Cognitive Computation (2019) 11:145–158
https://doi.org/10.1007/s12559-018-9603-8

Travel Time Functions Prediction for Time-Dependent Networks

Jiajia Li[1] · Xiufeng Xia[1] · Xiangyu Liu[1] · Liang Zhao[1] · Botao Wang[2]

Received: 13 November 2017 / Accepted: 10 October 2018 / Published online: 7 November 2018

Abstract
The studies on the TDN (time-dependent network), in which the travel time of the same road segment varies depending on the time of the day, have attracted much attention of researchers, but there is little work focusing on the travel time functions prediction problem. Though traditional methods for travel time or travel speed prediction problem can be used to generate the travel time functions, they have some limitations due to the need of less breakpoints, fine granularity, and long-term prediction. In this paper, we study the travel time functions prediction problem for TDN based on taxi trajectory data. In order to maintain a high degree of accuracy in fine-grained and long-predicted situations, we take into account not only the traffic incidents but also the data sparsity. Specifically, a traffic incident detection method is proposed based on k-means algorithm and a downstream-based strategy is proposed to estimate the speeds of segments considering the data sparsity. To make the breakpoints of function not so much, a prediction algorithm based on classification using ELM (extreme learning machine) is proposed, which predicts the speed classes taking both the weather and the adjacent segment conditions into account. In addition, a transformation method is presented to convert the discrete travel speeds into piecewise linear functions satisfying FIFO (First-In-First-Out) property. The experimental results show that ELM outperforms SVM (support vector machine) with regard to both the training time and prediction accuracy. Moreover, it also can be seen that both the weather conditions and the adjacent segment conditions have impact on the prediction accuracy.

Keywords Speed prediction · Travel time functions · Classification · Extreme learning machine

Introduction

Inexpensive GPS (Global Position System)-enabled devices become more and more popular, which gives the opportunity to collect large amounts of trajectory data from vehicles. The geo-spatial location data continuously recorded by GPS-enabled taxi contain very rich information for urban computing and developing intelligent transportation services, such as route planning, traffic flow analysis, and speed estimation, to name a few. Travel time prediction based on these trajectory data is the key to the development of advanced traveler information systems. With precise travel time prediction, a route planning system can suggest optimal routes to keep the users away from the traffic congestion, and users can decide an optimal departure time or estimate their expected arrival time based on predicted travel times.

Travel time prediction is also crucial in TDN (time-dependent network), in which the travel time of the same road segment can vary depending on the time of the day. TDN has attracted much attention of the temporal-spatial database researchers because of the increasing demand of shortest travel time by people in various applications, such as shortest path query [1, 2], k nearest neighbor queries

✉ Jiajia Li
lijiajia@sau.edu.cn

✉ Xiufeng Xia
xiaxiufeng@163.com

Xiangyu Liu
neulxy@gmail.com

Liang Zhao
lzhao@sau.edu.cn

Botao Wang
wangbotao@ise.neu.edu.cn

Extended author information available on the last page of the article.

[3–5], and its variants [6–10]. In TDN, the edge functions (travel time functions of edges) are usually presented by continuous piecewise linear functions. However, the experimental data of edge functions used by the existing works is either historical data or real-time data, without using the predicted data.

The historical data can not reflect the real-time traffic conditions. For example, the travel time of a road segment will increase if there is heavy rain or a traffic incident taking place. The query results based on these historical data may be slightly inaccurate. The real-time data and the short-term prediction (less than 1 h) is also not enough for deciding the optimal path if the destination point is too far away from the source point. Take the shortest path query as an example in Fig. 1, where the weights on the edges are the real-time data. As we can see, the quickest path from S to D is the path $< S, v_4, v_5, v_6, v_7, D >$. However, the arrival time to v_6 is 60 min later, at which time the travel time from v_6 to v_7 increases to 30 due to the peak hour or other reasons, which makes the path $< v_6, v_{10}, D >$ shorter than $< v_6, v_7, D >$. In this case, the quickest path computed by the real-time data may not be the quickest path in reality. Therefore, the route planning system should predict the travel time in order to give the optimal quickest path.

As described, edge functions prediction is very important in the TDN studies. But predicting the functions directly is hard to do in reality due to so many possibilities. It is more effective to predict the speeds first and then calculate the functions. However, to make the transformed functions applicable in the experiments, the fine-grained and long-term (no less than 1 h in urban road network) speed prediction is needed. In addition, considering that too many edge function breakpoints will bring great difficulties to the query problems under TDN [11], the predicted functions should have as few breakpoints as possible where a classification method is preferred.

In this paper, we are interested in the problem of predicting and generating the travel time functions for each road segment in TDN from the taxi trajectory data. There are many challenges to do this work, and the existing works can not solve the problem very well. (1) Some existing works only predict the travel time for the point of interests. But in TDN, the travel times of each road segment should be all predicted which poses a challenge to the efficiency of the prediction algorithm. (2) Some existing works for travel time prediction or speed estimate only consider the historical traffic data, but do not take the weather conditions into account. The latest literatures [12, 13] pointed out that the weather condition has a high effect on the traffic congestion. (3) Though there are some existing works that consider the weather conditions, they either only predict the degree of traffic congestion or predict only 30 min in advance. While in TDN, 30 min is not long enough because the cost of a path between two points in road network may exceed 30 min, which poses a challenge to the accuracy of the long-term prediction algorithm.

To address these challenges, we propose a new method to predict the fine-grained and long-term travel time functions based both on the taxi trajectory and weather data. The main contributions are as follows: (1) To make full use of the historical trajectory data, we analyze these data and use k-means algorithm to detect the traffic incidents ("Traffic Incident Detection"). (2) To handle the sparsity problem of the historical trajectory data, we propose a downstrem-based strategy to estimate the travel speed of segments which have not sufficient data ("Travel Speed Estimation Using Sparse Data"). (3) In order to make the breakpoints of function not so much, a prediction algorithm based on classification using ELM (extreme learning machine) is proposed, which predicts the speed classes taking both the weather and the adjacent segments conditions into account ("Travel Time Functions Prediction"). (4) To satisfy the need of the researches for TDN, a travel time functions estimation method is presented in "Travel Time Functions Estimation."

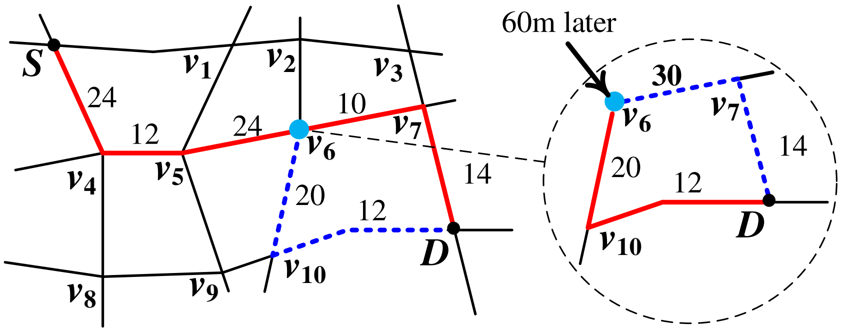

Fig. 1 Quickest path in time-dependent network

Related Work

Travel Time Functions Generation

Shortest path query [1, 2], *k* nearest neighbors query [3–5], and its variants [6–10] based on TDN have attracted a lot of research interests in the past few years. But in the experiments, they either used the historical data simply or generated the simulated data for the underlying network, which was unserviceable as described above.

Few works describe how to create a TDN or predict the travel time for the network. As far as we know, only the literatures [14] and [15] focused on the problem of computing the travel time functions for TDN. The literature [14] proposed a gravitational model method to compute road segment average speed from trajectory data and generated travel time functions from the computed average speeds. The literature [15] applied a map matching technique allied to a curve smoothing approach for generating piecewise linear functions in order to treat outliers and incomplete data.

However, they both computed the functions just using the historical trajectory data, and did not consider the functions prediction which is important for the intelligent transportation services.

Travel Time Prediction

To predict the travel time functions of the road network, the travel time or the travel speed can be predicted at first. There have been a large amount of algorithms to address travel time prediction problems. Vlahogianni et al. summarized existing short-term traffic prediction algorithms up to 2003 in literature [16], and summarized algorithms from 2004 to 2013 in literature [17]. The algorithms for travel time and speed prediction range from statistical model (SM) methods, Kalman filter (KF) theory, artificial neural networks (ANN), support vector regression (SVR), and hybrid approaches.

The advantages of SM [18, 19] are its good theoretical interpretability and clear calculation construction. The literature [20] modeled the traffic time series by phase space techniques. The literature [18] considered the effect of upstream and downstream and proposed a vector autoregressive model. The literature [19] considered the effect of the trend and seasonality in data on the generalized autoregressive conditional heteroskedasticity (GARCH) models and proposed two-component GARCH models that are able to model trend and seasonal components through decomposition.

KF theory [21, 22] captured regression problem in a state space form by minimizing variance for optimal solution. Chien et al. [23] modeled real-time and historic data for travel time prediction and applied KF for prediction, and explored the factors that can affect the prediction results. Yang et al. [24] proposed a recursive least square (RLS) approach for short-term traffic speed prediction by means of KF, based on the maximum likelihood method and Bayesian rule. To improve its online-learning ability, Wang et al. [22] proposed an extended KF method to predict highway travel time.

Comparing to SM and KF, ANN [25, 26] has been a widely used method in traffic prediction due to its strong generalization and learning ability as well as adaptability. Some algorithms utilized support vector machine (SVM) [27–29] to map data into a high-dimensional feature space via a nonlinear relationship and then performs linear regression within this space to predict travel time and speed. As every prediction algorithm has its own advantage and applicable conditions, some hybrid models combining different methods [30–32] are proposed and applied to improve prediction performance.

In addition, weather change has a high impact on the traffic conditions which is verified by Sigakova et al. [33] and Ding et al. [12]. Abdel-Aty et al. [34] and Qiao et al. [35] studied the problem of travel time prediction by considering the weather conditions.

However, the above algorithms just predicted the short-term travel time or speed and paid no attention to the travel time function generation problem. To generate more applicable underlying data for TDN, the algorithm should predict the travel time within at least 1 h for each road segment, which poses a challenge to the accuracy and efficiency of the algorithm.

Extreme Learning Machine

Recently, extreme learning machine (ELM) [36, 37] and its variants [38] have attracted increasing attention from more and more researchers. ELM has originally been developed based on single-hidden layer feedforward neural networks (SLFNs) and then extended to the "generalized" SLFNs, where the hidden layer need not be neuron alike [39, 40]. ELM randomly assigns the input weights and the hidden layer biases, and then analytically determines the output weights of SLFNs. ELM is less sensitive to user-specified parameters, and can be deployed faster and more conveniently than conventional learning algorithms for classification. There are many applications of ELM for regression and classification, such as adverse cardiac events prediction [41], surface material recognition [42], intrusion detection [43], huge hypotheses re-ranking [44], pattern classification [45], and optimization query for nearest neighbor queries [46, 47].

To predict the travel time for each road segment, high dimensions of features of the segment should be extracted and large scale of training data should be considered. ELM is a potential neural network to solve the problem due to its extremely fast learning speed, that is why we use ELM as our classifier in this work. Ban et al. [48] implemented the new neural networks based on ELM and designed a real-time traffic index in the data of a real-world city of Nanning in South China. The experimental results show that ELM provides good generalization performance at extremely fast learning speed and high accuracy compared with other state-of-art algorithms. However, this work did not take the weather conditions and the traffic incident data into account and cannot predict the travel time within 1 h in high accuracy.

Overview

Preliminaries

Definition 1 ***(GPS Trajectory)*** A GPS trajectory T is a point sequence $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$ linked by the time stamps of the GPS points. Each GPS point p_i is a triple $(p_i.lat, p_i.lng, p_i.t)$ which are its latitude, longitude, and time stamp respectively.

Definition 2 ***(Time-Dependent Network (TDN))*** A time-dependent network is a directed graph $G(V, E, C)$, where $V = \{v_1, \cdots, v_n\}$ is a set of vertices representing the intersections and terminal points of the road segments. $E = \{(v_i, v_j)|v_i, v_j \in V, i \neq j\}$ is a set of edges representing the road segments. $C = \{c_{(v_i,v_j)}(t)|(v_i, v_j) \in E\}$ is a travel time function which represents the weight for (v_i, v_j) depending on a time instant $t \in [0, T]$.

For each edge (v_i, v_j), a function $c_{(v_i,v_j)}(t)$ gives the cost of traversing (v_i, v_j) at the departure time $t \in [0, T]$. The formalization about the function cost is defined as follows:

Definition 3 ***(Travel Time Functions)*** The function $c_{(v_i,v_j)}(t)$ is a piecewise linear function where each edge (v_i, v_j) is associated with a street segment s_i, observing the timestamp TS_i, and the total time TT_i. The piecewise is the $\sum sce_i(TT_i, TS_i)$, where TT_i is the travel duration, and TS_i is the time of beginning of the edge (v_i, v_j), for all objects.

We assume that the travel times of the edges in the network follow the FIFO property, i.e., an object that starts traversing an edge first has to finish traversing this edge first as well. The general time-dependent shortest path problem in which the departure is immediate, i.e. the user departs exactly at the time t, and in which waiting is disallowed everywhere along the path through the network is NP-hard [1], but it has a polynomial time solution in FIFO networks. Since the travel times satisfy the FIFO property, waiting in an intermediary vertex in a path is not beneficial.

Framework

Figure 2 presents the overall framework of this paper. The framework consists of two major parts: the offline training and the online predicting.

In the offline part, there are two tasks. The first task is to prepare data for the ELM classifier due to that the original trajectories data may have some errors or not sufficient enough to the prediction as mentioned above. Another task is to generate the classification models for prediction.

Firstly, the collected taxi trajectories are projected into the underlying road network to get the information of the segments using a map-matching algorithm. The obtained original trajectory only contains the spatial and temporal information of the taxi and the road segments, and need to be processed in the second step. Specifically, since the location of the taxi may be not exactly located in the vertices (i.e., junctions) of the network when uploading its longitude and latitude to the server, the average travel speed should be recomputed to obtain the historical travel time for each segment by using some analysis and processing methods. Considering that the historical trajectory data may contain some abnormal data, we conduct a k-means algorithm to discover the traffic incident data which is useful for the prediction when an incident happens in the similar road segment. In addition, the trajectory data are too sparse to obtain the full information for each road segment. Hence, we propose a downstream-based strategy for the travel speed estimation utilizing sparse data. Thirdly, the speed data combined with the weather conditions data are used to feature selection, where the traffic conditions of the adjacent segments are considered. At last, the selected features are used to train the classifier ELM to obtain the models for each future time interval.

The task of the online part is to predict the travel time functions. Firstly, the real-time traffic as well as weather

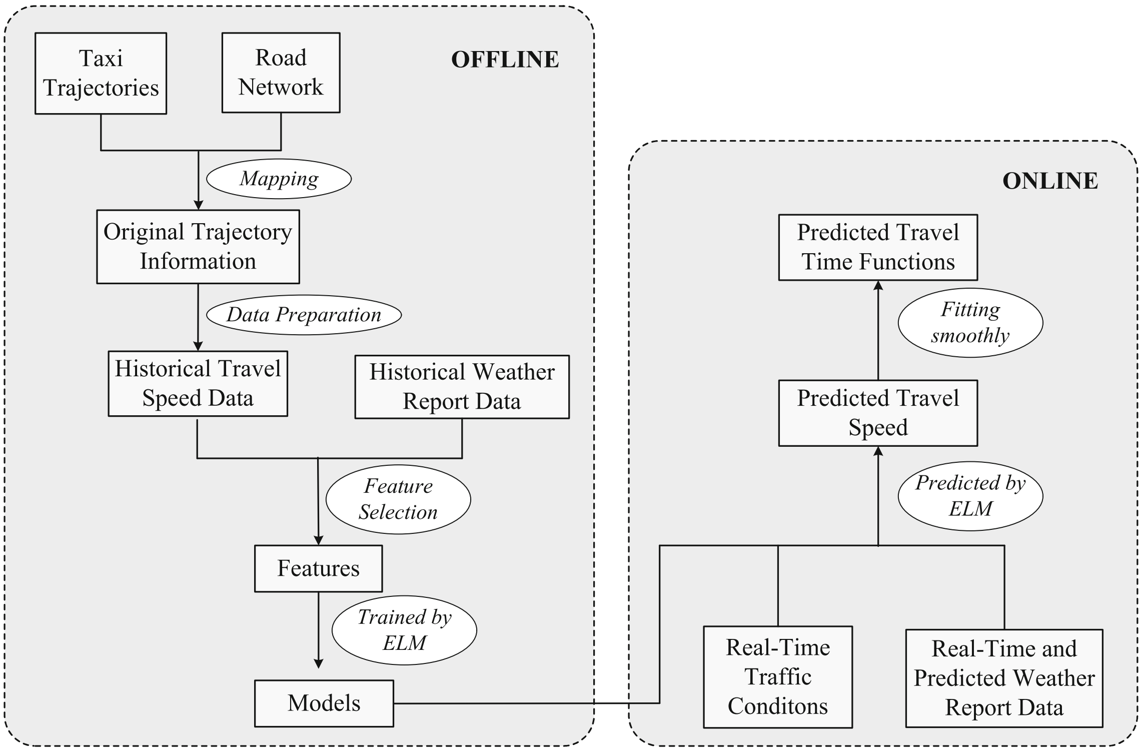
OFFLINE
Taxi Trajectories
Road Network
Mapping
Original Trajectory Information
Data Preparation
Historical Travel Speed Data
Historical Weather Report Data
Feature Selection
Features
Trained by ELM
Models
ONLINE
Predicted Travel Time Functions
Fitting smoothly
Predicted Travel Speed
Predicted by ELM
Real-Time Traffic Conditons
Real-Time and Predicted Weather Report Data

Fig. 2 Overall framework

conditions data are input into the trained models. Then, the predicted average speeds are used to create the piecewise constant travel time functions.

Historical Data Processing

We use the trajectory data downloaded from DATATANG,[1] which were collected by 12,000 taxis in Beijing during November 2012. The size of the zip file is 15.1 GB before decompression, and there are 48,156 files and each file contains 20,200 records. These data should be analyzed and handled firstly before used to feature selection.

Trajectory Mapping

Each record of the historical trajectory data contains time, longitude, latitude, speed, direction, and other useful information, and should be projected into the underlying road network firstly before used. In this paper, we use the interactive voting-based map matching (IVMM) algorithm proposed by Yuan et al. [49] which considers both the position context of GPS points and the road network topology. The IVMM algorithm consists of four phases: candidate preparation, score matrix building, interactive voting, and path finding. In the first phase, a range query is issued to select the candidate road segments and candidate points for each sampling point, then in the second phase, a candidate graph is constructed by performing a spatial and temporal analysis of the position context. After that, a static score matrix is built according to the candidate graph. The mutual influence is modeled utilizing a weighted score matrix which is constructed dynamically. Based on the weighted score matrix, all the candidate points vote in parallel for their best matching paths in the last phase. Then, a global optimal path is selected according to the voting result.

[1]http://www.datatang.com/data/44502

Historical Travel Speed Computing

In this subsection, we will describe how to compute and record the travel time for the segments using the historical trajectory information and the historical weather data. Since the points in the trajectory may be not exactly located in the vertices (i.e., junctions) of the network, to obtain the historical travel time for each segment, the average travel speed should be recomputed.

Take Fig. 3 as an example, there is one trajectory which consists of a point sequence p_1, p_2, p_3. Firstly, the travel time $t_{i,j}$ between each two adjacent points p_i and p_j belonging to the same trajectory is computed. In this figure, the travel time between p_1 and p_2 is $20121101061535 - 20121101061445 = 50$ s and it is 12 s between p_2 and p_3. Secondly, the distance $d_{i,j}$ of the shortest path of p_i and p_j is computed according to the network. The shortest path between p_1 and p_2 is $p_1 \rightarrow v_1 \rightarrow v_2 \rightarrow p_2$, and the distance $d_{1,2}$ is 300 m. While for p_2 and p_3, the shortest path is $p_2 \rightarrow v_3 \rightarrow p_3$, and $d_{2,3}$ is 120 m. Thirdly, the average speed of the edges $s_{i,j}$ involved in the shortest path is estimated by $d_{i,j}/t_{i,j}$, which is 6 m/s for p_1 and p_2 and 10 m/s for p_2 and p_3. That is to say, the average speeds of e_1, e_2, and e_3 are all 6 m/s, and 10 m/s for e_3 and e_4. As for e_3, the average speeds is $(6 + 10)/2 = 8$ m/s.

The travel time is time-varying and shows a regular change with daily and weekly patterns. For instance, daily patterns distinguish rush hour and late-night traffic and weekly patterns distinguish weekday and weekend traffic. Moreover, the travel time is also sensitive to weather conditions. Therefore, when we record the computed average speed for segments, we also record the time, the day, and the weather conditions. We split one day from 6:00 to 22:00 into 96 bins, corresponding to 10-min intervals. As for the weather, we obtain the historical weather data of Beijing in November 2012 from the Internet[2] and divided the weather conditions into two classes which are sunny (denoted by 0) and rainy/snowy (denoted by 1). For example, in Fig. 3, the starting time of p_1 to p_2 is 20121101061535, which belongs to the 38th interval of one day. That day was Thursday, and a sunny day. Therefore, for the edge e_3, we will record $\langle 6, 4, 38, 0\rangle$, which denotes that the travel speed of e_3 is 6 m/s during the 38th interval on Thursday on a sunny day. If there is a new record for e_3 obtained by another trajectory with the same day and the same interval, the travel speed will be updated by averaging these two records after verifying that it is not an outlier record (please refer to "Traffic Incident Detection" for details). For example, in Fig. 3, e_3 has another record according to the path p_2 to p_3, which is $\langle 10, 4, 38, 0\rangle$. It has the same day and the same interval with the last record, so the record will be updated with the average speed, which is $\langle 8, 4, 38, 0\rangle$.

Traffic Incident Detection

Since the traffic incident has a high impact on the travel time in urban network, it is necessary to detect the traffic incident from the historical trajectories. One purpose is to predict the travel time more accurately when a real incident happens, and another purpose is to improve the accuracy of the computed historical travel time. In this paper, we conduct the k-means algorithm to detect the outlier records.

After collecting all the records for segments according to the trajectories, we conduct the famous k-means (we set k equals to 1) algorithm for those records who have the same day and the same interval, as well as the same weather flag, to detect the outlier records (the abnormal speeds). These abnormal records should be further verified to distinguish the sudden incident from the sampling error. When an incident happens, the travel speed may have a sudden drop rather than a sudden rise. In addition, the travel speeds of the adjacent edges and the other trajectories containing the same segment may all slow within several minutes. Therefore, if the outlier records detected by k-means algorithm have a high travel speed, they will be regarded as the sampling errors and discarded from the records. If there are more than two outlier records with sudden deceleration at the close time and the adjacent edges under the same time interval have similar abnormal speeds, they will be regarded as the incident records. The incident records will be recorded by adding another flag to the tuple of the record of the edge, and they will be used to predict the travel time when there is an incident around this edge in real-time.

Algorithm 1 Traffic incident detection

```
Input: trajectories records $R_n$ for $n$ road segments;
Output: the incident records sets $icdS_n$ for $n$ road
        segments
1  for ($i$=1 to $n$) do
2      conduct $k$-means algorithm for the trajectories
       records in $R_n$ of the $i_{th}$ road segment with the
       same $\langle day, interval, weatherflag\rangle$ to obtain the
       center speed $cs$ and outlier records set $ORS$;
3      for ($j$=1 to $|ORS|$) do
4          if (the speed of $ORS_j > cs$) then
5              delete $ORS_j$ from $ORS$;
6  for ($i$=1 to $n$) do
7      if $|ORS_i| > 1$ then
8          $sum = 0$;
9          for (each adjacent edge $e_j$ of $i$ ) do
10             if $|ORS_{e_j}| > 0$ then
11                 $sum$ ++;
12         if ($sum > 2$ ) then
13             add each record of $ORS_i$ into the $icdS_i$;
```

[2]http://lishi.tianqi.com/beijing/201211.html

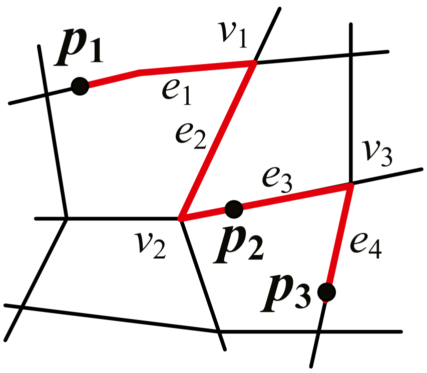

Fig. 3 An example of computing travel speeds

Algorithm 1 shows the details of the proposed traffic incident detection method. For each road segment, we conduct the k-means algorithm on the trajectory records which have the same day, the same interval, and the same weather condition (line 2). Then, the obtained outlier records (those outside the cluster) will be checked one by one to distinguish the traffic incident data from the sampling errors by comparing the speed with the center speed cs (lines 3–5). Note that the number of outlier record set is not fixed. Based on the observation that if there is an incident, the speeds of the adjacent edges usually will fall off. Therefore, if there are at least two outlier records in the adjacent edges, it is considered a traffic incident and the records will be put into the incident record sets for this road segment (lines 6–13).

Travel Speed Estimation Using Sparse Data

As mentioned above, we collect historical travel time under three dimensions for each road segment, which are different days (Monday to Sunday), different time intervals (1 to 96), and different weather conditions (sunny or rainy/snowy). According to these different conditions, each segment has at least $7 \times 96 \times 2 = 1344$ records. However, trajectory data are sparse as a driver can only travel a few road segments in a time

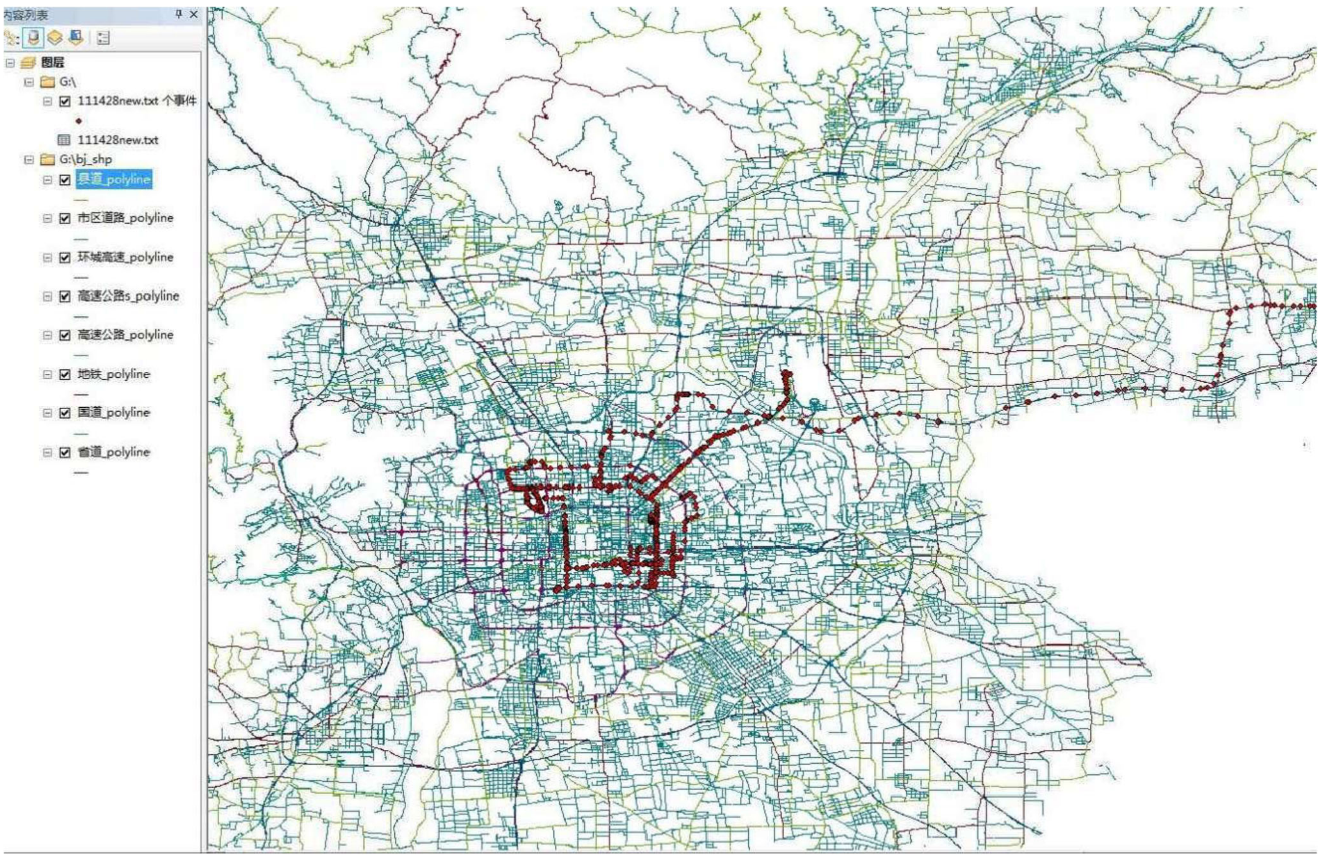

Fig. 4 One-day trajectory of one taxi in Beijing

interval as illustrated in Fig. 4, which shows 1-day trajectory of one taxi in Beijing. Even with millions of trip observations, there are a lot of road segments not covered by any trajectory because of the non-uniform spatial distribution of taxi locations in the city. In addition to the spatial sparseness, the trajectory data are temporally sparse too. Even if a road segment is covered by a few trajectories, these trajectories may not be sufficient to estimate the dynamic travel time that varies under different conditions (e.g., peak time and weekends). Hence, the estimation of historical travel time is not accurate due to the limited number of trajectories covering the road segment. To estimate travel speed more accurately, we propose a downstream-based strategy to handle the coarseness problem of trajectories.

Due to the topology of the road network, the movement of vehicles forms a traffic pattern. It leads to the spatial correlation of traffic features within a local area. In other words, the travel speeds of upstream traffic are correlated to those of downstream traffic. For example, Fig. 5 is a directed road network. The travel speed of $v_3 \rightarrow v_4$ is related to the speeds of $v_4 \rightarrow v_2$, $v_4 \rightarrow v_5$, and $v_4 \rightarrow v_7$, and we believe that they have the similar fluctuates. Therefore, we estimate the travel speeds for segments by considering both the speed of their own in other times and the speed pattern of their downstream segments.

More specifically, a k-means algorithm is conducted on each day (Monday to Sunday, sunny to rainy/snowy) of the segments to get two centroid values. The obtained two values are regarded as the base speeds of the off-peak hours (denoted by v_{bo}) and peak hours (denoted by v_{bp}), respectively. And the difference values between the actual speeds and v_{bo} or v_{bp} are regarded as the fluctuation margins (denoted by v_{fm}), which are used to estimate the speeds for the missing date of other segments. The road which has the least missing data compared to 1344 will be handled with high priority, while the roads which have no record will be handled in the end. For the segment e_i, which has no record on the interval m in n (Monday to Sunday) day on a w (sunny or rainy/snowy) day, the averages of the fluctuation margins under the same setting of its downstream segments are computed firstly. The result is just the fluctuation margins of e_i, denoted by $v_{fm}(i_{m,n,w})$. The speed on the interval m is estimated by $v_{fm}(i_{m,n,w})$ + $v_{bo}(i)$ or $v_{fm}(i_{m,n,w})$ + $v_{bp}(i)$. For the segment e_j, which has no record at all, the base speeds $v_{bo}(j)$ and $v_{bp}(j)$ are estimated firstly. They are represented by the average of v_{bo} or v_{bp} of its downstream segments. And then the speeds on each interval and each day on two weather conditions are all estimated by the same method as e_i.

Algorithm 2 Travel speed estimation for sparse data

```
   Input: historical trajectories records $R_n$ for $n$ road
          segments; value of $k$;
   Output: the full speed data under 3 dimensions for $n$
           road segments
1  for (i=1 to n) do
2      conduct a $k$-means ($k$=3) algorithm to get three
       centroid values $ctd_{1,2,3}$;
3      set the average speeds of the off-peak hours ($v_{bo}$)
       and peak hours ($v_{bp}$) based on $ctd_{1,2,3}$;
4      count the number of missing records
       $numMissing_i$ for road segment $i$;
5      sort $n$ road segments in ascending order by
       $numMissing$;
6  for (the sorted segment i=1 to n) do
7      for (the missing record j=1 to $numMissing_i$) do
8          for (each downstream segments $ds$) do
9              compute the fluctuation margins
               $v_{fm}(ds_{m,n,w})$ = actual speed of $ds$ - $v_{bo}$ or
               = actual speed of $ds$ - $v_{bp}$ based on the
               interval $m$;
10         the fluctuation margins of $i$th segment is
           estimated by $v_{fm}(i_{m,n,w})$ =
           AVERAGE($v_{fm}(ds_{m,n,w})$);
11         if ($i$th segments has no $v_{bo}$ or no $v_{bp}$) then
12             estimate $v_{bo}(i)$ = AVERAGE($v_{bo}(ds)$);
13             estimate $v_{bp}(i)$ = AVERAGE($v_{bp}(ds)$);
14         the estimated speed of $i$th segment
           $v_{estm}(i_{m,n,w}) = v_{fm}(i_{m,n,w}) + v_{bo}(i)$ or =
           $v_{fm}(i_{m,n,w}) + v_{bp}(i)$ based on $m$;
```

Algorithm 2 shows the details of the proposed travel speed estimation method for sparse data using the downstream segment strategy discussed above. Lines 1–5 is a preprocessing phase, the purpose of which is to compute the average speeds of the off-peak hours and peak hours for those segments that have enough data, and sort the road segments in ascending order by their number of missing record. Lines 6–14 estimate the speed for a segment according to its downstream segments based on that they may have the similar fluctuation margins. In line 9, the fluctuation margins of the downstream segments of ith segment on the mth interval, n day, and w weather are denoted as $v_{fm}(ds_{m,n,w})$. Then, estimate the fluctuation margins by the average fluctuation margins $v_{fm}(i_{m,n,w})$ of the downstream segments of the ith segment under the same setting as the jth record (line 10). Next, estimate the speeds of off-peak hours and peak hours for the ith segment by its downstream segments

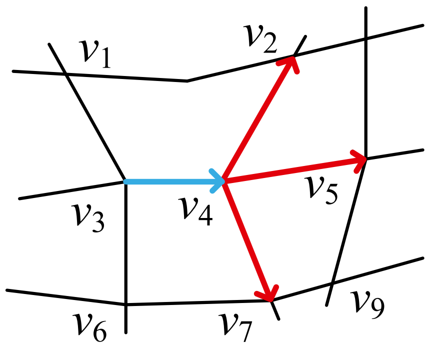

Fig. 5 Downstream segments for $v_3 \rightarrow v_4$

if the ith segment has no data (lines 11–14). If the interval m belongs to the off-peak hours, the estimated speed of the ith segment will be calculated by $v_{fm}(i_{m,n,w}) + v_{bo}(i)$. Otherwise, it will be calculated by $v_{fm}(i_{m,n,w}) + v_{bp}(i)$ (line 14).

Travel Time Functions Prediction

In this section, the details of the travel time functions prediction algorithm are described. Firstly, the features used to classify the speeds are introduced. Then, the ELM based travel speed classification algorithm is proposed. At last, the travel time functions estimation method is presented.

Feature Selection

Feature selection plays an important role in the process of classification and affects the predicted results to some extent. In this subsection, we discuss which features should be selected for travel time prediction.

As discussed in "Historical Data Processing," the travel time is time-varying and shows a regular change with daily and weekly patterns. Moreover, the travel time is highly sensitive to weather conditions and a sudden traffic incident. And the travel time of one edge is also influenced by the conditions of its downstream segments. Therefore, we select the following features for each segment in the urban network.

- Current time interval: 1, 2, $\cdots$, 96, representing the time of 06:00–06:10, 06:10–06:20, $\cdots$, 21:50–22:00.
- Current date: 1, 2, $\cdots$, 7, representing the date of Monday, Tuesday,$\cdots$, Sunday.
- Weather conditions: 0, 1, representing the sunny day or rainy/snowy day respectively.
- Traffic incident flag: 0, 1, representing there is no or there is a traffic incident respectively.
- Base speeds of the off-peak hours v_{bo}: 1, 2, $\cdots$, 9.
- Base speeds of the peak hours v_{bp}: 1, 2, $\cdots$, 9.
- Actual speeds of last 6 time intervals respectively: 1, 2, $\cdots$, 9.
- v_{bo}s of downstream segments respectively: 1, 2, $\cdots$, 9.
- v_{bp}s of downstream segments respectively: 1, 2, $\cdots$, 9.
- Actual speeds of the last six time intervals of downstream segments respectively: 1, 2, $\cdots$, 9.

It is worth mentioning that the travel speeds are divided into nine classes from 1 to 9, which represent the speed range from 0 to 2 m/s, from 2 to 4 m/s, $\cdots$, 16 to 18 m/s. The maximum speed is set to 18 m/s, that is because the speed limit in urban network is usually 60 km/s, which is about 16.7 m/s.

Travel Speed Prediction Using ELM

As mentioned above, many selected features must be collected online, and the processing of travel speed prediction should be finished in real-time. Hence, the chosen classifier should have fast prediction speed. Since ELM requires less training time and provides better performance than traditional learning machines, it is adopted to classify the travel speeds for the next six time intervals within 1 h for each road segment in this paper.

As illustrated in Fig. 2, after the phase of historical data processing, the obtained historical travel speed data and the historical weather report data will be used to feature extraction. Extracted features will be input to ELM classifier

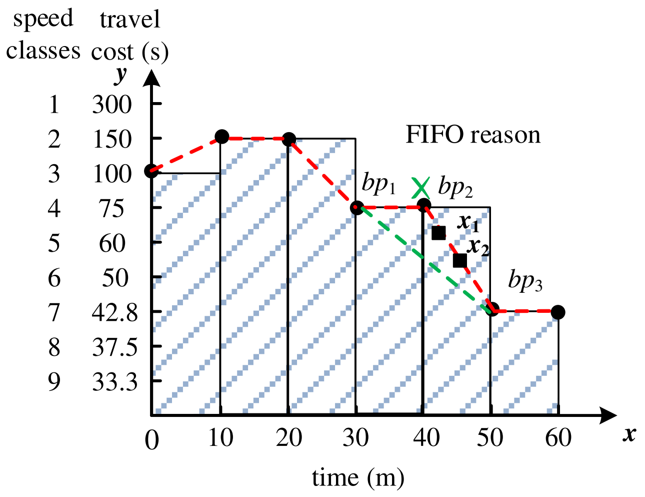

Fig. 6 From histogram to linear-piecewise profile

and trained. For each segment, six ELM classifiers will be trained to predict the categories of travel speeds of the next six time intervals. After getting the real-time and predicted weather report data as well as the real-time traffic conditions (whether there is an incident), the categories of travel speeds of each segment within six time intervals will be predicted by the trained six ELM classifiers respectively.

Algorithm 3 Travel speed prediction

```
Input: preprocessed trajectories records $R_n$ for $n$ road
       segments; historical weather data for $n$ road
       segments;
Output: the predicted speed classes for further 6 time
        intervals
1 //off-line part
2 extract the features based on the preprocessed
  trajectories data and historical weather data for $n$ road
  segments;
3 for ($i$=1 to $n$) do
4     train 6 ELM classifiers based on the extracted
      features;
5 //on-line part
6 for ($i$=1 to $n$) do
7     collect the real-time traffic conditions data,
      real-time and predicted weather report data;
8     extract the corresponding features;
9     predict the speeds of the next 6 intervals using the
      generated 6 ELM classifiers;
```

Algorithm 3 shows the details of the processing for prediction.

Travel Time Functions Estimation

A TDN consists of many vertices and many edges, where the vertices represent the network junctions, starting and ending points of a road segment, and the edges' connect vertices. The cost (travel time) of traversing an edge is a function of the departure time. Many researches have focused on the problem of k nearest neighbor query and the variants in TDN. In order to reduce the complexity of the problem, the researches always assume that these functions satisfy the FIFO property (if these functions dissatisfy the FIFO property, the problem is proved NP-hard). In this subsection, we describe how to estimate the travel time functions based on the predicted speed classes of each road segment.

Firstly, the travel times are calculated by the length and predicted speed classes of the road segment. Take Fig. 6 as an example, the length of this road segment is 600 m. For the first time interval, the predicted speed classes is 3 which means that the corresponding speed is 4 m/s. So, the travel cost is 100 s. Then, by connecting the starting point of each time interval, multiple piecewise functions can be obtained easily. It is important to note that when generating these linear functions, the y value is the travel time, and the slope of each linear function must be larger than -1 to satisfy the FIFO property. This conclusion can be obtained from the following theorem.

Theorem 1 *The linear function $y = k \times x + b$ satisfies the FIFO property if $k > -1$, where x represents the entry time and y represents the travel time.*

Proof FIFO property means the person who has earlier entry time will leave early. Take Fig. 6 as an example, x_1 has earlier entry time than x_2 ($x_1 < x_2$). FIFO wants the departure time of x_1 earlier than x_2, which means $x_1 + y_1 < x_2 + y_2$. Now, we use the function to represent y_1 and y_2, and we can obtain the following inequation $x_1 + k \times x_1 + b < x_2 + k \times x_2 + b$. That is, $(k + 1)x_1 < (k + 1)x_2$. Since we have $x_1 < x_2$, the inequation $k + 1 > 0$ holds. Hence, the theorem is proved. □

When the calculated slope of the linear function is less than -1, we re-fit it by removing the breakpoint. For example in Fig. 6, assuming that the slope of segment bp_2 to bp_3 is less than -1 (the slope is larger than -1 based on the values in the figure, but we just make an assumption here), bp_2 will be deleted and the new linear function will be generated by connecting bp_1 and bp_2 directly.

Experimental Evaluation

Experiment Setup

In this section, the performance of the proposed prediction algorithm is evaluated. Moreover, we also compare the performance of our ELM-based prediction algorithm with the famous SVM.

As described in "Trajectory Mapping," we use the trajectory data generated by the taxis in Beijing and handle the historical data using the proposed method in "Historical Data Processing". In addition, the road network of Beijing contains 171,504 vertices and 433,391 edges. And the algorithms proposed are applied and implemented in C++ using STL. All the programs are run in a PC that has an Intel Quad Core 2.4GHZ CPU, 8 GB memory under Windows 10.

We choose the real data from 11 November to 17 November as the predicting data which contains the days from Monday to Sunday and the days from sunny day to rainy/snowy day, and the data of other days are treated as the training data. We issue six time intervals prediction queries on each time interval during these seven days for all the road segments, and compare the prediction results with the real speeds. The average prediction accuracy of each interval and the training time are illustrated in the next subsection.

The details of all the examined algorithms are introduced as follows:

- WAELM which is the prediction algorithm considering *w*eather conditions and the *a*djacent segment conditions using *ELM* to classify.
- WASVM which is the prediction algorithm considering *w*eather conditions and the *a*djacent segment conditions using *SVM* to classify.
- NAELM which is the prediction algorithm considering the *a*djacent segment conditions using *ELM* to classify but not take the weather conditions into account.
- WNELM which is the prediction algorithm considering the *w*eather conditions using *ELM* to classify but not take the adjacent segment conditions into account.

Table 1 summarizes the parameters mentioned earlier with their default values for those algorithms. It is worth mentioning that to improve the prediction accuracy, we divide the whole road network into many grids equal in size. Different models will be generated by the classifier using the historical trajectory data of the segments in different grids, and further used to predict the classes of speeds of the segments in the corresponding grid. That is to say, the historical trajectory data of the segments which are located in the same grid will be input to the classifier as training data, and the generated model will be used to predict the classes of speed of the segments in this grid. The parameter of *number of grids* describes the number of the total grids which range from 1 to 433,391, and the default value is 256.

Evaluation Results

Figure 7 illustrates the average prediction accuracy of the prediction algorithms with different intervals using SVM and ELM, respectively. The accuracy decreases as the duration time gets longer, this is because the travel speed is highly related with the latest traffic conditions and weather conditions. Hence, it is difficult to predict the speed 1 h later based on the current conditions like the weather forecasting. The accuracies of WASVM and WAELM both have a sudden drop on the third time interval, which shows that the

Table 1 Specifications of parameters for algorithms

Parameters	Default value
The length of a time interval	10 min
Number of hidden nodes for ELM	150
Number of classes of speeds	9
Number of grids	256

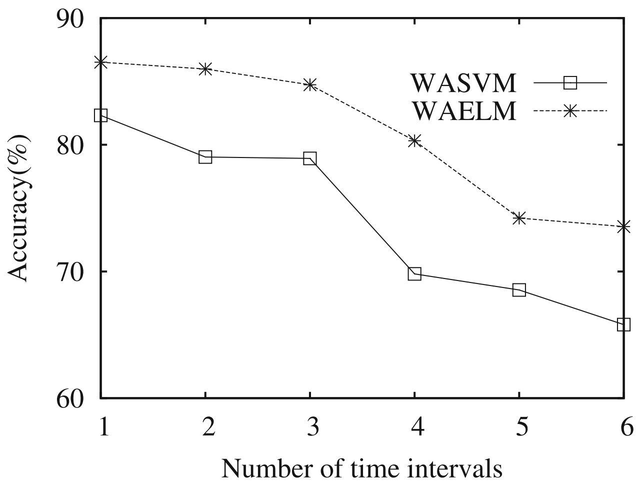

Fig. 7 Accuracy vs. number of time intervals

long-term prediction is still a challenge for the prediction methods. Moreover, the accuracy of WASVM is lower than that of WAELM by nearly 10% which indicates that the ELM classifier is slightly superior to SVM in accuracy on speed prediction.

Figures 8 and 9 illustrate the prediction accuracy and the required training time of WASVM and WAELM with regard to the number of grids. It is not surprising to see that the accuracies of the two algorithms both increase smoothly as the number of grids gets larger. This is because more grids mean more generated models for the same network, hence the prediction can be executed by the local model which is more suitable than the global one. The results also show that the travel speeds of the nearby segments are more likely to have the similar pattern. In the extreme case, the number of the grids is 433,391 which is equal to the number of edges of the network. This means that the classifier generates models for each segment, so the accuracy is higher, which is up to

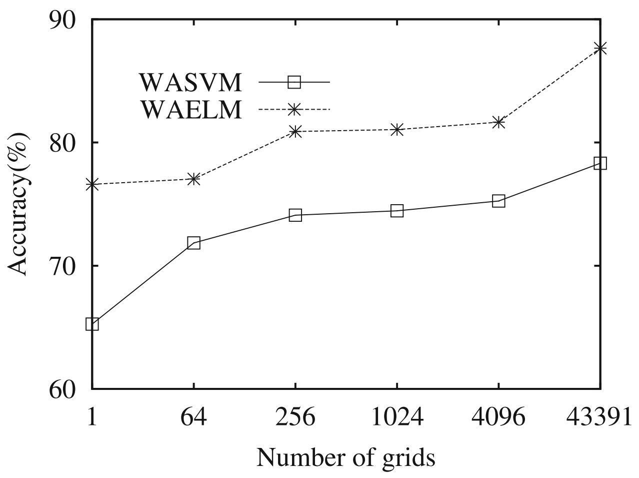

Fig. 8 Accuracy vs. number of grids

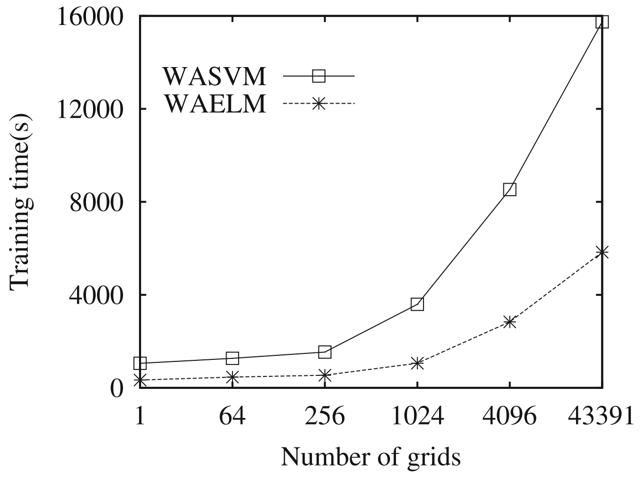

Fig. 9 Training time vs. number of grids

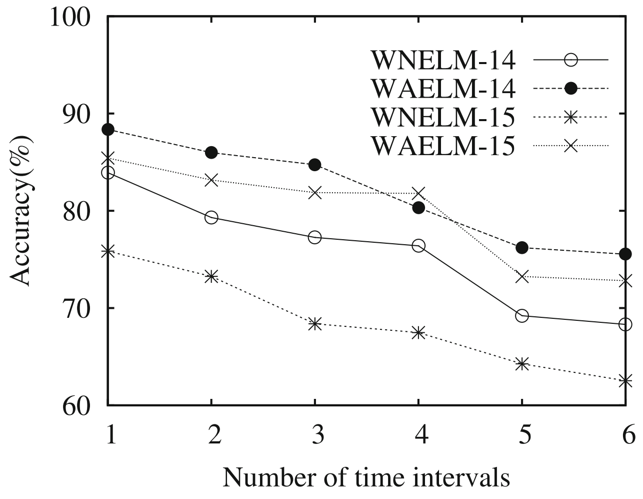

Fig. 11 Accuracy vs. Number of time intervals considering adjacent segments conditions

87% for WAELM. However, more generated models mean more times of iterations which bring more required training time as shown in Fig. 9. When the number of grids increases to 1024, the training time has a dramatic increase. This is why we choose 256 as the default value of the number of grids for other experiments.

To investigate the effect of weather conditions on the prediction accuracy, we compare NAELM with WAELM predicting only the two days of 14th (a sunny day) and 15th (a snowy day) November. Since the traffic conditions in the central region of a city are more sensitive to weather conditions than the suburbs', only the segments located in the center 4 grids are tested by the two algorithms. Figure 10 illustrates the results, from which we can see that the accuracy of NAELM is lower than that of WAELM by nearly 10% especially for the first half hour, which indicates that weather conditions have high impact on the

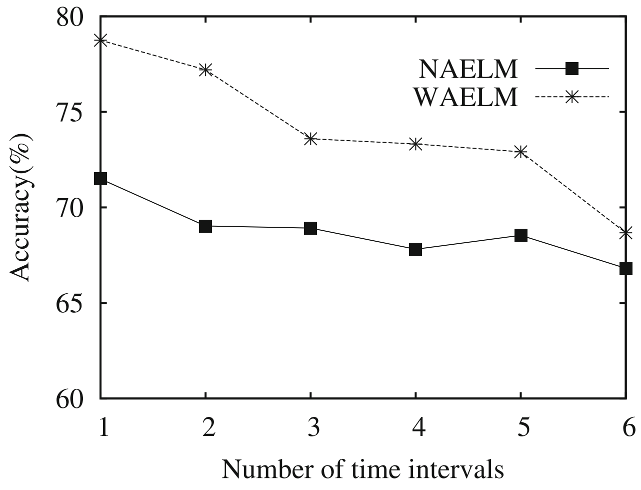

Fig. 10 Accuracy vs. number of time intervals considering weather conditions

travel speeds. For the last half hour, the gap of the accuracy of the two algorithms becomes smaller due to that the prediction for long-term becomes a bottleneck and the features of weather have a relatively smaller impact on the accuracy compared with the long time intervals. Moreover, the average accuracy is lower than the accuracy in Fig. 7 which shows that the traffic conditions at the center are more complex than the suburbs'.

Figure 11 investigates the effect of features of the adjacent segment conditions on the prediction accuracy. We test the average prediction accuracy of WNELM and WAELM for all the segments where the data of 14th and 15th November are regarded as the testing data. As expected, the accuracies of WNELM-14 and WNELM-15 are both lower than those of WAELM-14 and WAELM-15, respectively, which indicates that the conditions of the adjacent segments have an impact on the travel speed of the current segment. The difference of the date of 15th (a snowy day) is much larger due to that bad weather may slow the vehicle speeds and make the speeds more sensitive to the adjacent segments.

Conclusion

In this paper, we study the problem of predicting the travel time functions for each road segment in TDN from the taxi trajectory data. A new framework consisting of two parts is proposed. Some methods for processing the historical data are proposed in the offline part. And in the online part, a prediction algorithm based on classification using ELM is proposed to predict the speeds over 10-min intervals and up to 1 h in advance. The weather conditions and the adjacent segment conditions are both taken into account to improve the prediction accuracy. The results show that the performance of the prediction algorithm based on ELM classifier outperforms SVM classifier.

For future work, we will analyze and detect more factors that affect the traffic to improve the prediction accuracy, e.g., the road type, the road region, more types of changes of weather.

Funding Information This research was partially supported by the National Natural Science Foundation of China under Grant No. 61502317; and the Natural Science Foundation of Liaoning Province under Grant No.201602559; and the Natural Science Foundation of Liaoning Province under Grant 201602568; and the National Natural Science Foundation of China under Grant No. 61701322, 61502316.

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Informed Consent Informed consent was obtained from all individual participants.

Human and Animal Rights This article does not contain any studies involving human participants and/or animals by any of the authors.

References

1. Orda A, Rom R. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. J ACM (JACM). 1990;37(3):607–625.
2. Delling D. Time-dependent sharc-routing. In: European symposium on algorithms, pp 332–343. Springer. 2008.
3. Demiryurek U, Banaei-Kashani F, Shahabi C. Towards k-nearest neighbor search in time-dependent spatial network databases. In: International workshop on databases in networked information systems, pp. 296–310. Springer. 2010.
4. Komai Y, Nguyen DH, Hara T, Nishio S. knn search utilizing index of the minimum road travel time in time-dependent road networks. In: IEEE 33rd international symposium on reliable distributed systems workshops (SRDSW), pp 131–137. IEEE. 2014.
5. Li J, Liu X, Liu X, Xia X, Zhu R. Improved td-ftt algorithm based on dynamically selecting heuristic values. J Comput Appl. 2018;38(1):120–125.
6. Costa CF, Machado J, Nascimento MA, Macêdo JA. Aggregate k-nearest neighbors queries in time-dependent road networks. In: Proceedings of the 4th ACM SIGSPATIAL international workshop on mobile geographic information systems, pp 3–12 ACM. 2015.
7. Borutta F, Nascimento MA, Niedermayer J, Kröger P. Reverse k-nearest neighbour schedules in time-dependent road networks. In: Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems, p 27. ACM. 2015.
8. Li L, Hua W, Du X, Zhou X. Minimal on-road time route scheduling on time-dependent graphs. Proc VLDB Endowment. 2017;10(11):1274–1285.
9. Li L, Zheng K, Wang S, Hua W, Zhou X. Go slow to go fast: minimal on-road time route scheduling with parking facilities using historical trajectory. The International Journal on Very Large Data Bases. 2018;27(3):321–345.
10. Yang Y, Gao H, Yu JX, Li J. Finding the cost-optimal path with time constraint over time-dependent graphs. Proc VLDB Endowment. 2014;7(9):673–684.
11. Foschini L, Hershberger J, Suri S. On the complexity of time-dependent shortest paths. In: Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete algorithms pp. 327–341 SIAM. 2011.
12. Ding Y, Li Y, Deng K, Tan H, Yuan M, Ni LM. Detecting and analyzing urban regions with high impact of weather change on transport. IEEE Transactions on Big Data. 2017;3(2):126–139.
13. Zhao L, Ahmed A, Tang X, Lin N, Cuiwei L, Jiajia L. A weather-assisted driver experiences based path selection method. In: 2018 IEEE 20th international conference on high performance computing and communications (HPCC). IEEE. 2018.
14. Cintia P, Trasarti R, De Macedo JA, Almada L, Fereira C. A gravity model for speed estimation over road network. In: IEEE 14th international conference on mobile data management (MDM), vol 2, pp 136–141. IEEE. 2013.
15. Nascimento SM, Chucre MR, de Macedo JAF, Monteiro J, Casanova MA. On computing temporal functions for a time-dependent networks using trajectory data. In: Proceedings of the 20th international database engineering & applications symposium, pp 236–241. ACM. 2016.
16. Vlahogianni EI, Golias JC, Karlaftis MG. Short-term traffic forecasting: Overview of objectives and methods. Transp Rev. 2004;24(5):533–557.
17. Vlahogianni EI, Karlaftis MG, Golias JC. Short-term traffic forecasting: Where we are and where we are going. Transportation Research Part C: Emerging Technologies. 2014;43:3–19.
18. Chandra S, Al-Deek H. Cross-correlation analysis and multivariate prediction of spatial time series of freeway traffic speeds. Transportation Research Record: Journal of the Transportation Research Board. 2008;2061:64–76.
19. Zhang Y, Haghani A, Zeng X. Component garch models to account for seasonal patterns and uncertainties in travel-time prediction. IEEE Trans Intell Transp Syst. 2015;16(2):719–729.
20. Shang P, Li X, Kamae S. Chaotic analysis of traffic time series. Chaos, Solitons & Fractals. 2005;25(1):121–128.
21. Okutani I, Stephanedes YJ. Dynamic prediction of traffic volume through kalman filtering theory. Transp Res B Methodol. 1984;18(1):1–11.
22. Wang Y, Papageorgiou M, Messmer A. Renaissance–a unified macroscopic model-based approach to real-time freeway network traffic surveillance. Transportation Research Part C: Emerging Technologies. 2006;14(3):190–212.
23. Chien SI-J, Kuchipudi CM. Dynamic travel time prediction with real-time and historic data. J Transp Eng. 2003;129(6):608–616.
24. Yang F, Yin Z, Liu H, Ran B. Online recursive algorithm for short-term traffic prediction. Transportation Research Record: Journal of the Transportation Research Board. 2004;1879:1–8.
25. Ma X, Yu H, Wang Y, Wang Y. Large-scale transportation network congestion evolution prediction using deep learning theory. PloS One. 2015;10(3):e0119044.
26. Tang J, Liu F, Zou Y, Zhang W, Wang Y. An improved fuzzy neural network for traffic speed prediction considering periodic characteristic. IEEE Transactions on Intelligent Transportation Systems. 2017.
27. Wu C-H, Ho J-M, Lee D-T. Travel-time prediction with support vector regression. IEEE Trans Intell Transp Syst. 2004;5(4):276–281.
28. Asif MT, Dauwels J, Goh CY, Oran A, Fathi E, Xu M, Dhanya MM, Mitrovic N, Jaillet P. Spatiotemporal patterns in large-scale traffic speed prediction. IEEE Trans Intell Transp Syst. 2014;15(2):794–804.
29. Zhang Y, Liu Y. Traffic forecasting using least squares support vector machines. Transportmetrica. 2009;5(3):193–213.
30. Dimitriou L, Tsekeris T, Stathopoulos A. Adaptive hybrid fuzzy rule-based system approach for modeling and predicting

urban traffic flow. Transportation Research Part C: Emerging Technologies. 2008;16(5):554–573.
31. Zheng W, Lee D.-H., Shi Q. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. J Transp Eng. 2006;132(2):114–121.
32. Dong C, Richards SH, Yang Q, Shao C. Combining the statistical model and heuristic model to predict flow rate. J Transp Eng. 2014;140(7):04014023.
33. Sigakova K, Mbiydzenyuy G, Holmgren J. Impacts of traffic conditions on the performance of road freight transport. In: IEEE 18th international conference on intelligent transportation systems (ITSC), pp 2947–2952. IEEE. 2015.
34. Abdel-Aty MA, Pemmanaboina R. Calibrating a real-time traffic crash-prediction model using archived weather and its traffic data. IEEE Trans Intell Transp Syst. 2006;7(2):167–174.
35. Qiao W, Haghani A, Hamedi M. Short-term travel time prediction considering the effects of weather. Transportation Research Record: Journal of the Transportation Research Board. 2012;2308:61–72.
36. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: a new learning scheme of feedforward neural networks. In: 2004 IEEE international joint conference on neural networks, 2004. Proceedings. vol 2, pp 985–990 IEEE. 2004.
37. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: theory and applications. Neurocomputing. 2006;70(1):489–501.
38. Qu B.-Y., Lang B, Liang JJ, Qin AK, Crisalle OD. Two-hidden-layer extreme learning machine for regression and classification. Neurocomputing. 2016;175:826–834.
39. Huang G-B, Chen L. Letters: Convex incremental extreme learning machine. Neurocomputing. 2007;70(16-18):3056–3062.
40. Huang G.-B., Chen L. Enhanced random search based incremental extreme learning machine. Neurocomputing. 2008;71(16-18):3460–3468.
41. Liu N, Sakamoto JT, Cao J, Koh ZX, Ho AFW, Lin Z, Ong MEH. Ensemble-based risk scoring with extreme learning machine for prediction of adverse cardiac events. Cogn Comput. 2017;9(4):545–554.
42. Liu H, Fang J, Xu X, Sun F. Surface material recognition using active multi-modal extreme learning machine. Cognitive Computation, pp 1–14. https://link.springer.com/article/10.1007/s12559-018-9571-z. 2018.
43. Atli BG, Miche Y, Kalliola A, Oliver I, Holtmanns S, Lendasse A. Anomaly-based intrusion detection using extreme learning machine and aggregation of network traffic statistics in probability space. Cognitive Computation. 2018;10(5):848–863.
44. Liu Y, Vong CM, Wong PK. Extreme learning machine for huge hypotheses re-ranking in statistical machine translation. Cogn Comput. 2017;9(2):285–294.
45. Guo T, Zhang L, Tan X. Neuron pruning-based discriminative extreme learning machine for pattern classification. Cogn Comput. 2017;9(4):581–595.
46. Li J, Wang B, Wang G, Zhang Y. Probabilistic threshold query optimization based on threshold classification using elm for uncertain data. Neurocomputing. 2016;174:211–219.
47. Li J, Xia X, Liu X, Wang B, Zhou D, An Y. Probabilistic group nearest neighbor query optimization based on classification using elm. Neurocomputing. 2018;277:21–28.
48. Ban X, Guo C, Li G. Application of extreme learning machine on large scale traffic congestion prediction. In: Proceedings of ELM-2015 vol 1 pp 293–305. Springer. 2016.
49. Yuan J, Zheng Y, Zhang C, Xie X, Sun G-Z. An interactive-voting based map matching algorithm. In: 11th international conference on mobile data management (MDM), pp 43–52. IEEE. 2010.

Affiliations

Jiajia Li[1] · **Xiufeng Xia**[1] · **Xiangyu Liu**[1] · **Liang Zhao**[1] · **Botao Wang**[2]

[1] School of Computer Science, Shenyang Aerospace University, Shenyang, China

[2] School of Computer Science, Northeastern University, Shenyang, China