Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 829256, 7 pages
http://dx.doi.org/10.1155/2014/829256

Research Article

Trip Travel Time Forecasting Based on Selective Forgetting Extreme Learning Machine

Zhiming Gui and Haipeng Yu

School of Computer Science, Beijing University of Technology, Beijing 100124, China

Correspondence should be addressed to Zhiming Gui; zmgui@bjut.edu.cn

Received 21 September 2014; Revised 26 November 2014; Accepted 30 November 2014; Published 18 December 2014

Academic Editor: Tarek Ahmed-Ali

Travel time estimation on road networks is a valuable traffic metric. In this paper, we propose a machine learning based method for trip travel time estimation in road networks. The method uses the historical trip information extracted from taxis trace data as the training data. An optimized online sequential extreme machine, selective forgetting extreme learning machine, is adopted to make the prediction. Its selective forgetting learning ability enables the prediction algorithm to adapt to trip conditions changes well. Experimental results using real-life taxis trace data show that the forecasting model provides an effective and practical way for the travel time forecasting.

1. Introduction

Real-time estimation of the travel time between locations in city can help the individuals and transporters to plan their trips more accurately. Meanwhile, people are more likely to choose public transportation if they can know in advance that the practically quickest driving route to a destination would be still slower than the public transportation such as subway. It may affect their travels and schedules very much. Therefore, travel time prediction is important for both end users and governments aiming to ease traffic and protect environment [1].

Intuitively, taxi drivers are experienced in finding the quickest driving routes based on their knowledge and they generally know the routes between any two locations and often follow the same routes [2, 3]. Hence, historically recorded taxi trips should contain abundant information for predicting the duration for a future trip.

In this paper, we propose a machine learning based method to predict the travel time for a taxi at a given start time, origin and destination. Our approach consists of the following two steps.

(1) History trip information is extracted from taxis trajectories. The trips that have same origin/destination place are put together in chronological order. And then, we use an average on all durations for these trips whose start time is within the same time slot to represent the travel time of the two locations during the time slot.

(2) The trip information from step (1) is served as a training set for our prediction method. The selective forgetting extreme learning machine (SF-ELM) algorithm is adopted to provide an optimal estimate of the travel time.

To be noticed, the methods used in this paper are designed for individual travel schedule purpose. It has potential for using in traffic planning purposes. Dynamic travel time estimation is not considered in this paper.

The rest of the paper is organized as follows. In Section 2 we overview related work. In Section 3 we define the problem and explain our methodology. The experimental setup and evaluation are described in Section 4. We conclude in Section 5 with a summary and directions for future work.

2. Related Work

Quite a few researches have been proposed to estimate travel time between two locations using taxis trace. These works can be divided into two categories: road segment based

and path based. Road segment based method separated trip travel time into link travel times and intersection delays. However, explicitly modeling the time delay at an intersection is not easy. Thus some works just represent a trip by a sequence of connected road segments and estimate the trip travel time by the summation of the travel time of each individual road segment. Rahmani et al. [4] consider the correlation between different road segments in terms of their historical traffic patterns to infer the travel time on a road segment and the delay at intersections. Yuan et al. [5] propose a variant of road segment based method. Based on the trajectories generated by a large number of taxis, they build a landmark graph, where a node (entitled a landmark) is a road segment frequently traveled by taxis and an edge denotes the aggregation of taxis' commutes between two landmarks. The travel time of a path is then approximated by the summation of the travel times between landmarks.

Path-based trip travel time estimating approaches estimate the travel time of a path as a whole based on frequent trajectory patterns. It first mines frequent patterns from historical trajectories in advance [6] and then uses the average travel time of a pattern to represent the travel of the path corresponding to the pattern. A PTTE model based method is proposed to estimate the travel time of a path [7]. They infer the travel time of a road segment through a context-aware tensor decomposition approach at first and then search for the most optimal concatenation of trajectories for a query path using a dynamic programing solution. Though they infer the travel time for individual segments, the time is combined with trajectory patterns to formulate a subpath rather than simply concatenating them one by one. Based on this method, some work considered the frequent trajectory patterns as subpath and concatenated the subpaths into target path [8]. The travel time of a path is then approximated by the summation of the travel times of these subpaths. These approaches do not need to model intersection delay. However, the query paths may not fit into any patterns in current time slot as well as in the history. To be able to answer various query paths, these methods need to select more trajectory patterns by using a small support.

A few works use machine learning techniques to predict travel time. Blandin et al. use machine learning techniques and convex optimization to estimate arterial travel time [9]. Sampled travel time from probe vehicles is assumed to be known and served as a training set for a machine learning algorithm to provide a nonlinear estimate of the travel time. They use convex optimization to improve the performance of the nonlinear estimate through kernel regression. A dynamic Bayesian network based approach is proposed in [10] to estimate travel time on road links. The travel times on the road links are assumed to be independent and to be log-normally distributed, and the parameters are estimated using Markov chain Monte Carlo methods.

Most of these works focus on predicting travel time on road links or routes other than trips. In reality, individuals may not know the real route the taxi driver will choose. Furthermore, for a new trip, it is nearly impossible to find historical trips with the exact same path traversed for long journeys. In our work, we propose to use a trip based approach. We think that the changes of trip duration over time imply the traffic condition dynamics. So we use the history trip duration as training sample and use an optimized online sequential learning method to build the prediction model.

3. Methodology

3.1. Problem Statement. We aim to forecast the trip travel time of a certain origin and destination location at a future start time of a day based on the past trip travel time of that day. The insight we have for a viable solution is that the travel time between two locations at a certain time interval can be well predicted using its historical durations of four former intervals. However, trip durations can be affected by many factors, such as weather and time of day. In the current work, we assume that the history trip information implies this knowledge and only focuses on short term travel time forecasting.

Given a training set consisting of k samples $S = \{(x_n, t_n) \mid x_n \in R^u, t_n \in R^v, n = 1, 2, \ldots, k\}$, where x_n is a $u * 1$ input vector and t_n is a $v * 1$ target vector, $u = 4$. Each x_{ni} represents the trip duration during ith time interval of nth day and each t_{ni} represents the trip duration during target time interval of nth day. We would like to learn a function $h : R^u \rightarrow R^v$ which, given L, would provide an estimate of the travel time t for any $x \in R^u$. This is a typical regression problem.

3.2. Data Preparation. A taxi's trace is a series of state records in chronological order. Each state is sampled in a fixed time interval and consists of the following fields:

TAXI ID: the unique ID of sampled taxi;

GPS POSITION: the longitude and latitude of that taxi at the sampling time;

SPEED: the taxi speed at the sampling time, in kilometer per hour;

ORIENTATION: the direction of that taxi at the sampling time;

METER STATE: indicating whether the taxi is heavy at the sampling time, where 1 means the taxi is occupied (with passenger) and 0 means the taxi is empty (without passenger). METER STATE turning from 0 to 1 is a pick-up event and from 1 to 0 is a drop-off event;

TIME: the sampling time.

A taxi's trip contains the information beginning with a pick-up event and lasting until encountering a drop-off event. The target of the data preparation is to derive the sample training data from original taxi's trace. We need to put similar trips together to derive a sample observation.

DEFINITION 1 (trip): a trip Tr is a quaternion containing the following four items: start location (Tr.sl), start time (Tr.st), destination location (Tr.dl), and duration (Tr.du).

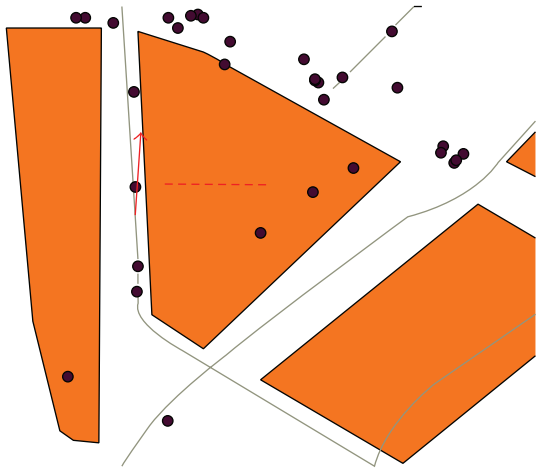


Figure 1: Determining the start/end region.

The start location, the end location, and the start time of a trip are the three basic features of each trip. If two trips have basic features similar to each other, they are similar. In the original taxis trace, the start and end location are recorded as points. The similarity between two points is meaningless. We use the start region and end region to replace the points. The start and end regions are determined by computing the nearest region at the right hand of taxi's move direction. The detailed implementation of this method can be found in many literatures. For example, the red arrow in Figure 1 shows the move direction and the red dotted line labels the right hand side of the move direction.

For the similarity between two time stamps, we choose to replace the time stamp with the time interval it belongs to. An hour-of-day time granularity is used in our work. Thus the definition of trip can be redefined by start time interval, start region, end region, and duration. Finally, we put the similar trips together and compute the average duration of them to derive a sample observation.

3.3. ELM, OS-ELM, and SF-ELM. Extreme learning machine is a learning algorithm for the single hidden layer feed forward neural networks used in classification and regression [11]. Originating from the batch learning extreme learning machine (ELM), OS-ELM inherits the advantage of ELM which can provide good generalization performance at an extremely fast learning speed. In addition, OS-ELM has the online sequential learning ability which does not require retraining when new data are received [12]. The SF-ELM (selective forgetting extreme learning machine) is proposed by Zhang and Wang in [13]. It adopts the latest training sample and weights the old training samples iteratively to insure that the influence of the old training samples is weakened. The output weight of SF-ELM is determined recursively during online training procedure according to its generalization performance.

(A) ELM. Given a training set consisting of k samples $S = \{(x_n, t_n) \mid x_n \in R^u, t_n \in R^v, n = 1, 2, \ldots, k\}$, where x_n is a $u * 1$ input vector and t_n is a $v * 1$ target vector. The number of nodes in hidden layer is L and $f()$ is the activate function:

$$\begin{aligned} \sum_{i=1}^{L} \beta_i f(a_i, b_i, x_1) &= t_1 \\ &\vdots \\ \sum_{i=1}^{L} \beta_i f(a_i, b_i, x_k) &= t_k, \end{aligned} \tag{1}$$

where a_i is the weight connecting the input nodes and the ith hidden node, b_i is the bias of ith hidden node, and β_i is the weight connecting the ith hidden node and the output nodes. Consequently, (1) can be written as

$$H_k \beta_k = T_k, \tag{2}$$

where

$$H_k = \begin{bmatrix} f(a_1, b_1, x_1) & \cdots & f(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ f(a_1, b_1, x_k) & \cdots & f(a_L, b_L, x_k) \end{bmatrix} \tag{3}$$

is hidden layer output matrix of the network and $\beta_k = [\beta_1, \beta_2, \ldots, \beta_L]^T$ and $T_k = [t_1, t_2, \ldots, t_k]^T$ are output weight matrix and target matrix, respectively:

$$\beta_k = (H_k^T H_k)^{-1} H_k^T T_k. \tag{4}$$

Finally, the prediction model can be formulated as

$$t = \sum_{i=1}^{L} \beta_i f(a_i, b_i, x). \tag{5}$$

(B) OS-ELM. OS-ELM, originated from basic ELM, is an online sequential learning algorithm that can learn data not only one-by-one but also chunk-by-chunk with fixed or varying chunk size [12]. It consists of two phases: initialization phase and sequential learning phase. In the initialization phase, a base ELM model is trained using a small chunk of initial training data. For instance, the output weight for an initial training dataset S_k with k training samples is obtained as

$$\beta_k = P_k H_k^T T_k, \tag{6}$$

where $P_k = (H_k^T H_k)^{-1}$. Then, in the sequential learning phase, when a new training data (x_{k+1}, t_{k+1}) arrives, calculate the $(k+1)$th hidden-layer output vector:

$$h_{k+1} = [f(a_1, b_1, x_{k+1}), f(a_2, b_2, x_{k+1}), \ldots, f(a_L, b_L, x_{k+1})]. \tag{7}$$

Then compute P_{k+1} by P_k and h_{k+1} as follows:

$$P_{k+1} = P_k - \frac{P_k h_{k+1}^T h_{k+1} P_k}{I + h_{k+1} P_k h_{k+1}^T}. \tag{8}$$

The output weights can be calculated recursively by

$$\beta_{k+1} = \beta_k + P_{k+1} h_{k+1}^T \left(t_{k+1} - h_{k+1}\beta_k\right). \tag{9}$$

In OS-ELM, the number of training data required in the initial phase has to be equal to or greater than the number of hidden nodes. The rank of H_k is required to be equal to the number of hidden nodes to ensure that OS-ELM can achieve the same learning performance as ELM.

(C) SF-ELM. SF-ELM is an extension algorithm of OS-ELM. It can selectively update the output weight based on a predefined allowable error. Assuming β_k has been obtained from the initial phase of OS-ELM, when new training data (x_{k+1}, t_{k+1}) arrives, β_{k+1} can be represented by

$$\begin{aligned}\beta_{k+1} &= \left(\begin{bmatrix} H_k \\ h_{k+1} \end{bmatrix}^T \begin{bmatrix} H_k \\ h_{k+1} \end{bmatrix}\right)^{-1} \begin{bmatrix} H_k \\ h_{k+1} \end{bmatrix}^T \begin{bmatrix} T_k \\ t_{k+1} \end{bmatrix} \\ &= \left(H_k^T H_k + h_{k+1}^T h_{k+1}\right)^{-1} \left(H_k^T T_k + h_{k+1}^T t_{k+1}\right),\end{aligned} \tag{10}$$

where $H_k^T H_k$ and $H_k^T T_k$ are obtained by the old training samples. To weaken the influence of old training samples, it adds weight to the two components. Equation (9) can be rewritten as

$$\beta_{k+1} = \left(\omega H_k^T H_k + h_{k+1}^T h_{k+1}\right)^{-1} \left(\omega H_k^T T_k + h_{k+1}^T t_{k+1}\right), \tag{11}$$

where ω is the forgetting factor and $0 < \omega < 1$; set

$$P_{k+1} = \left(\omega H_k^T H_k + h_{k+1}^T h_{k+1}\right)^{-1}. \tag{12}$$

When inverting both sides of (11), it can get

$$P_{k+1}^{-1} = \omega P_k^{-1} + h_{k+1}^T h_{k+1}. \tag{13}$$

Combining formula (12) with formula (10), the output weights can be calculated recursively as

$$\begin{aligned}\beta_{k+1} &= P_{k+1}\left(\omega H_k^T T_k + h_{k+1}^T t_{k+1}\right) \\ &= P_{k+1}\left(\omega P_k^{-1}\beta_k + h_{k+1}^T t_{k+1}\right) \\ &= P_{k+1}\left(\left(P_{k+1}^{-1} - h_{k+1}^T h_{k+1}\right)\beta_k + h_{k+1}^T t_{k+1}\right) \\ &= \beta_k + P_{k+1} h_{k+1}^T \left(t_{k+1} - h_{k+1}\beta_k\right).\end{aligned} \tag{14}$$

Applying Sherman-Morrison matrix inversion lemma to (11), the recursion formula of P_k can be written as

$$P_{k+1} = \frac{P_k}{\omega} - \frac{Q_k Q_k^T}{\omega\left(\omega + h_{k+1} Q_k\right)}, \tag{15}$$

where $Q_k = P_k h_{k+1}^T$. By (14) and (15), we can update the output weight β_{k+1} by the known output weight β_k and the newly coming training data (x_{k+1}, t_{k+1}). Based on the solid foundation presented above, our trip travel time estimation algorithm based on SF-ELM can be summarized as follows.

3.4. Proposed Forecasting Algorithm. Assume that the number of initial training data is larger than the number of hidden nodes and new training data are coming one by one. It should also be noted that there is one problem in OS-ELM and SF-ELM. If the term $H_k^T H_k$ is singular, then $P_k = \left(H_k^T H_k\right)^{-1}$ is unsolvable. To avoid this situation, we adopt the regularized idea in ReOS-ELM and add a small positive value to $H_k^T H_k$.

During the offline calibration phase, the historical trip duration sample data are used to build up the initial OS-ELM model. During the online phase, new coming trip information will be integrated with the initial OS-ELM model to selectively update and generate a revised model, in order to reflect the traffic dynamics.

(A) Offline Calibration Phase. Suppose that N days trip duration data $x_1, x_2, \ldots, x_N$ for a certain OD have been extracted from taxis trace, $N \geq L$, $x_i = [x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}]^T$, where x_{i1}, x_{i2}, x_{i3} and x_{i4} are trip duration values of the start four time intervals in the ith day respectively, x_{i5} is the trip duration value of the target time interval in the ith day. We reconstruct it to the sample training data as $(x_1, t_1), (x_2, t_2), \ldots, (x_k, t_k)$, where $x_i = [x_{i1}, x_{i2}, x_{i3}, x_{i4}]^T$ are adopted as the training input X, $t_i = x_{i5}$ are the training target or output of the model, and $k = N$. The detailed steps can be illustrated as follows.

Step 1. Assign random values for input parameters: input weights a_i and bias b_i, $i = 1, \ldots, L$.

Step 2. Calculate the initial hidden layer output matrix H_k.

Step 3. Then the initial output weight can be got from the $(x_1, t_1), (x_2, t_2), \ldots, (x_k, t_k)$ as

$$\beta_k = P_k H_k^T T_k, \tag{16}$$

where $P_k = (H_k^T H_k + \lambda I)^{-1}$, where I is the unit matrix and λ is the coefficients of ridge regression.

Although we have required $N \geq L$, we could not ensure that daily training examples are distinctive. According to the ridge regression theory, adding a small positive value into the diagonal $H_k^T H_k$ can avoid singular problem when the number of initial training data is less than the hidden nodes number. Thus the algorithm can also be effective if enough training samples are difficult to obtain ahead of time. In addition, the term λ can also control the relative importance between the training error and the norm of output weights [14].

(B) Online Calibration Phase. During this phase, new trip duration is statistically computed from trajectory data and adopted as online training samples. When a new x_{k+1} comes, the revised model can be obtained by the following steps.

Step 1. Compute the input vector h_{k+1}, and then the prediction value of t_{k+1} can be computed by

$$\bar{t}_{k+1} = h_{k+1}\beta_k. \tag{17}$$

Step 2. Calculate the output weight β_{k+1}. When the real value of t_{k+1} had been got from the new trip information, selectively update the P_k according to the following formula:

$$P_{k+1} = \begin{cases} \dfrac{P_k}{\omega} - \dfrac{Q_k Q_k^T}{\omega\left(\omega + h_{k+1} Q_k\right)} & E > \varepsilon \\ P_k & E \le \varepsilon, \end{cases} \tag{18}$$

where $E = |t_{k+1} - \bar{t}_{k+1}|$ and ε is the predefined threshold value. Then update the output weight β_k according to

$$\beta_{k+1} = \beta_k + P_{k+1} h_{k+1}^T \left(t_{k+1} - h_{k+1}\beta_k\right). \tag{19}$$

Step 3. Set $k = k + 1$ for the next online calibration.

4. Experiments and Analysis

4.1. The Dataset. In our experiments, we used real life dataset. The dataset consists of one-month GPS trajectories collected from over 30,000 taxis in Beijing between 01/11/2012 and 30/11/2012. The region data used in O/D extraction is the real residential areas of Beijing. In the current digital map dataset, Beijing has 590 regions as shown in Figure 2. We extracted trip information from the trace dataset at first and divide it into two datasets. Each day's trip data is composed of 24 time intervals of trip OD and duration information. The first 20 consecutive days of trip data are used as training dataset for the offline calibration. The rest of ten days' data is used for online calibration.

We cannot guarantee that there are sufficient taxis traversing between each O/D pair anytime even if we have a large number of taxis. To ensure that the experiment has enough sample data, we choose the OD pairs that traversed by enough taxis as the prediction target. The OD pairs that are connected by red line in Figure 3 are the pairs meeting the requirement. Ten OD pairs within downtown area are selected from these OD pairs and taken as the trip time prediction target. The straight line distance of these OD pairs ranges from 2 km to 10 km. For the four sample time intervals, we choose the time intervals between 7 and 11 o'clock.

4.2. Experiment Setting. In our experiment, the radial basis function (RBF) is chosen as the activation function. The optimal hidden node number is set to 7 through a tenfold cross-validation. We define the travel time estimation error to be the time lag between the real traveled time and the system estimated travel time. The forgetting factor ω is set to 0.98. The coefficient of ridge regression λ is set to 0.001.

4.3. Experiment Result and Analysis. With the selected setting, we run our algorithm with the sample dataset. At first, the algorithm is performed on history trips from each OD pair in our selected ten OD pairs to forecast its travel duration during two time intervals, respectively. We choose two typical time intervals, one during peak-off period and another during peak period, to do the prediction. Figure 4 shows the result.

In this round of experiment, the maximal percentage error of our prediction method is 19.48%. The real trip happened in peak time period. We checked the digital map and verified that the corresponding O/D pair is located at the surrounding areas of central business district. In Figure 5, the O/D pair is labeled by thick red line. The destination of the trip lies in the area which is labeled by blue rectangle. In this area, a jewelry exhibition was held in Beijing National Agricultural Center. It caused widespread congestion in the region.

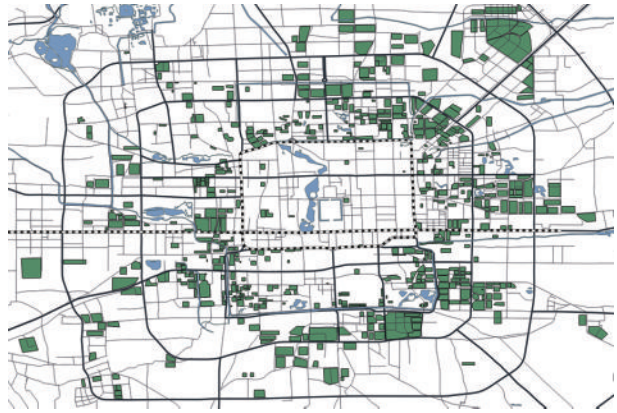
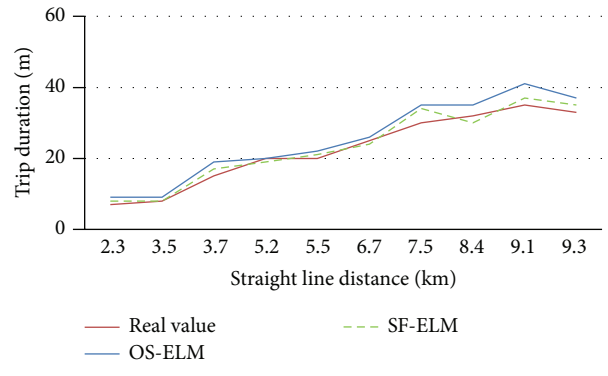


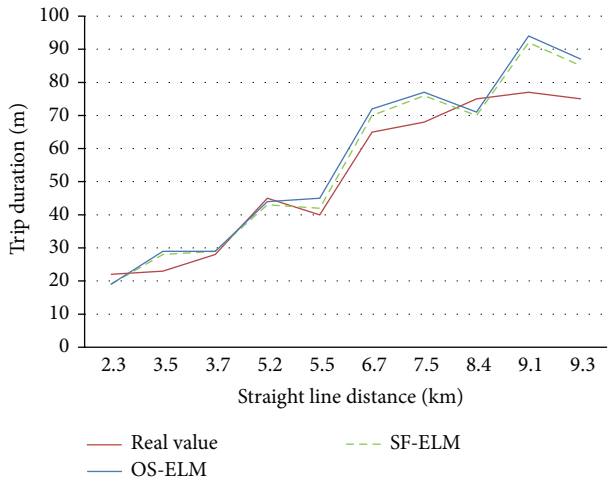
Figure 2: Residential areas distribution of Beijing city.

Figure 3: Trip count statistical.

4.3.1. Performance of Selective Update. To test the online learning and selective update ability of SF-ELM, we select one OD pair as the prediction target and make prediction for ten days. The result is shown in Figure 6. The degree of prediction error occurs randomly and could not reflect the online learning ability of SF-ELM well. The unforeseen traffic events account for part of the reasons. The comparison of computational cost between SF-ELM and OS-ELM has been listed in Table 1. The selective update ability of SF ELM enables it to skip one time's model update. Thus it achieves a higher efficiency than OS-ELM.



(a) 14:00-15:00



(b) 17:00-18:00

Figure 4: Trip durations of ten pairs OD on date 23.

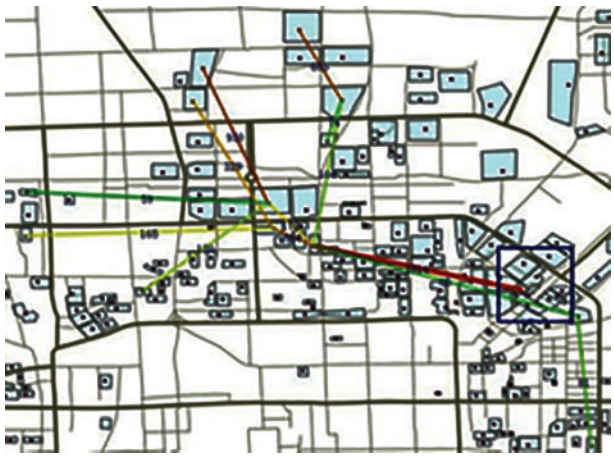
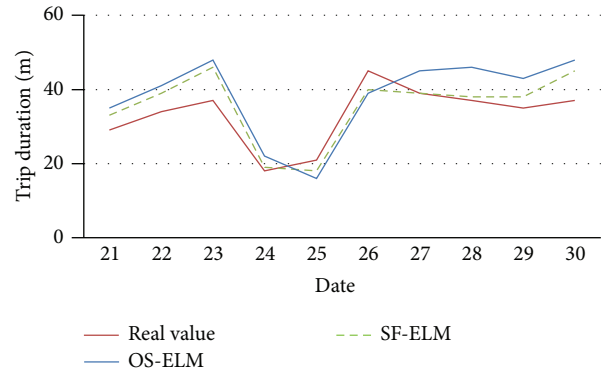
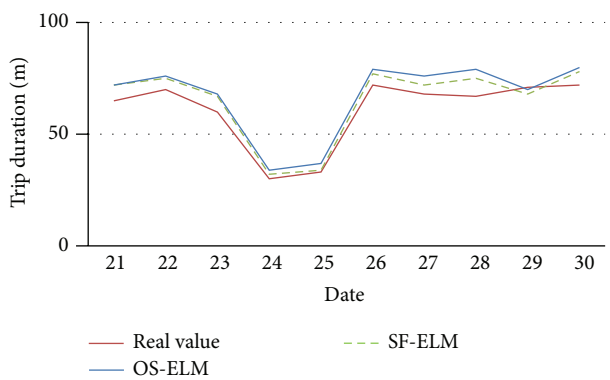


Figure 5: The OD pair with the maximum.

4.3.2. Influence of N and L. In our experiment, the training sample arrives in a one-by-one manner. Because the number of initial training data is much larger than the hidden nodes number L, we can only consider the case that $\text{Rank}(H_k) = \text{Rank}(P_K) = L$. To analyze the influence of N and L on the model, we assume that a larger number is assigned to L,



(a) 14:00-15:00



(b) 17:00-18:00

Figure 6: Trip durations of an OD pair in ten days.

Table 1: Computational cost between SL-ELM and OS-ELM.

Number of calibration points (offline + online)	Average testing time (S)	
	OS-ELM	SL-ELM
20 + 2	0.016	0.018
20 + 4	0.033	0.037
20 + 6	0.048	0.039
20 + 8	0.062	0.055
20 + 10	0.081	0.072

$L = 30$, and the matrix $H_k^T H_k$ tends to be singular. As λ is already introduced in our model, the rank of P_K remains at L due to the λI. However, when $L = 30$, the norm of output weights $\|\beta_k\|^2$ is much smaller than that of $L = 7$. The training error has been improved while the prediction error remains randomly in this case. Due to the limited training samples, the stablity of output weight norm is not presented in the current work.

5. Conclusion and Future Work

In this paper, we proposed a novel trip travel time forecasting algorithm based on SF-ELM. Trip travel times are modeled as average running travel times of all trips between the certain O/D other than the sum up of link travel times and intersection delays. Since road network components such

as traffic signals have significant effects on travel times and these factors are difficult to integrate into road link-based prediction model, our trip based model can indirectly reflect the trip conditions change and our methods are simple and practicable and can be used in engineering. The empirical results showed that it can provide a reasonable forecasting value in most cases. In the meanwhile, the selective and forgetting ability of SF-ELM made it possible to reflect and adapt to the trip condition changes better than OS-ELM.

ELM and its variant have received increasing attentions in recent years and many efforts have been dedicated to apply them in various applications [14–16]. Although they have obvious advantages in theory, the actual application field is limited at present [17]. Therefore, how to apply ELM to the daily life effectively is an important aspect in future research. For example, how to integrate various trip conditions in the forecasting model is understudied by us.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of paper.

References

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

[2] J. Yuan, Y. Zheng, C. Zhang et al., "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '10)*, pp. 99–108, 2010.

[3] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*, pp. 316–324, August 2011.

[4] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos, "Route travel time estimation using low-frequency floating car data," in *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems: Intelligent Transportation Systems for All Modes (ITSC '13)*, pp. 2292–2297, The Hague, Netherlands, October 2013.

[5] J. Yuan, Y. Zheng, and X. Xie, "Discovering regions of different functions in a city using human mobility and POIs," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*, pp. 186–194, August 2012.

[6] W. Luo, H. Tan, L. Chen, and L. M. Ni, "Finding time period-based most frequent path in big trajectory data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '13)*, pp. 713–724, 2013.

[7] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," *Proceedings of the Conference on Knowledge Discovery and Data Mining (SIGKDD '14)*, pp. 25–34, 2014.

[8] A. Hofleitner and A. Bayen, "Optimal decomposition of travel times measured by probe vehicles using a statistical traffic flow model," in *Proceedings of the 14th IEEE International Intelligent Transportation Systems Conference (ITSC '11)*, pp. 815–821, October 2011.

[9] S. Blandin, L. El Ghaoui, and A. Bayen, "Kernel regression for travel time estimation via convex optimization," in *Proceedings of the 48th IEEE Conference on Decision and Control held jointly with 28th Chinese Control Conference (CDC/CCC '09)*, pp. 4360–4365, Shanghai, China, December 2009.

[10] A. Hofleitner, R. Herring, P. Abbeel, and A. Bayen, "Learning the dynamics of arterial traffic from probe data using a dynamic bayesian network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1679–1693, 2012.

[11] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[12] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[13] X. Zhang and H.-L. Wang, "Selective forgetting extreme learning machine and its application to time series prediction," *Acta Physica Sinica*, vol. 60, no. 8, Article ID 080504, 2011.

[14] P. K. Wong, C. M. Vong, X. H. Gao, and K. I. Wong, "Adaptive control using fully online sequential-extreme learning machine and a case study on engine air-fuel ratio regulation," *Mathematical Problems in Engineering*, vol. 2014, Article ID 246964, 11 pages, 2014.

[15] N. Liu and H. Wang, "Evolutionary extreme learning machine and its application to image analysis," *Journal of Signal Processing Systems*, vol. 73, no. 1, pp. 73–81, 2013.

[16] L. Mao, L. Zhang, X. Liu, C. Li, and H. Yang, "Improved extreme learning machine and its application in image quality assessment," *Mathematical Problems in Engineering*, vol. 2014, Article ID 426152, 7 pages, 2014.

[17] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie, "Extreme learning machine: algorithm, theory and applications," *Artificial Intelligence Review*, 2013.