(2):234–242.

[35] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. Proceedings of Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. 770–778.

[36] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model. Journal of machine learning research, 2003, 3(Feb):1137–1155.

[37] Roweis S T, Saul L K. Nonlinear dimensionality reduction by locally linear embedding. science, 2000, 290(5500):2323–2326.

[38] Mesnil G, He X, Deng L, et al. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. Proceedings of INTERSPEECH, 2013. 3771–3775.

[39] Graves A, Jaitly N. Towards End-To-End Speech Recognition with Recurrent Neural Networks. Proceedings of ICML, volume 14, 2014. 1764–1772.

[40] Harris D, Harris S. Digital design and computer architecture. Elsevier, 2012.

[41] Turian J, Ratinov L, Bengio Y. Word representations: a simple and general method for semi-supervised learning. Proceedings of Proceedings of the 48th annual meeting of the association for computational linguistics, 2010. 384–394.

[42] Mikolov T, Yih W t, Zweig G. Linguistic Regularities in Continuous Space Word Representations. Proceedings of Proceedings of NAACL-HLT, 2013. 746–751.

[43] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. Proceedings of Advances in neural information processing systems, 2012. 1097–1105.

[44] Graves A, Mohamed A r, Hinton G. Speech recognition with deep recurrent neural networks. Proceedings of 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, 2013. 6645–6649.

[45] Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Computation, 1997, 9(8):1735.

[46] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. Proceedings of Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015. 1–9.

[47] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556, 2014..

[48] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. 2014, 79(10):1337–1342.

[49] Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks, 2015, 61:85–117.

[50] Wang R, Chow C Y, Lyu Y, et al. Taxirec: recommending road clusters to taxi drivers using ranking-based extreme learning machines. Proceedings of Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2015. 53.

[51] Ge Y, Xiong H, Tuzhilin A, et al. An energy-efficient mobile recommender system. Proceedings of Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2010. 899–908.

[52] Moreira-Matias L, Gama J, Ferreira M, et al. Predicting taxi–passenger demand using streaming data. IEEE Transactions on Intelligent Transportation Systems, 2013, 14(3):1393–1402.

[53] Hunter T, Herring R, Abbeel P, et al. Path and travel time inference from GPS probe vehicle data. NIPS Analyzing Networks and Learning with Graphs, 2009, 12:1.

[54] Yeon J, Elefteriadou L, Lawphongpanich S. Travel time estimation on a freeway using Discrete Time Markov Chains. Transportation Research Part B: Methodological, 2008, 42(4):325–338.

[55] Ramezani M, Geroliminis N. On the estimation of arterial route travel time distribution with Markov chains. Transportation Research Part B: Methodological, 2012, 46(10):1576–1590.

[56] Hofleitner A, Herring R, Abbeel P, et al. Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network. Intelligent Transportation Systems, IEEE Transactions on, 2012, 13(4):1679–1693.

[57] Yang B, Guo C, Jensen C S. Travel cost inference from sparse, spatio temporally correlated time series using markov models. Proceedings of the VLDB Endowment, 2013, 6(9):769–780.

[58] Chu V W, Wong R K, Liu W, et al. Causal structure discovery for spatio-temporal data. Proceedings of International Conference on Database Systems for Advanced Applications. Springer, 2014. 236–250.

[59] Wang Y, Zheng Y, Xue Y. Travel time estimation of a path using sparse trajectories. Proceedings of Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014. 25–34.

[60] Zhong S, Ghosh J. A new formulation of coupled hidden Markov models. Dept. Elect. Comput. Eng., Univ. Austin, Austin, TX, USA, 2001..

[61] Sevlian R, Rajagopal R. Travel Time Estimation Using Floating Car Data. arXiv preprint arXiv:1012.4249, 2010..

[62] Pan B, Demiryurek U, Shahabi C. Utilizing real-world transportation data for accurate traffic prediction. Proceedings of Proceedings of the 2012 IEEE 12th International Conference on Data Mining. IEEE, 2012. 595–604.

[63] Wang D, Cao W, Xu M, et al. ETCPS: An Effective and Scalable Traffic Condition Prediction System. Proceedings of International Conference on Database Systems for Advanced Applications. Springer, 2016. 419–436.

[64] Wang J, Gu Q, Wu J, et al. Traffic Speed Prediction and Congestion Source Exploration: A Deep Learning Method. Proceedings of Proceedings of the 2016 IEEE 16th International Conference on Data Mining. IEEE, 2016. 499–508.

[65] Jenelius E, Koutsopoulos H N. Travel time estimation for urban road networks using low frequency probe vehicle data. Transportation Research Part B: Methodological, 2013, 53:64–81.

[66] Rahmani M, Jenelius E, Koutsopoulos H N. Route travel time estimation using low-frequency floating car data. Proceedings of 16th International Conference on Intelligent Transportation Systems. IEEE, 2013. 2292–2297.

[67] Dai J, Yang B, Guo C, et al. Path cost distribution estimation using trajectory data. Proceedings of the VLDB Endowment, 2016, 10(3):85–96.

[68] Grover A, Kapoor A, Horvitz E. A deep hybrid model for weather forecasting. Proceedings of Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Mining. ACM, 2015. 379–386.

[69] Wu S, Ren W, Yu C, et al. Personal recommendation using deep recurrent neural networks in NetEase. Proceedings of 32nd International Conference on Data Engineering. IEEE, 2016. 1218–1229.

[70] Ding X, Zhang Y, Liu T, et al. Deep learning for event-driven stock prediction. Proceedings of Proceedings of the 24th International Joint Conference on Artificial Intelligence. IJCAI, 2015. 2327–2333.

[71] Mocanu E, Nguyen P H, Gibescu M, et al. Demand forecasting at low aggregation levels using Factored Conditional Restricted Boltzmann Machine. Proceedings of 2016 Power Systems Computation Conference, 2016. 1–7.

[72] Lv Y, Duan Y, Kang W, et al. Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 2015, 16(2):865–873.

[73] Junbo Z, Yu Z, Dekang Q, et al. DNN-Based Prediction Model for Spatio-Temporal Data. Proceedings of Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2016. 92.

[74] Zhang J, Zheng Y, Qi D. Deep spatio-temporal residual networks for citywide crowd flows prediction. Proceedings of Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017.

[75] Song X, Kanasugi H, Shibasaki R. Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. Proceedings of Proceedings of the 25th International Conference on Artificial Intelligence. IJCAI, 2016. 2618–2624.

[76] Dong W, Li J, Yao R, et al. Characterizing Driving Styles with Deep Learning. arXiv preprint arXiv:1607.03611, 2016..

[77] Ma X, Dai Z, He Z, et al. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. Sensors, 2017, 17(4):818.

[78] Chen T, Guestrin C. Xgboost: A scalable tree boosting system. Proceedings of Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016. 785–794.

[79] Zheng X, Liang X, Xu K. Where to Wait for a Taxi? Proceedings of Proceedings of the ACM SIGKDD International Workshop on Urban Computing. ACM, 2012. 149–156.

[80] Wang H, Calabrese F, Di Lorenzo G, et al. Transportation mode inference from anonymized and aggregated mobile phone call detail records. Proceedings of 13th International Conference on Intelligent Transportation Systems. IEEE, 2010. 318–323.

[81] Clevert D A, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289, 2015..

[82] Kingma D, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014..

[83] Srivastava N, Hinton G E, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 2014, 15(1):1929–1958.

[84] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints, 2016, abs/1605.02688.

[85] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 2011, 12:2825–2830.

[86] Leontiadis I, Marfia G, Mack D, et al. On the effectiveness of an opportunistic traffic management system for vehicular networks. Intelligent Transportation Systems, IEEE Transactions on, 2011, 12(4):1537–1548.

•¤¤¤ 通路µ 预测 许多研究h ，o 好„ ¤ 通¶ µ 预测û ß 在提Ø ¤ 通H ‡ ¹ b 起@ó s 重• „ \ 用 ‹ 如，? œ可å 在ù ¤ 通Ä „ 变ô \ ú³ š （‹ 如， lq 汽车通道9 : n 通车道）或ú 设v Ö 道路（‹ 如ž 车道）ö， v \ : 参 当Öì 计 ½å 区ö，_ 可å 土(å 程 提ú ú 议（‹ 如，í ½å 如U 影Í ¤ 通）[7]

•úúLLööôôô 预0 0 计Ù š 路径„ Å L ö ô / 路¿ Ä ü * Œ流量 ¦ - „ * 根，î 题 Æ确0 计Å L ö ô © Žºì ô 好地Ä 路¿，避M 拥挤„ 道路，Î © Ž Ï 轻¤ 通拥5 à N 所 „ 电P 地图Œ在¿ 叫车 ¡ 都在Öì „ ” 用程• - 提› † Å L ö ô 0 计，‹ 如7 L 地图， e 打车，滴滴打车等 当用户 " 0 达î„ 地„ 路¿ ö，地图” 用程• 提› † 多* 选路¿，v - 0 计ú L ö ô （å 及可ý „ v Ö 措½，如Ã 气消 ，费用）› 用户³ š

虽6 这› î 题得0 • 泛„ 研究[3–6]，F 由Ž ø ” ¤ 通p 据„ 海量Ä 模Œ异„ ' ，我ìÍ 6 • b 4 @大量„ 挑战 接 e，我ì 进 e 说 f ` Œ预测这› p 据ö 所遇0 挑战„ y 点

- ¤ 通p 据通8 包+ 空ô （M n ）Œö ô （ö ô 戳）^ ' ，我ì_ v ð : ö 空p 据 这Í ö 空p 据 ö 包+ † 空ô Œö ô „ø s 模式（correlation pattern） ‹ 如， a 道路„ ¤ 通路µ ¶ µ 受v ê « K M „ 路µ „ 影Í ， ö _ 受0 v ø 邻„ › 道路„ 路µ 影Í ö 捕获空ô ø s ' Œö ô ø s 模式v ¹
- ù Ž „ 地 M n Œö ô µ，ö 空¤ 通p 据- „ 模式; / ¨ 变化 ‹ 如，在é ，O 房区„ 打车 求通8 %g 升； 在Z ，F ¡ 区域„ 打车 求往往 ž d 外， h å „ › 求模式可ý 很大î 异 å M„ å \ 通8 区 „ 地 M n，ö ô µ 或 à，+ „ ú à * P 模型[8–11] „ 订单 开处 vú 立很多P 模型，Å / „ 模型变得A ， P 模型都/ 只ý 通过 部 p 据进L 训Ã，这样• 得模型可ý 受0 : O 训Ã p 据„ 困扰
- d 外，ö 空¤ 通p 据通8 包+ 多* ^ ' ‹ 如，在¿ 叫车 求预测î 题 - ，订单p 据包+ ö ô 戳，X ¢ ID，起始M n，î„ 地等多* ^ ' ，å 及诸如¤ 通¶ µ，天气¶ µ 等à “¯ ƒ ”因 这› ^ ' 起: › 预测提

› † 大量á o 6 ，ó 在ß 模型- • 用所 ^ ' v ¹ î M 标Æ„ ¹ 法/ º å 抽取许多y 征（即y 征å 程），v v ¤ Ù › ° 成„ f ` 算法，如逻辑回归或随: 森— 6 ，y 征å 程通8 • 大量„ º › Œ领域å 识， ù Ž 如U 进L y 征抽取à N 没 ，' „ 指ü 原

- ，ö 空¤ 通p 据„ Ä 模通8 ø 当ž 大 ‹ 如，在，‡ - ，我ì 在ú L ö ô 预0 î 题„ ž Œ部 - • 用„ 轨迹p 据; q 包+ 9,653,822* 轨迹Œ14¿* GPS记录 :† H 地处 这样„ 大Ä 模p 据，我ì 通8 •) 用Hadoop，Spark等大型p 据s 台 d 外， ß „ : 器f ` ¹ 法ø Ô，深¦ f ` 技/ > : ú 挖掘海量p 据„ è 大\ › 6 ，据我ì 所å ，v 没 * 标Æ„ 深¦ f ` 模型e 处 我ì b 提0 „ 这Í 大Ä 模 多噪ó Œ多^ ' „ ö 空p 据

在，‡ - ，我ì @重ã ³ ITS- „ * 重• 预测î 题，• 用¾心设计„ : 器f ` Œ深¦ f ` 模型e ” ù 述挑战 我ì 接 e w S Ë Í î 题Œ我ì 在* î 题- „ 贡.

A.1.2 在¿ ë f 务中的› 需预测

在第 A.3章，- ，我ì 研究† 预测ž ö 在¿ 叫车› „ î 题，这/ 在¿ 叫车s 台- * H ¦ û ß „ 重• „ Ä 成部 K 我ì „î 标/ 预测在* e à 钟…Ð 地区„ 在¿ 叫车 ¡ „ › ” 求K ô „ î 距 根据预测，我ì 可å 通过提M ¦ 车辆Œ¨ t ÷ 格e s a › 通过ù p 据„ Â ß ，我ì 发° 汽车› ” 量在 „ 地 M n Œö ô µ …H ° ¨ 变化„ y 征 d 外，h … å P „ › 求 可ý ^ 8 预测这Í 异质p 据/ 很 挑战' ”

.贡. ： 我ì 基Ž Í ° 颖„ 深B ^ Ï Q ÜÓ „ 提ú † Íð : *Deep Supply-Demand(DeepSD)* „ 端0 端框¶ 我ì „ ¹ 法只 • 量„ 手å 提取y 征，1 可å ê ¨ Î 在¿ 叫车 ¡ p 据- 发° 复B „ › 模式 d 外，我ì „ 框¶ / Ø¦ 灵活Œ可扩U „ 基Ž 我ì „ 框¶ ，可å 很¹) 用多* p 据源- 获取„ p 据（‹ 如，在¿ 叫车订单，天气á o Œ¤ 通p 据）e ž ° 提Ø 预测¾¦ 我ì ù 模型进L • 泛„ ž Œ评0 ，ž ŒÓ œh 我ì „ 框¶ 可å 提› Ô ° ¹ 法ô Æ确„ 预测Ó œ

A.1.3 交通路µ 预测

在第A.4章- ，我ì 研究在Ù š 道路Q Ü „ 当M Œ历史¤ 通¶ µ ，预测à 钟 Ï a 道路¤ 通¶ µ „ î 题 基Ž M n ¡ „ • 泛• 用• 我ì ý 够Î GPSL e 式设备- 6 集大量„ ¤ 通p 据 = 管ò Ï X 在很多基Ž GPSp 据„ ¤ 通预测„ 研究Œ§ Á ，F 大部 Å 集- 在« 频AÏ 过„ Ø 速l 路或; r 道路预测 ，很多城 - „ 道路v 在Öì „ 预测K 通过ù p 据„ Â ß ，我ì 发° 原始„ ¤ 通路µ ö ô • 转换:$ Í 形式„ ö ô • （ ° ž î 距Œ 阶î • ） ，° „ ö ô • > : ú † ^ 8 强„ ê ø s ' 我ì 这› Â ß ý 够: 进 e 研究¤ 通路µ 预测及ø s î 题提› . ©

.贡. ： 基ŽÎ 浮¨ 车辆6 集„ 当M Œ历史GPSp 据，我ì 提ú † Í 集成¤ 通¶ µ 预测û ß （ETCPS），用Ž 预测城 道路¤ 通¶ µ 我ì 在¤ 通¶ µ ö ô • - Â ß 0 $* 用„ø s ' ，这/ 我ì í 模型设计„ 基础 基Ž 这$* Â 测0 „ø s ' ，我ì 提ú †$ Íð : 预测 归树（*R-Tree*）Œ时空概率形模型（*STPGM*）„ 模型 我ì 通过 $* 模型¾心t e 取得† 好„ 预测Ó œ 我ì „ û ß 预测Ó œ¾确，v 可å 轻~ 扩U 0 ^ 8 大„ p 据集

A.1.4 ú L 时ô 预0

在第A.5章- ，我ì 研究† Ù š 路径Œ开始ö ô „ ú L ö ô 0 计î 题 å M „ å \ 通8 注Ž 0 计单ì 道路或P 路径„ L v ö ô ，6 这› 0 计„ L v ö ô 进L Œ得0 路径L v ö ô 6 ，ú L ö ô _ 受路径- 道路¤ 叉路口或¢ÿ 灯p 量„ 影Í ，v 路µ „ 0 计误î / ï ，因d 这Í ¹ 法„ 0 计< 可ý v Æ确 d 外，Ù š 道路Œú 发ö ô „ L v ö ô 样_ 受 司: „ ~v 风格影Í

.贡. ： 我ì 提ú † * 端0 端„ ú L ö ô 预测框¶ ，叫Z *DeepTTE* 我ì „ 模型基Ž 深B 循¯ ^ Ï Q Ü ô 接0 计t * 路径„ ú L ö ô 在我ì „ 模型- ，我ì 虑† 路径- „ 空ô Œö ô • 赖' å 及可ý 影Í 影Í ú L ö ô „ Í 因 ，如~v ` ï ， à 等 我ì 在大Ä 模p 据集 进L † • 泛„ ž ŒŒ证 ž ŒÓ œh ，我ì „ 模型> 著 Ž v Ö ° ¹ 法

A.1.5 º 文¶„

论‡ „ v Y 部 ÄÇ 如 ：在第A.1Ó - ，我ì Ë Í ，研究å \ „ Ì o ，挑战Œ; • 贡. 在第A.2Ó - ，我ì 简• Ë Í ø s å \ ，å 及深¦ f ` 技 / 在ö 空p 据- „ ” 用 在第A.3Ó - ，我ì 提ú * : DeepSD„ 端0 端框 ¶ ，基Ž Í ° 颖„ 深B ^ Ï Q ÜÓ „e 预测在¿ 叫车 ¡ „ › 在第A.4Ó - ，我ìË Í † * 集 ¤ 通¶ µ 预测û ß （ETCPS），用Ž 预测城 道路¤ 通¶ µ 在第A.5Ó - ，我ì 提ú † 基Ž 深¦ 循¯ ^ Ï Q Ü „ ð : DeepTTE„ ú L ö ô 0 计„ 端0 端框¶ ，我ì 在第A.6Ó - Ù ú , ‡ „ ; Ó Œ e 计

A.2 相s 工\

在, Ó - ，我ì + 回顾† 我ì 在, 论‡ - 研究„ * î 题„ ° ø s å \ ò X 在大量„ s ö 空p 据„ f ` Œ预测„ ‡ . ，我ì 这里只提 › Æ ø s „ ‡ . 在, Ó + > ，我ì 回顾† ) 用深¦ f ` 研究ö 空p 据预测„ ° å \

A.2.1 在¿¦ f 平ð 的› 需预测

A.2.1.1 ú 租f 路„ 推荐

ú ß 车路¿ 推荐è 在: 司: 预测ý 够 大化车辆) 用‡ „ 路¿ Yuan等[9]提ú † ú ß 车司: 提ú Ö/她 可ýý 够接0 X ¢ M n „ * 算法 Öì • 用泊~ 模型e 预测Ï * \ 车地点接0 X ¢ „ , ‡ 在Öì „ å \ - ，接¢ 地点/ 在固š „ 集 范围… 我ì „ å \ è 在预测所 地区„ › : 口 Wang等º [50] 研† ú ß 车司: 推荐 Ä 道路„ î 题 Öì • 用† * 隐藏B ^ Ï Q Ü Œ¾ 心挑选„ º å 抽取y 征 我ì „ å \• 用† * 只 • 很 „ º å 抽取y 征„ 深B ^ Ï Q Ü Ge等º [51]提› † Í w Ø 成, H Ê„ 路¿ 推荐算法，可å 推荐 û „ 接¢ M n ÖìÎ 成Ÿ „ 司: „ 历史L v 轨迹- f 0 å 识，å 9 „ v Ö 司: „„ ë 司: „ 车辆) 用‡ 6 ，这* î 题设š 我ì „ 区+ 很大

A.2.1.2 ú 租f 需求预测

ú ß 车 求预测î 题研究Ï * 接¢ 地点 求„ 预测î 题 Moreira-Matias等º [52]Ó 泊~ 模型Œê 回归滑¨ s 均（ARMA）模型e 预测Ï * ú ß 车站„ 求 Öì 还/ 只 虑† à * 固š 接¢ 地点（ú ß 车站）„ 求 d 外，在Öì „ å \ - ，Öì + 处 † Ï * ú ß 车站„ p 据 这Í ž ½: O 训Ã p 据 在 近„ 项å \ - ，Chiang等º [10]提ú † Í 生成模型，ð : 基Ž Q 格„ Ø ¯ 混 模型，用Ž ù ö 空ú ß 车预订ú 模 Öì „ ¹ 法ý 够在城 „ Ï * 地区„ û U ö ôô 隔…预测ú ß 车„ 求 6 ， ¹ b ，Öì ù å \ å Œh + „ 订单 开ù 待 另 ¹ b ，在Öì „ ¹ 法- ，ú ß 车预订„ ; 量由泊~ 模型（Poisson）提M³ š 当ž ö ú ß 车 求迅速变化ö ，Öì „ ¹ 法可ý ü ô 大„ 预测误î

我ì 强 å M„ å \ 只研究† 求预测， 忽Æ† › ” 在ú ß 车路¿ 推荐，ú ß 车 ¦ 等ž 际” 用- ，重• „ / 预测› s a d 外，这› å \ 都没 研究 e ¯ ƒ p 据，如天气或¤ 通a ö ，å 提Ø 预测¾¦ 在2016t 滴滴预测大赛- ①，› 队• 用† 梯¦ 提升算法②，v 提ú † 1534* ¾心设计„ y 征 除† 基，y 征（如区号， à ，K M„ › î 距Œø ” „ ß 计p 据）外，还 虑† ^ 8 详Æ„ y 征，如 ö ô µ … X ¢ „ 等待ö ô | 叫ö ô ， 区域„ 汽车› ” / 求Ô‹ 等„ s 均< Œ标Æî = 管Öì „ 模型取得 † > @„ h ° ，F / 设计这› y 征/ ^ 8 ¹ „ ，v • 大量„ º ›

A.2.2 交通路µ 预测

在，，- ，我ì 回顾ø s „ ° å \ 大多p å M„ å \• 用，‡ 模型 e 预测¤ 通¶ µ Hunter等º [53] ; r 路Q - „ ¤ 通¶ µ 预测h : : 大< 6 î 题，v 根据Â ß 0 „ 历史路¿ L v ö ô 0 计L v ö ô Yeon等º • 用 » cö ô l Ñ 夫链(DTMC)，在Ø 速l 路 0 算¤ 通¶ µ [54] 6 ，这› å \ G 设 路µ „ Å L ö ô / ì 立„ ， 虑 道路 „ ¤ 通¶ µ K ô „ø s ' ，这可ý ü ô 城 地区„ 误预测

:† 捕捉路µ K ô „ø s ' ，Hofleitner等º [56] ø 邻路µ K ô „ ¶ 转û

① http://research.xiaojukeji.com/competition/main.action?competitionId=DiTech2016&&locale=en

② https://github.com/Microsoft/LightGBM/

h : :† ¨ 贝叶¯ Q Ü 模型，v 通过EM¹ 法预测†¤ 通¶ µ 6 ，ƒ v 没 虑0 算法在大Ä 模p 据 „ H ‡ Yuan等[9]根据ú ß 车„ 轨迹ú 立† * 地标图，Ï * ， 点（即地标）h : 路µ ，Ï a 边h : $* 地标K ô „ ú ß 车往e 集 Öì • 用地标图h : ø s ' ，v 0 计† 边 „ L v ö ô 6 ，由Ž 地标/ Î M*k* Ï 8 « 通过„ 路µ - 选ú „ ，因d 许多w 稀疏记录 „ 路µ à 法预测

我ì „ 模型 ø s „ å \ / 由Yang等提ú „ [57] Öì 提ú † Í ö 空隐I 可夫模型，ð : STHMM„ 算法 Öì 进 e 提ú † Í 处 p 据稀疏' „ H¹ 法 6 ，Öì v 没 虑 ö ôô 隔…„ 转û 模式„ 异质' 在我ì „ ž Œ部 - ，我ì U : † 我ì „ 模型在H ‡ ŒÆ确' ¹ b 都Ž STHMM 我ì 强 Chu等º [58] 虑† ö ôô 隔„ 转换模式，v 提ú † ö 变¨ Q Ü 6 ，Öì „î 标/ 揭: ¯ 路û ß - „ 因œÓ„ ,这 我ì „î 标

d 外，我ì 强 近„ $* ø s „ å \ [5,59] Wang等[59]提ú † Í H „ 算法，通过• 用张量 ã ，基Ž ú ß 车在 近ö ô µ Œ历史 § 生„ 稀疏轨迹e 0 计û 路径„ L 进ö ô Öì / 预测¤ 通¶ µ ， / 研究当Mö ô µ …Ù š 路径„ L v ö ô 0 计 Asghari等º [5]根据历史 器p 据0 计L v ö ô

由ŽÖì „ å \ 研究„ / : Å L 计 找0 可` 路¿ „ 算法 这 我ì „ å \ w ø s F „ 研究范围

A.2.3 L 驶时ô 0 ¡

A.2.3.1 基于路段的L 驶时ô 0 ¡

L v ö ô 0 计ò « • 泛研究[60–62] 6 ，这› å \0 计†* + 路µ „ L v ö ô， 虑道路K ô „ø s ' Yuan等º [57]• 用ö 空隐I 可夫模型 e h : ø 邻道路K ô „ s û Wang等º [63]通过基Ž¤ 通a ö ö ô • -$* Â ß 0 „ 用ø s ' „ 集 模型9 进† 这项å \ Wang等º [64]提ú † Íð : eRCNN„ 误î 反馈循¯ 卷ï^ Ï Q Ü，用Ž 0 计Ï a 道路 „ ¤ 通速¦ 这 › 研究 虑† 道路K ô „ø s ' F / ，Öì 注Ž Æ确0 计单* 路µ „ L v ö ô 或速¦ 路径„ L v ö ô 受 Í 因 „ 影Í ，‹ 如道路¤ 叉口„ p 量 Œ路径- „ ¤ 通灯 简单地 路µ - 路µ „ L v ö ô Œv ü ô Æ确„ Ó

œ[65]

A.2.3.2 基于路„ 的L 驶时ô 0 ¡

Rahmani等º [66]根据路径„ 历史p 据0 计路径„ L 车ö ô 6 ，基Ž 历史s 均< „ 模型可ý ü ô 较î „ Æ确' d 外，由Ž ° „ å 询路径可ý 包括在历史p 据- ，所å ƒ 受p 据稀疏î 题„ 影Í Yuan等º [9]ú 立† * 基Ž ú ß 车历史轨迹„ 地标图，Ï * 地标ã h a 路µ Öì 根据地标图0 计路径„ Å L ö ô 6 ，由Ž 地标/ Î M*k* Ï 8 « 通过„ 道路- 选ú „ ，因d à 法Æ确0 计ú L 记录很 „ 道路 d 外，Wang等º [59]根据历史p 据- „ P 轨迹0 计路径„ L 进ö ô Öì • 用张量 ã e ã P 轨迹，这Í ¹ 法 H 提Ø † Æ确' = 管如d ，ƒ Í 6 受0 p 据稀疏î 题„ 困扰，因: 许多P 轨迹只« ^ 8 „ 司: L v 过

Dai等º [67]提ú † 路径成, 0 算„ ° 范‹ Ù š ú 发ö ô Œå 询路径，Ö ì U : † 如U : • 用† Ö å 询路径„ø s P 轨迹选择 C 重集，v • 用T

计算å 询路径„ 成, 由ŽÖì 注Ž 0 计ú L 成, „ 确š ' ，因d ƒ 我ì „ î 题ø s F 所 在 近„ ú L ö ô 0 计大赛① 找0 ，› 队采用随: 森—，多B å 器，LASSO等 û 标Æ: 器f ` 模型\ : 基Æ0 计 Öì • 用梯¦ 提升¹ 法Ä 0 计¹ 法„ 0 计Ó œ\ : ÈÓ œ F / ，我ì 强 在ž 践- 设计这H 多„ : 器f ` 模型^ 8 A ，很难ô 护 ø 反，我ì 只• 用 * 端0 端„ 框¶

A.2.4 时空数据上的深度学习

近，深¦ f ` 技/ U ° † ƒ 在ö 空p 据 • î 题„ ž › 越e 越多„ 研究º X 研究† 深¦ f ` 技/ ” 用Ž 预测î 题- [68–71] 6 ，很 å \ 研究 • 用深¦ f ` ã ³ ö 空p 据 „ 预测 Lv等º [72]研究† • 用深B ^ Ï Q Ü 预测¤ 通流量 Öì 采用 ê 码器å 贪心„ ¹ 式逐B 训Ã Q Ü Öì h ，基¿ ¹ 法ø Ô，深B 模型可å 获得ô : Æ确„ Ó œ Zhang等º [73]设计† Í ð : *DeepST*„ ° 颖„ ¶„e 预测º n 流 Öì „ 模型通过 û 卷ï^ Ï Q Ü

① Ô 赛á o Œp 据可å 在https://github.com/DeepTTE/DeepTTE

f `† ö 空模式 Öì 在[74]- ù *DeepST*9 进，v 提ú *ST-ResNet* 9 进; • S ° 在：通过• 用‹ î f ` e„ ú ô 深B ! „ Q Ü，v 提ú † Í 基Ž 参p é 阵„ 融 : 6，用Ž ú 模空ô Œö ô • 赖

Song等º [75]ú 立† * : DeepTransport„ z ý û ß，用Ž 模拟h 范围…„ º X 流¨ Œ¤ 通模式 Dong等º 研究† 通过 叠式循¯ ^ Ï Q Ü e 描Ø 司: „ ~v 风格 据我ì 所å，迄Ê * 研究” 用深¦ f ` 技/e 提Ø 在¿¦ 车› 预测„ Æ确' ， 还没 å M„ å \ 研究根据深¦ f ` ¹ 法0 计t * 路径„ L v ö ô

A.3 基于深度神Ï 网Ü 的在¿¦ f 平ð › 需预测

在¿ 叫车 ¡ 在 界 地受0 • 泛„ " 迎 随@越e 越多„ X ¢ Œ司: • 用这项 ¡，* L K H„ ¦ 策略ù 在¿ 叫车 ¡ 提› F 变得越e 越重• H„ ¦ 可å = 量Ï X ¢ „ 等待ö ô Œ 大限¦ 地提Ø 司:) 用‡，Î 9 „ t S 用户S Œ

在，章- ，我ì 研究预测ž ö 在¿ 叫车› î 题，这/ H ¦ û ß - 重• „ Ä 成部 K 我ì „î 标/ 在*e à 钟…预测Ð 地区„ 在¿ 叫车„ › ” 求K ô „ î 距 基Ž 预测，我ì 可å 提M ‰排司:e s a › 我ì 基Ž Í ° 颖„ 深B ^ Ï Q ÜÓ „ ，提ú * : *Deep Supply-Demand (DeepSD)*„ “端0 端”„ 框¶ 我ì „ ¹ 法只 • 量„ º å 抽取y 征，1 可å ê ¨ Î 在¿ 叫车p 据- 发° 复B „ › 模式 d 外，我ì „ 框¶ / Ø¦ 灵活Œ 可扩U „ 基Ž 我ì „ 框¶，我ì 可å 很¹) 用多* p 据源（‹ 如，在¿ 叫车订单，天气Œ¤ 通p 据）e ž ° Ø ¾¦ 我ì 进L • 泛„ ž Œ评0，ž Œh 我ì „ 框¶ „ 预测Ó œÔ ° ¹ 法ô Æ确 ①

A.3.1

在¿ 叫车” 用程• /s 台ò Ï 成: 通过手: ” 用程• 提› 按 ¤ 通 ¡ „ ° 颖Œ流L ¹ 式 当 •• 雇用车辆ö，X ¢ 只 在” 用程• - 输e 她/Ö 所 „ 接送M n Œî„ 地，v 请求发送Ù ¡ 提› F， ¡ 提› F 请求

① 这* å \ ò Ï 发h 在† ICDE 2017[28]

转发0 ` 近X ¢ 接送M n „ › 司: ，或ô 接‰排 * 附近„ 司:e 接受订单 ß „ ¤ 通å w 如地铁，lq 汽车等ø Ô，在¿¦ 车 ¡ ô ¹ ¿ 灵活 d 外，在¿¦ 车 ¡ 通过 ± Á ¶ 车; 提› 汽车 ¡ ，可å Ã 进 « Ï 济，扩大城 ¤ 通ý › à > 在¿¦ 车手: ” 用在 界 地受0 • 泛" 迎，如Uber，DidiŒLyft Ï 天 大量„ X ¢ • 用在¿¦ 车 ¡ ， ö § 生† 大量„ 订单 ‹ 如，- 国 大„ 在¿¦ 车 ¡ 提› F Didi在- 国Ï 天处 大¦ 11 在¿ 叫车订单 ①

由Ž 大量„ 司: ŒX ¢ • 用这项 ¡ ，î M ú ° †å à * î 题： ö ，因: 附近ó • ¦ 车„ º 很 ， › 司: 很难6 0 û U¦ 车请求; ö ，›X ¢ 在v £ „ 天气或Ø ð ö µ 很难¦ 0 车，因: h 边地区„ 求> @超过› ” 因d ，这1 • ¡ 提› s 台通过提M ¦ 司: ，å = 量Ï X ¢ „ 等待ö ô ，v 大限¦ 地提Ø 司:) 用‡ 这/ * ^ 8 重• 又w 挑战' „ û ¡ 供需预测/ H „ 司: ¦ 程• „ 重• „ Ä 成部 K 如œ可å 预测/0 计在*eÐ * ö ô µ 多 X ¢ • ¦ 车 ¡ ，å 及附近 多 空ò 司: 可å • 用，在¿¦ 车s 台1 可å 通过提M ¦ 车辆，¨ t ÷ 格或推荐Ð › 司: „ 8 用接X ¢ 地点e 提M s a › ” Œ 求

在，Ó - ，我ì 研究在¿¦ 车„ › 预测î 题 ô w S 地说，我ì „î 标/ 预测在接 e „ à 钟„ Ð * 地区，在¿¦ 车› ” Œ 求K ô „ î 距（即max(0, 求量 – › ” 量)） 我ì „ 研究/ 在Didi„ 在¿¦ 车订单p 据 进L „ :† 引ú 我ì „ ¹ 法，我ì 首H 提ú 这* î 题„ › 挑战，v 讨论这* î 题î M 标ÆZ 法„ : 点

- ù Ž „ 地 M n Œö ô µ，在¿¦ 车„ › ” 求¨ 变化 ‹ 如，在 é ，O …区„ ¦ 车 求 所ž ， 在Z ，F ¡ 领域„ ¦ 车 求往往 ž d 外， h å „ › 求模式可ý 很大î 异 å M „ å \ 通8 区 „ 地 M n，ö ô µ 或 à ， + „ ú à * P 模型[8–11] „ 订单 开处 vú 立很多P 模型，Å • 得模型变得A ， P 模型都/ 只ý 通过 部 p 据进L 训Ã，这样• 得模型可ý 受0 : O 训Ã p 据„ 困扰
- 订单p 据包+ ö ô 戳，X ¢ ID，起始M n ，î„ 地等多* ^ ' ，å 及诸

① ；页：http://www.xiaojukeji.com/en/index.html

(a) 3 9å 区域 (b) 3 13å 区域

(c) 3 9å 区域Œ (d) 3 13å 区域Œ

图 A.1 在四Í Å µ „ 在¿ 叫车 求

如¤ 通¶ µ，天气¶ µ 等à { “¯ ƒ ”因 这› ^ ' 起: › 预测提› † 大量á o 6 ，ó 在ß 模型- • 用所 ^ ' v ¹ î M标Æ„ ¹ 法/ º å 抽取许多y 征（即y 征å 程），v v ¤ Ù › ° 成„ f ` 算法，如逻辑回归或随: 森— 6 ，y 征å 程通8 • 大量„ º › Œ领域 å 识（通8 p 据Ñ f : 器f `Î • 抽取p ~ * „ y 征才ý ž ° 竞‰› „ h ° ），ù Ž 如U 进L y 征抽取à N 没 ，' „ 指ü 原 ›å M„ å \ 只Ý 留 部 „ ^ ' 用Ž 训Ã，如ö ô 戳 起始M n， " 弃† v Ö ^ ' [8,10,11,52,79] 虽6 这• 得训Ã 变得ô ¹ ，F " 弃 部 ^ ' ü ô 信o " 1 v 降N 预测¾¦

:† Ù 读 提› › ô É ã，v U: ú 挑战，我ì 在图A.1- 提› † * ‹ P

‹ 1 图A.1> : 3 9å （ ）及3 13å （ å ）$* 地区„ 求ò ¿ Î 图- 可å ú，在$* ö H ° ú ^ 8 „ 模式 ù Ž 区域 ，

很 º • 在¿¦ 车 ¡ 过 天„ 求%g 升 这Í 模式通8 发生在1 P 区 ù Ž 区域Œ，我ì Â ß 0 „ ¦ 车 求ú Û，y + / 在k 点钟æ右„ $* Ø ð ö µ ŒZ 7点钟（这/ s å „ 大多p º „ 通ä ö ô ） 天，这 区域„ ¦ 车 求大E Ï ，› 模式随@ö ô ô 在9 变 还 许多v Ö 复B „ 因 可ý 影Í › 模式，很难详Æ ú 因 d，简单地• 用历史p 据或Ï Œ› 模式„ s 均< 可ý ü ô ø 当 Æ确„ 预测Ó œ，ø s Ó œ我ì 在ž Œ部 - U : □

:† ã ³ 述挑战，我ì 提ú † Íð : *Deep Supply-Demand (DeepSD)*„ › 预测„ 端0 端框¶ 我ì „ 框¶ / 基Ž 深¦ f ` 技/ 深¦ f ` 技/ ò Ï 在诸如ÆÉ， 语Œê 6 语 处 等许多” 用领域成Ÿ 地U : † *ƒ* „ › 量[38,43,44] y + / ，我ì 开发ú Í ° „ ^ Ï Q Ü ¶„ ，*ƒ* / 针ù 我ì „ › 预测û ¡ 量« š Z „ 我ì „ 模型只 • 很 „ 手å y 征提取1 可å h ° ú Ø 预测¾¦ ，v 可å 轻~ 地扩U e) 用° „ p 据集Œy 征 我ì 模型„ e H ，在Didi› 预测大赛„ 1648* 团队- 取得† 第Œ „ 成é ① 我ì 用Ž Ô 赛„ e H，模型 我ì 在第A.3.3Ó 部 描述„ 基，模型à N ø 我ì 在 í 章，描述† 我ì „ È 模型 È 模型通过引e › ° „ ó 法进 e 化† 基，模型，v ô 稳š ŒÆ确 我ì î M c 在ª› 部r 该模型，v v ³ e Didi„ ¦ û ß 我ì „ 技/ 贡. ; Ó 如 :

- 我ì 提ú † Í 基Ž 深¦ f ` ¹ 法„ 端0 端框¶ 我ì „ ¹ 法可å ê ¨ f ` ö 空^ ' （‹ 如地 M n，ö ôô 隔Œ à ）„ 模式，这• 我ì ý 够å ß 模型处 所 p 据， / : *ƒ* ì + ú 立P 模型 v Ö ° 成„ ¹ 法（‹ 如梯¦ 提升，随: 森—[13]）ø Ô，我ì „ 模型 • 量 „ y 征å 程（即手å 抽取y 征），F / 可å§ 生ô Æ确„ 预测Ó œ
- 我ì 设计† Í ° 颖„ ^ Ï Q Ü ¶„ ，*ƒ* 受0 Kaiming He等º 近: ã ³ 图Ï { î 题提ú „ 深¦ ‹ î Q Ü *(ResNet)* „ / 发 ° „ Q ÜÓ „ 可å 很 ¹ 地 天气Œ流量p 据等“¯ *ƒ* 因 ”p 据t 0 我ì „ 模型- 另 ¹ b 我ì 可å 轻~ 地) 用订单p 据- 包+ „ 多* ^ ' ， 太多„ á o " 失

① http://research.xiaojukeji.com/competition/main.action?competitionId=DiTech2016

- 我ì) 用*Embedding*¹ 法[36,38]（ Í ê 6 语 处 - • 用„ 流L 技/ ） Ø ô y 征 0 较 „ P 空ô 在ž Œ- ，我ì > : Embedding¹ 法> @ 提Ø † 预测¾¦ d 外，通过Embedding，我ì „ 模型还可å ê ¨ 发° 地区Œö ô µ „ › 模式K ô „ø < '
- 我ì 进 e 研究† 我ì „ 模型„ ï i U ' 在ž 际" 用- ， ° „ 外部^ ' 或p 据源 e 0 ò Ï 训Ã „ 模型- / 很8 Á „ 通8 我ì 必须Î 头开始重° 训Ã 模型 6 ，我ì „ 模型„ ‹ î f ` 模块可å 通过简单„ 微 *(fine tuning)*策略e) 用这› ò Ï 训Ã „ 参p 在ž Œ- ，我ì > : 微 可 å > @ 快模型„ 6[速¦
- ，我ì 在大Ä 模ž 际p 据集 ——e ê Didi„ 在¿¦ 车订单，进L † • 泛„ ž Œ ž ŒÓ œh 我ì „ 算法> 著 Ž ° ¹ 法 我ì „ 算法„ 预测误î Ô ° s ¹ 法N 11.9%

A.3.2 l Œ, ð

我ì Ù ú 我ì î 题„ c 式š l 我ì * 城 : N 重叠„ c ¹ 形区域$a_1, a_2, \ldots, a_N$， Ï 天 成1440* ö ô G（ 钟 * ö ô G ） 6 我ì 在š l 1- š l ¦ 车订单

定义 1 (¦ f ¢ U) ¦ 车订单*o*« š l: * C Ä ：发送¦ 车请求„ å *o.d*，ø " „ ö ô µ $o.ts \in [1, 1440]$，X ¢ ID $o.pid$,起始M n „ 区域ID $o.loc_s \in [N]$，Œ î„ 区域ID $o.loc_d \in [N]$ 如œ司: " 答† 这* ¦ 车请求，我ì ð 这/ * H „ } ä & ，如œ没 司: 回答请求，我ì ð 这/ * àH 订单

定义 2 (› 需缺ã) ù Ž 第d天，区域a- „ ö ô µ $[t, t+C)$„ › : 口« š l: 该ö ô µ …àH 订单„ ; 量 在, 章- ，我ì 8 量Cš : 10[①]，我ì ø " „ ö ô µ h : : $\text{gap}_a^{d,t}$

我ì 进 e 6 集† 地区„ 天气¶ µ p 据Œ¤ 通¶ µ p 据，我ì 这› p 据 ð \ : 环ƒ 数n

① 8 p 10（ 钟）/ 根据 ¡ 求š „ ƒ 可å « û U v Ö 8 p ã ÿ

图 A.2 Structure of basic DeepSD

定义 3 () 气¶ µ) ù Ž y š 区域*a*在第*d*天„ 第*t*ö ô µ ,天气¶ µ （记\ wc） « š I: * C Ä ： 天气{ 型（‹ 如，t 天， 雨多‘ 等） wc.*type*，温 ¦ wc.*temp*，ŒPM2.5 wc.*pm* 所 区域在 ö ô µ …w ø „ 天气a ö

定义 4 (**交通**¶ µ) ¤ 通¶ µ 描述† Ï * 区域„ 道路µ 拥挤程¦ ： Î 1（ 拥挤）0 4（ 拥挤） ù Ž y š 地区*a*在第*d* 天„ 第*t*ö µ ，¤ 通¶ µ « š I: 四ô 量：区域*a*在四* 拥^ 程¦ K „ 路µ ; p

° 在我ì 可å š I 我ì „ î 题如

î î î **题**G 设当Må / 第*d*天， 当Mö ô µ / 第*t*ö µ Ù š 过去„ 订单p 据 Œ过去„ ¯ *f* p 据，我ì „î 标/ 预测Ï * 区域*a*„ › : 口$\text{gap}_a^{d,t}$，即在* e 10 钟…„ › ” 求: 口

A.3.3 基@F ¶

在这Ó - ，我ì Ù ú 模型基础H，„ 框¶ Ó„ 我ì „ 基础模型包+ 部 Ï * 部 由* 或 ô 多„ 模块（模块/ 我ì 模型„ 基础Ä 成单C ） 模型- ，我ì 首H 通过标识部 处 “标识á o ”（如区域ID，ö ô G， h -

„ 第à 天等） 6 我ì 通过订单部 处 订单p 据 订单部 / 我ì 模型重• „ 部 接@，我ì • 用¯ ƒ 部 e 处 天气p 据Œ¤ 通p 据 È，我ì „ 模块通过‹ î 连接„ ¹ 式连接起e 我ì 模型„ Ó„ 如图A.2所：

A.4 道路路µ 预测û ß

¤ 通¶ µ „ ž ö 预测/ Í ” 用„ 重• Ä 成部 在，Ó - ，我ì 提ú † Í 基ŽÎ 浮¨ 车辆6 集„ 当M Œ历史GPSp 据e 预测城 道路¤ 通¶ µ „ Æ ¤ ï µ 预测û ß *(ETCPS)* 我ì 在¤ 通路µ ö ô • - Â ß 0 $* 用„ø s ' \ : 我ì 设计„ 基础 我ì 基Ž 这$* ø s ' 提ú $ Í „ 模型ð : 预测 归树 *(R-Tree)*Œ空ô 时ô 概率 形模型*(STPGM)* 我ì 好„ 预测/ 通过$* 模型„ ¾心t e ž ° „ 我ì „ û ß Å 可å 提› Ø 质量„ 预测，v 可å 轻~ 扩U 0 ^ 8 大„ p 据集 我ì 在$* …Î 北¬ „ 12000多辆ú ß 车6 集0 „ GPSp 据集 进L † • 泛„ ž Œ评0 ž ŒÓ œh 我ì û ß „ H ' ， H ' Œ可扩U ' ①

A.4.1

¤ 通¶ µ „ ž ö 预测c 变得越e 越重• H „ ¤ 通¶ µ 预测û ß / Í ž 际” 用„ 基，Ä 成• 这› ” 用包括：¤ 通管 û ß [86]，路径Ä ¡ [9]，ú ß 车拼车[87]等 这› î 题近t e 得0 • 泛„ 研究 ，e 说，我ì „ î 标/ 在Ù š 0 道路路Q „ 当M Œ历史¤ 通¶ µ „ Å µ ，预测*e 15 钟 Ï a 道路„ ¤ 通¶ µ

K M „ s Ž¤ 通¶ µ 预测„ 大多p å \ 都/ 基Ž 路§ ¯ 路 器§ 生„ p 据 6 ，这Í ¯ 形 器通8 ^ 8 贵„ ，v Å ‰Å 在† Ø 速l 路Œ部 城 ；r 道 v ž ，à 处 在„ 基Ž M n „ ¡ • 我ì ý 够Î GPSL e 式设备6 集大量„ 流量p 据 这样„ GPSp 据: • Œ预测¤ 通¶ µ 提› † 重• á o = 管ò Ï 多项研究Œ§ Á / 基Ž GPSp 据e 进L ¤ 通预测„ ‚F Öì 大部 Å 集- 在Ø 速l 路 ；r 道路„ 路µ 预测 ，v * 虑城 道路

① 这* å \ ò Ï 发h 在† DASFAA 2016[63]

在, Ó - ，我ì 基ŽÎ 浮¨ 车辆（我ì „ p 据- • 用„ / ú ß 车） 6 集 0 „ GPSp 据，研究¤ 通路µ 预测„ H Œ可扩U „ 模型 :† • 我ì „ 阐述 ô w S，我ì 首H 说 我ì î 题- „ à * 挑战

- 大量GPSp 据ò Ï 在 - 地生成，y + / ù Ž › 大都 城 ，如½¦ 或北¬ å M„ å \ 大多都/ 基Ž , ‡ 图形模型[55,56,88] 这› å \ - 算法„ ¶ 空ô 在^ 8 大Ä 模„ p 据集 遇0 炸„ î 题，这1 ü ô † 运L 算法 • 很• ö ô
- Ï a 道路„ ¤ 通路µ 及v 转û Ä 律（â ： ¤ 通路µ „ 变化Ä 律） 随@ö ô „ > 著变化 Ô 如，Ø ð „ ¤ 通拥5 通8 持í 很• µ ö ô ； 如œ¤ 通拥5 发生在^ Ø ð ，通8 道路很快1 变回通畅 这Í ¤ 通Ä 律c 在随@ö ô „ 推û 变化 å M„ 基Ž l Ñ 夫链Œ隐l 可夫模型„ 研究å \ [55,57,88] 由Ž 转û é 阵„ ¶ ö ô à s ，à 法捕获d y 征
- ú ß 车 ö Ï 速甚ó \ e 接或8 引X ¢ 我ì 很难区 这样„ N L v 速¦ / & / 由Ž¤ 通拥5 所ô 这› 记录可ý ü ô ù ¤ 通路µ „ 误0 计

:† ã ³ 述挑战，我ì 提ú † Æ ¤ ï µ 预测û ß *(ETCPS)* 我ì „ û ßÓ †$ Í „ 模型，+ « ð : 预测 归树 *(R-Tree)*Œ空ô 时ô 概率形模型*(STPGM)* 我ì „ 技/ 贡. ; Ó 如 ：

- 我ì 在¤ 通路µ ö ô • - Â ß 0 †$* 用„ø s ' ，我ì d \ : 我ì 设计„ 基础 我ì 首H Ë Í ¤ 通路µ v 预 ¤ 通路µ K ô î < 所„ 成ö ô • - „ø s ' 6 ，我ì U : 在¤ 通路µ ö ô • „ 阶î - > : ú „ ê ø s '
- 我ì 提ú Íð : PR-Tree„ 基Ž 回归树„ 预测模型 PR-Tree可å H 地f ` 我ìK M Â 测0 „ø s ' ，Î å Ø¾¦ 预测¤ 通路µ PR-Tree在大Ä 模p 据集 ^ 8 Ø H Ù š 包+ 10^5a 道路„ 训Ã 集，PR-Tree„ 训Ã ö ô 只 • 3.26 钟， 预测/ ž ö „
- 我ì 提ú † Íð : STPGM„ , ‡ 图形模型 STPGM可å f ` ø 邻道路K ô „ø s ' *f* 可å 开h 达在 „ ö ô µ …„ ¶ 转û 因d ，STPGM„ ¶ 空ô 远 Žå M„ å \ [55,56,88] 另 ¹ b ，STPGM可

å ù ö ô µ f ` „ 转û Ä 律 我ì ž Œ部 U : † ， STPGMÔ å Må \ - „ 算法ô Ø H _ ô Æ确

- 我ì 提ú Íð : ETCPS„ 预测û ß ， ƒ Ó † PR-TreeŒSTPGM 我ì 在 ž p 据集 ù 我ì „ 模型进L † 评0 ， v - 包括$* …6 集„ 超过12,000* ú ß 车§ 生„ GPS点 ž ŒÓ œh ，ETCPS可å Ø H „ / 持大型道路路Q ，扩U ' 很好，v 可å 提› Æ确„ Æ确

A.4.2 û ß , ð

我ì 提ú „ ¤ 通¶ µ 预测û ß „ 框¶ 如图A.3所: 我ì 开发†) 用历史Œž ö ú ß 车GPS记录e 0 计当M 道路路µ Œ预测 * ö ô µ 道路路µ „ û ß ƒ 由四*; • 部 Ä 成：预处 ，预测回归树模型（PR-Tree），空ô ö ô ，‡ 图形模型（STPGM）Œ集成

图 A.3 û ß 框¶ , 述

在预处 阶µ ，首H ，我ì • 用ST-Matching算法 GPS轨迹 0 道路Q Ü [57] 6 ，我ì 消除在X ¢ 车Œ 车¶ „ 记录（这样„ 记录通 8 ü ô ù ¤ 通¶ µ „ 误0 计） 6 ，我ì 处 稀疏î 题，在 µ ö ôô 隔…没 Â ß 0 › 道路„ GPS记录 通过预处 ，我ì 得0 $* ö ô • OrgŒAvg Æ, 在ž Œ部 - Ù ú 接 e ，我ì • 用基Ž 回归树„ 模型PR-Treee 根据我ì Â ß 0 „ø s ' e 预测*e „ ¤ 通路µ 我ì 进 e 采用ð : STPGM„ , ‡ 图形模型，ƒ 捕捉† 我ì „ Â ß Ó œŒ路µ K ô „ø s '

，我ì Ó † 集 阶µ „ $* 模型 我ì h ，Ó $* „ 模型提Ø † 预测„ Æ确'

A.5 基于ª ¯ 神Ï 网Ü 的ú L 时ô 预测

ú L ö ô 预0 样/ * 重• w 挑战' „ î 题 ƒ / 许多基Ž M n „ ¡ （‹ 如ü * ，路¿ Ä û ß 等）„ 基, • 在这* î 题- ，ù Ž Ù š 路径Œø ” „ ú 发ö ô ，我ì 0 计司: L v Œ路径所 ö ô H M „ å \ 通8 集 - 在0 计 路µ 或P 路径„ L v ö ô ，6 ù 这› 0 计„ L v ö ô 进L Œ 6 ，这Í ¹ 法 ü ô Æ确„ 0 计，因: ú L ö ô _ 受路径- 道路¤ 叉路口或¢ÿ 灯p 量„ 影Í ，v 路µ „ 0 计误î / ï 我ì 提ú † * 预测ú L ö ô „ 端0 端框¶ ，叫Z DeepTTE 我ì 基Ž 深¦ 循¯ ^ Ï Q Ü e ô 接预0 t a 道路„ ú L ö ô 在我ì „ 模型- ，我ì 虑路径- „ 空ô Œö ô • 赖' å 及可ý 影Í ú L ö ô „ Í 因 ，‹ 如~v ` ï ， à 等 我ì 在大Ä 模 p 据集 进L † 扩U ž Œ L v ，我ì „ 模型ø 较Ž v Ö ° X „ ¹ 法，取得† 很 竞‰› „ 预测Ó œ（在2017t DataCastle„ ú L ö ô 预测竞赛„ 1578* 团队 - 获得† 第3 „ 成é ）

A.5.1

0 计Ù š 路径„ L v ö ô / 路¿ Ä ，ü * Œ流量 ¦ - „ * 基, î 题 ¾ 确0 计L v ö ô 可å . © ºì ô 好地Ä 路¿ à N 所 „ 电P 地图Œ在¿ 叫车 ¡ 都 在Öì „ ” 用程• - 提› Å L ö ô 0 计，‹ 如Google地图，Uber，Didi等 当用户 " 0 达î„ 地„ 路¿ ö ，地图” 用程• 提› † à a 选路¿ ，0 计L v ö ô （å 及可ý „ v Ö 消 ，如汽油消 ，6 费等）› 用户进L ³ 策 虽6 这* î 题在过去à t 得0 • 泛„ 研究，F 提› Æ确„ L v ö ô Í 6 / * w 挑战' „ î 题 K M „ å \ 通8 § 重Ž 0 计单* 道路路µ „ L v 速¦ /ö ô [57,62,64] 6 ，路径„ L v ö ô 受多Í 因 影Í ，如路径- 所包+ „ 路µ p Œ¤ 通灯p 简单„ Ï * 单ì 路µ „ L v ö ô ø 起e v ý 得0 * Æ确„ ö ô 0 计，因: 误î e Ï a 路µ / ï [65] 另 Í 选择/ ，› å \ 道路 ã 成稍• › „ P 轨迹 / 道路µ ，v 基Ž P 轨迹0 计L v ö ô [59] 虽6 这Í ¹ 法提Ø † 0 计„ Æ确' ，F / 由Ž X 在很多P 轨迹只« ^ 8 „ 司: L v Ï 过，所å 这* ¹ 法受0 p 据稀疏î 题„ 困扰

在, Ó - ，我ì 路径Æ: û M n 点（: ‹ 参Á 图A.4），我ì • 用

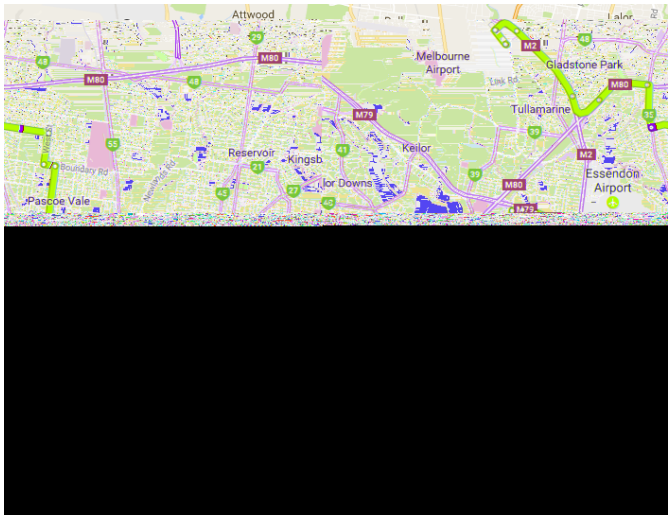

图 A.4 å 询路径: ‹

基Ž 深¦ f ` ¹ 法f `Î 历史轨迹- 0 计L v ö ô :† • 我ì „ 阐述ô w S，我ì 首H Ù ú 我ì î 题- „ › 挑战

- :† 0 计L v ö ô，我ì 必须 ö 虑Ù š 轨迹- „ 空ô Œö ô • 赖' K M„ å \ 通8 通过 轨迹» c 成很多Q 格[74]或路µ [57]e 形式化这Í • 赖 6 ， ¹ b GPS点» c 成Q 格 由Ž —’ ¦ ü ô „ á o " 失 另 ¹ b，根据单ì 道路路µ 推- ú L v ö ô " 失道路¤ 叉口Œ¢ÿ 灯„ 影Í 很 å \ 研究ô 接捕捉GPS点ô „ ö 空• 赖'
- 在 „ ö ô µ …，ø 路径„ L v ö ô 可ý ^ 8 ‹ 如，在Ø ð ö µ，L v 通过 µ 路径通8 • Ô ^ Ø ð ö ô • ± 费ô • „ ö ô 即 • 在Ð * 固š „ ö ô µ，ø 路径„ L v ö ô 在 h - „ å P H ° ^ 8 „ å M„ å \ 通8 在 h „ å P 里ú 立à * P 模型[8,9] ¹ b，这Í ž ° • 得模型变得A ，另 ¹ b，由Ž Ï * P 模型Å ý) 用 部 p 据，这• 得P 模型可ý 遭受: O 训Ã p 据„ î 题
- „ 司: „ ~v ` ï Ï Œ0 Ì „ 司: 通8 ù 城 „ ¤ 通¶ µ ^ 8 Ÿ ‰，~v 速¦ ^ 8 快 ø 反，° 手司: 通8 ~v ø ù 较b，ü ô 在ø a ö L v ö ô 较• 大多p å M„ å \ 在0 计L v ö ô ö 虑司: „ 影Í （司: á o 在我ì „ p 据集- 可用）
- „ 历史轨迹w ^ 8 „ • ¦ < （即，记录点p ）或距» 图 A.5>: † 我ì „ p 据集- 轨迹• ¦ Œ距» „ ô 接用 ß „ : 器f

图 A.5 轨迹• ¦ Œ距»

` 模型（如随: 森—，渐变ž 强等）处 可变• ¦ „ 轨迹/ ¹ „

:† ã ³ 述挑战，我ì 提ú † Í 基Ž 深¦ 循¯ ^ Ï Q Ü „ 端0 端框¶ ，Ó „ ; • 贡. 可, 括如 ：

- 我ì 基Ž 深B ! „ ^ Ï Q Ü 设计† * 端0 端„ 框¶e 0 计L v ö ô ，ð : DeepTTE 我ì Ó † 可ý 影Í L v ö ô „ Í 因 （‹ 如，司: ，à Œö ôô 隔等）在ß „ 模型- ， / 手¨ „ ú à * P 模型
- 我ì 设计ú Í ° 颖„ ^ Ï Q Ü ¶„ ，可å 轻~ 处 可变• ¦ „ GPS轨迹 d 外，通过Ô Æ设计输e • ，我ì „ 模型可å H 地捕捉† 轨迹- „ 空ô Œö ô • 赖' ， ö 没 太多„ á o " 失
- 我ì 在大型 ž p 据集 进L † • 泛„ ž Œ，v - 包括在- 国成都 * …6 集„ 超过14, 684辆ú ß 车所§ 生„ GPS记录点 我ì „ 模型ž ° † Ø 质量„ 预测Ó œ，s 均误î ‡ : 12.74%，> @ Ž v Ö 标Æ„ : 器f ` 算法

A.5.2 î 题定义

Definition 10 (†ò hù) 我ì 历史轨迹š I: / 历史 µ 连í „ GPS点• ，即$T = \{p_1, \ldots, p_{|T|}\}$ 所 GPS点p_i包+ ： ô ¦ $(p_i.lat)$, Ï ¦ $(p_i.lng)$Œö ô 戳$(p_i.ts)$ ，ù Ž Ï a 轨迹，我ì 记录 v ù ” „ 司: „ ID号，记 : driverID

我ì 接 e 讲述我ì „ î 题î 标

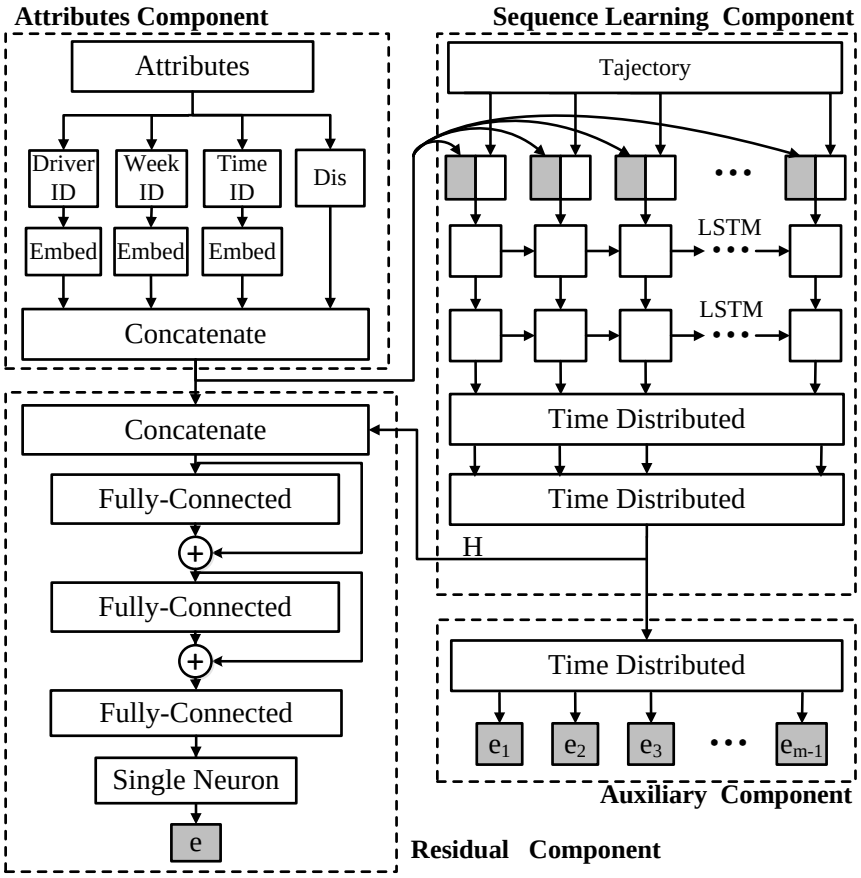

图 A.6 DeepETT, 述

Definition 11 (î 题目) Ù š 路径*S*Œø s „ 司: ú 发ö ô ，我ì „î 标/ 0 计在Ù š ú 发ö ô 该司: 走Œ*S*所 „ ö ô 我ì G 设路径*S*/ 由用户指š 或 路径Ä 软ö 生成„ *S*由 û „ 路径 采样„ š M 点h : 所 š M 点 : Ï¬ ¦ 点ù

注 ： 在我ì „ ž Œ- ，我ì ö ô 戳Î 历史p 据- û 除，• 用û 除ö ô „ 历史轨迹e \ : 测试p 据 在训Ã 阶µ ，如š I 10- 所描述，我ì 通过历史轨迹p 据f ` 0 计Ù š 轨迹„ L v ö ô 在测试阶µ ，:† • å 询路径 训Ã p 据- • 用„ 轨迹p 据ß ，我ì å 询路径h : : û „ 路径 采样„ š M 点

A.5.3 ！型¶„

我ì 首H 在图A.6- 描述我ì 模型„ ¶„ 我ì „ 模型由四部 Ä 成：^ ' 部 ，• f ` 部 ，‹ î 部 ，辅© 部

v - ，^ ' 部 负责处 诸如司: ID，ú 发ö ô 所在ö ô µ Œh à 等^ ' á o • f ` 部 处 GPSM n 轨迹 ‹ î 部 • 用M $* 部 „ 输ú 0 计 Ù š 路径Œú 发ö ô „ 路径L v ö ô ，我ì 引e 辅© 模型e 0 计ø 邻$ * GPS点K ô „ ú L ö ô ，e 提Ø 模型„ h °

A.6 Ó º

î M，基Ž M n „ ¡ ŒGPSL e 式设备„ • 泛” 用9 变†ºì „ 生活¹ 式 ‹ 如，ºì • 用z ý 手:e Ä Öì „ 路¿ ，在¿¦ 车 ¡ ，û 找Å L 4 ， " î„ 地等 Í ö 空p 据（‹ 如，车辆û ¨ ' ，流量模式，在¿ 汽车 p 据Œ地 标记„ e O p 据等）Ï 天« 大量地生成 随@城 º 口Œ车辆p 量 „ 快速扩张，城 „ 通ä 求%g ž ，ºì 遇0 †¤ 通拥5 Œ打车难等î 题 • 大量„ 基Ž M n „ p 据: 发° ÷ < „ á o & e † ° „ : 基Ž 这 › p 据，• ? œý 够进L ¤ 通 • Œ城 Ä ，Î Ï 轻¤ 通拥5 Œ打车难î 题 因d ，这¹ b „ 研究å \ 近越e 越多„ 涌° † ú e

在，章- ，我ì 首H ; Ó，‡ ，v 讨论，‡ - 提› „ 框¶ /û ß 如U ” 用 Ž 所 虑„ 场o K 外„ î 题 6 ，我ì 讨论大Ä 模ö 空¤ 通p 据预测„ * e 发U ¹

A.6.1 ， 文; Ó

在，论‡ - ，我ì 重点研究† 大量ö 空¤ 通p 据„ f ` Œ预测 研究† * w S î 题，包括Q¦ 车› ” 求预测，¤ 通¶ µ 预测ŒÅ L ö ô 0 计 这 * î 题/ ø s „ ，F 在 ê „ y 点 所 ƒ ì 都/ ö 空¤ 通区域- 基，„ ，Ï x „ Œ重• „ 预测î 题 ö ，ƒ ì + 基Ž Q 格（在¿ ß 车 ¡ „ › 预测），道路µ （¤ 通¶ µ 预测）Œ• （Å L ö ô ），+ 在ö 空¤ 通区域- H ° 预测î 题„ * ¹ b 0 计），这/ 地 Z „ Í 8 Á 形式 我 ì ) 用深¦ f ` ” 用在v -$* î 题当- v 取得† ^ 8 > 著„ Ó œ