ETCPS: An Effective and Scalable Traffic Condition Prediction System

Dong Wang, Wei Cao, Mengwen Xu, and Jian Li(✉)

Institute for Interdisciplinary Information Sciences, Tsinghua University, 10084 Beijing, BJ, China
{wang-dong12,cao-w13,xmw12}@mails.tsinghua.edu.cn, lijian83@mail.tsinghua.edu.cn

Abstract. Real-time prediction of the traffic condition is an important ingredient for a variety of applications. In this paper, 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. We conduct extensive experimental evaluations with a large GPS data set collected from more than 12,000 taxis in Beijing during two months. The experimental results demonstrate the effectiveness, efficiency, and scalability of our system.

1 Introduction

Real-time prediction of the traffic condition becomes increasingly important. A well-performed traffic condition prediction system is the fundamental ingredient of various real applications. Examples include the traffic management [6], routing service [13], taxi ride sharing [8] etc. Such problem has been widely studied in recent years [1,10,11,15]. Generally, given the current and historical traffic conditions of the road network, our goal is to predict the traffic condition of each road after a few minutes or hours.

Most prior works on traffic condition prediction are based on the data generated by the road side loop sensors. However, such loop sensors are usually expensive and only embedded in highways and part of urban main roads. Alternatively, ubiquitous location based services enable us to collect a large volume of traffic data from GPS-embedded devices. Such GPS data provides valuable information for analyzing and predicting the traffic conditions. 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.

S.B. Navathe et al. (Eds.): DASFAA 2016, Part II, LNCS 9643, pp. 419–436, 2016.
DOI: 10.1007/978-3-319-32049-6_26

In this paper, we study the efficient and scalable models for traffic condition prediction based on the GPS data collected from floating vehicles (taxis in our data). To make our exposition more concrete, we first illustrate several challenges in our problem.

- Large volume of GPS data has been generated routinely, especially for some metropolises such as New York or Beijing. Most prior works are based on probabilistic graphical models [3,5,9]. The state spaces explode in these algorithms under very large scale datasets. Thus, it takes a very long time to run the algorithms.
- The traffic conditions and their transition patterns (i.e., the patterns in which the traffic condition varies) for each road vary significantly under different time intervals. For example, if the traffic is in a jam during a peak hour, it usually lasts for a long time. However, if such congestion happens in a non-peak hour, the traffic usually become light soon. Such traffic pattern is changing over time. Prior works based on the Markov Chain and Hidden Markov Model (HMM) [5,9,11] can not capture such feature since the states of transition matrices are not related with time.
- The taxis sometimes slow down or even stop for picking or attracting the passengers. It is hard to distinguish whether such low travel speed is due to the congestion of the traffic. Such records may lead to erroneous estimations of the traffic condition.

To address the above challenges, we propose the *Ensemble based Traffic Condition Prediction System (ETCPS)*. Our system combines two different models called *Predictive Regression Tree (PR-Tree)* and *Spatial Temporal Probabilistic Graphical Model (STPGM)*. We summarize our technical contributions below:

- We present two useful observations in the traffic condition time series which are the bases of our design. We first present the correlations between the gaps of the traffic condition and its expected traffic condition. Then, we show the autocorrelations in the first order difference of the traffic condition series (See Sect. 2).
- We propose a regression tree based model called PR-Tree. PR-Tree can effectively capture the proposed correlations and thus predict the traffic conditions with a high accuracy. PR-Tree is very efficient on large scale datasets. Given a training set with 10^5 roads, it only takes 3.26 min to train a PR-Tree and the prediction of PR-Tree is real-time (See Sect. 5).
- We propose a probabilistic graphical model called STPGM. STPGM can capture the correlations between adjacent roads. It formulates the state transitions in different time intervals separately. Thus, the state space for STPGM is much smaller than the prior works [3,5,9]. On the other hand, STPGM captures different traffic patterns in different time intervals. We show that in the experiment STPGM is more efficient and accurate than the algorithms in prior works (See Sect. 6).
- We propose a prediction system called ETCPS which combines PR-Tree and STPGM. We evaluate our model with real dataset which consists of GPS

points generated by over 12,000 taxis collected in two months. It provides an experimental evidence that ETCPS is efficient, scalable in terms of supporting large size road networks, and achieves a high-quality prediction (See Sect. 7).

2 Preliminary

Road Network. We are given a data set consisting of GPS records of taxis. The GPS records of the j-th taxi is represented by $Tr_j = \{p_1, p_2, \ldots, p_{|Tr_j|}\}$. Each p_i represents a GPS record $(cid, time, location, speed)$ indicating the id of the j-th car, the time stamp when the record is generated, the latitude and longitude of the current location and the instantaneous speed respectively. We define a real urban road network as a directed graph $G = (V, E)$ where V is the set of nodes representing the terminal points of road segments and E is the set of road segments. A road segment r_i is a directed edge associated with a start point v_s, an end point v_e with length l_i. See Fig. 1 for an illustration. Utilizing the technique of map-matching [7], each GPS record p_i on the trajectory Tr_j can be located to a road segment r_i in which the car j is traveling on.

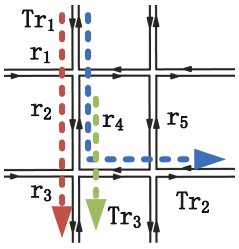


Fig. 1. Time cost

Table 1. Time cost (Million seconds)

Traj	Time (Intv)	Road segment	Speed (km/h)	Traj	Time (Intv)	Road segment	Speed (km/h)
Tr_1	34	r_1	56	Tr_2	34	r_1	60
Tr_1	35	r_2	60	Tr_2	35	r_2	58
Tr_1	35	r_3	61	Tr_2	35	r_4	58
Tr_3	35	r_2	15	Tr_2	36	r_5	60
Tr_3	35	r_3	60				

Traffic Condition. We define the traffic condition for a road segment r_i during a specific period as below. Given a GPS data set collected during D days, we split the period of D days into several intervals, and each time interval spans λ minutes. We assume that the traffic condition of a specific road segment remains unchanged in one interval. Such assumption is widely used in the transportation literature [11,15].

As each day has $M = \frac{60 \cdot 24}{\lambda}$ time intervals, for a GPS data set collected during D days, there are $T = M \cdot D$ time intervals. The t-th interval is $[t \cdot \lambda, (t+1) \cdot \lambda)$. For example, if we set $\lambda = 15, D = 31$, then we have $M = 96$, $T = 2976$, and the interval 34 is a time period from $8:30$ to $8:45$ in the first day.

By mapping each GPS record to a road segment, we consider the average speed of all the records observed in the t-th interval on a road segment. For example, in Table 1, the observed average speed for r_2 in the 35-th interval is $(60 + 58 + 15)/3$. However, some taxis may run at a very low speed or even

stop for *boarding* or *balling* when the road is not congested. We regard such records as the noise which is eliminated in the pre-processing stage (see Sect. 8 for details). Then, the *traffic condition* of a road 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 Sect. 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 $226 \bmod 96 = 34$. 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 1, 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.

3 Useful Observations

Most of prior works predict the future traffic conditions directly based on the traffic condition time series. However, it is difficult to extract the patterns in the traffic condition time series Org^i. We find that by transforming the Org^i into two different 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 traffic 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 traffic condition series by subtracting the corresponding expected traffic condition from each of

the traffic conditions. Specifically, we set $g_t^i = o_t^i - a_t^i$ and we thus obtain a new series $\text{Gap}^{\text{i}} = \{g_t^i | t = 1, \ldots, T\}$. Intuitively, if $g_t < 0$, it means that the traffic condition in the time interval t is more congested than usual. We find that there exists a strong correlation between g_{t+1} and g_t. Figures 2 and 3 show the scatter diagram of (o_t, o_{t+1}) and (g_t, g_{t+1}) of a specific road segment respectively. As we can see, by transforming the traffic condition series Org^{i} to the gap series Gap^{i}, we essentially extract the "pattern" of the traffic condition series.

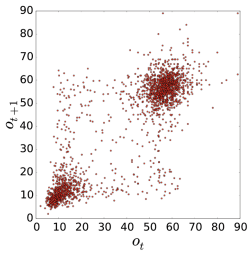


Fig. 2. o_t and o_{t+1}

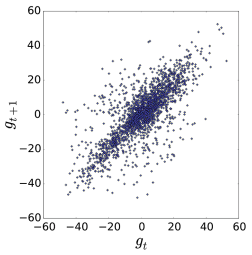


Fig. 3. g_t and g_{t+1}

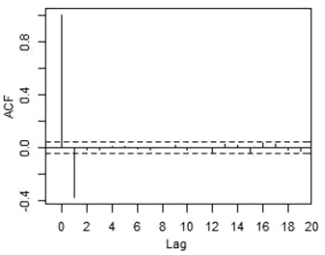


Fig. 4. ACF of Diff(Org)

First Order Difference of Traffic Condition Series. We use $\delta_t^i = o_t^i - o_{t-1}^i$ to represent the first order difference of traffic condition series, denoted as Diff(Org). We use ACF (Auto Correlation Function) to analyze the autocorrelation in the time series of δ_t^i. The autocorrelation of a random process describes the correlation between values of the process at different times with a time lag τ. Given a time series and time lag, ACF returns a value between $+1$ (total positive correlation) and -1 (total negative correlation) inclusive. If the absolute value of ACF is beyond ± 0.05, we usually think the time series is autocorrelated at time lag τ. In Fig. 4, we show the ACF value of the time series δ_t of a random road segment. The horizontal axis represents the time lag τ, and vertical axis represents the ACF value at lag τ. As the ACF value at lag $\tau = 1$ is far beyond the threshold -0.05, we conclude that there exists a correlation between δ_t and δ_{t+1}.

4 System Overview

The framework of our proposed traffic condition prediction system is illustrated in Fig. 5. We develop a system that utilizes the historical and real time taxi GPS records to estimate the current travel condition and predict the travel conditions in the next time intervals. It is composed of four major components: Pre-processing, Predictive Regression Tree Model (PR-Tree), Spatial Temporal Probabilistic Graphical Model (STPGM) and Ensemble.

In the pre-processing phase, first, we map match the GPS trajectories to road networks using the ST-Matching algorithm [11]. Then, we eliminate the

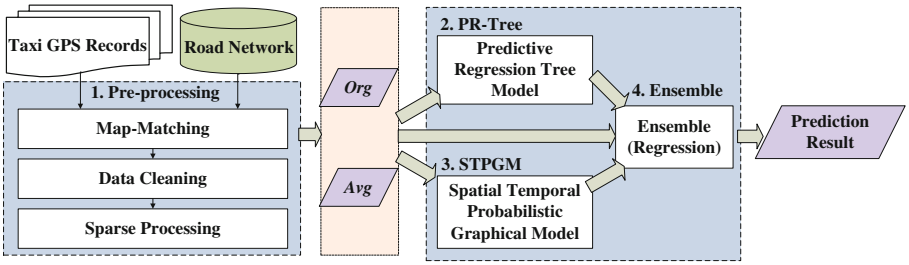


Fig. 5. Overview of system framework.

records which are under boarding or balling state. We then deal with the sparsity issure that no GPS record is observed for some roads during some time intervals. With the pre-processing, we thus obtain two time series Org and Avg as defined in Sect. 2. The details are presented in the experiment part (Sect. 8). Next, in Sect. 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 Sect. 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 Sect. 7. We show that combining two different models enhances the accuracy of the prediction in Sect. 8.

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 Sect. 5.1. Then in Sect. 5.2, we present the training algorithm of PR-Tree.

5.1 Description of PR-Tree

Recall that the time series Gap shows a strong autocorrelation as we claimed in Sect. 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 Sect. 2). From Fig. 3, 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 closed to -10, g_{t+1} is usually around 1.2 times g_t whereas if g_t is closed to -8, g_{t+1} is usually around 1.4 times g_t. Motived 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 inner node of PR-tree has a splitting value and each leaf node has a 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 return the corresponding θ on the leaf node as $R(g_t)$. For simplicity, we use R to represent the corresponding PR-Tree.

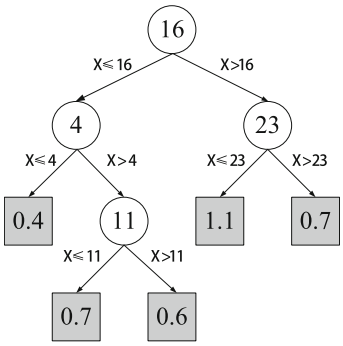


Fig. 6. An example of PR-Tree

We show an example of a PR-Tree in Fig. 6. 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 $p_{t+1} = a_{t+1} + R(g_t)$. Figure 6 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 $a_{t+1} + R(g_t) \cdot g_t = 46.5$.

5.2 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 optimal PR-Tree (function R^*) that

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

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

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 $\mathrm{argmin}_\alpha\ f(S_x, \alpha)$, 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.,

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

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. Otherwise, we terminate the recursion.

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. If the qualification on CV does not decrease, we undo the splitting operation (line 19 to line 21) and terminate the recursion. Otherwise, we split its children nodes recursively (line 14 to line 17). See Algorithm 1 for the pseudo code.

6 Predicting Traffic Condition with STPGM

Despite that the PR-Tree performs well in most of our data (which we show in Sect. 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 its self 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.

6.1 States of STPGM

We first discretize the traffic conditions into different states. Recall that as we claimed in Sect. 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 [3,5,11,13]), 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-Medoids where k is a parameter to be specified (see Sect. 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.

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.,

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

For the prediction, it is unnecessary to compute the denominator, which we show in Sect. 6.3. As for the numerator, the state space in Eq. 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 Eq. 2 by

$$P(s_{t+1}^i = c_{i,t+1}^{x_i}) \prod_{j=1}^{n} P(s_{i_j}^t = c_{i_j,t}^{x_{i_j}} | s_{t+1}^i = c_{x_i,t+1}^i) \tag{3}$$

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 mod 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

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

Thus, we get the approximation of the numerator of Eq. 2.

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 Eq. 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 Eq. 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.

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^i_t, 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^i_{t+1} by $o^i_t \cdot \theta(o^i_t)$ and we predict the traffic condition directly use $\theta(o^i_t)$. 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 Sect. 2) as our input or the traffic conditions filtered with Kalman filtering. The details are presented in Sect. 8.

8 Experimental Study

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

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 [7,14] to road network[2] of Beijing. 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 $= \frac{1}{|E|}\sum_{i=1}^{|E|}\sum_{t=1}^{T}|p^i_t - o^i_t|$, MRE $= \frac{1}{|E|}\sum_{i=1}^{|E|}\sum_{t=1}^{T}|p^i_t - o^i_t|/o^i_t$, MSE $= \frac{1}{|E|}\sum_{i=1}^{|E|}\sum_{t=1}^{T}(p^i_t - o^i_t)^2$. 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 2.1 GHz for each core and 32 GB memory on Ubuntu 14.04 LTS operate system).

[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.

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 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.

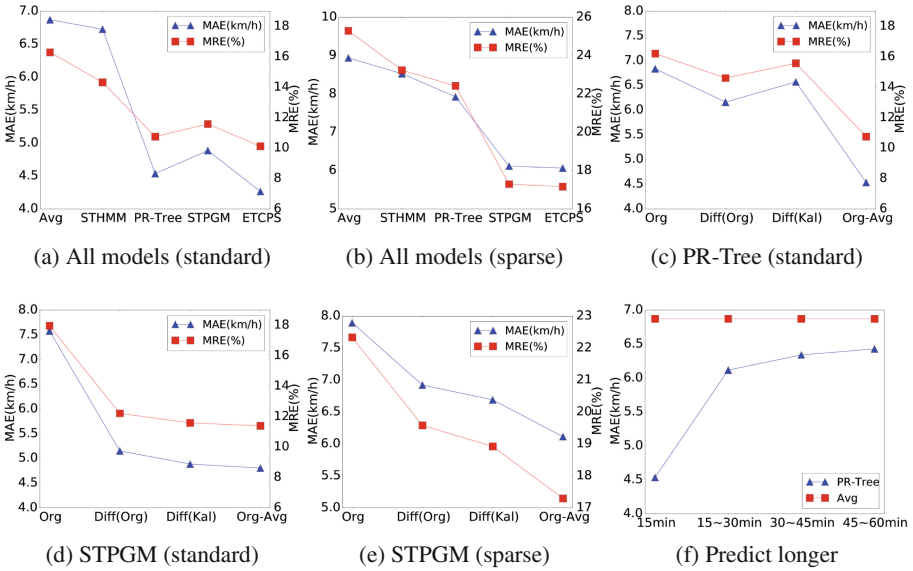
Deal with Sparsity. Recall that as we claimed in Sect. 2, some road segments may not contain any GPS record during the time interval t for some $t \in [T]$. Thus, the corresponding traffic condition o_t^i is not defined. To solve this issue, for the road segment r_i, if the GPS record set observed in 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 $t \bmod M$-th interval in each day. We define $\bar{a}_t^i$ as the mean of $\overline{A_t^i}$ and the series $\text{Bias} = \{b_t = \bar{o}_t^i - \bar{a}_t^i | \forall \bar{o}_t^i \neq -1\}$. 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 $o_t^i = \bar{a}_t^i + b_t$.

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. [11] 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. 7a, 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. 7a whereas in the sparse case STPGM performs better. By combining PR-Tree and STPGM, our system ETCPS achieves the best performs in both two cases.

Verifying the Observed Patterns. Recall that as we claimed in Sect. 7, any time series related with traffic can be taken as the input of both PR-Tree and STPGM, and predict the traffic condition in the proper way. To illustrate the



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

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

Fig. 7. Performance analysis.

effects of the observations which we proposed in Sect. 3, we design four different experiments with different time series and evaluate each experiment on PR-Tree and STPGM respectively. The first two time series are Org and Gap = Org−Avg, as we used in Sects. 5 and 6. Then, we use the first order difference of Org as the input time series, denoted as Diff(Org). The t-th element in Diff(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 filtering to process the traffic condition series Org. We take the first order difference of the processed time series as the input as well, denoted as Diff(Kal).

We show the experimental results in Fig. 7c–e. Both PR-Tree and STPGM perform badly if we use Org as input directly. However, by using Diff(Org) and Gap instead, the performances improve significantly which verifies our observations.

Predict Longer Time Intervals. The PR-Tree model can be also used to predict the traffic conditions in the longer term. Given observations in interval t denoted as o_t, we first obtain the predicted traffic condition p_{t+1} and we take p_{t+1} as the "true traffic condition" in the time interval $t + 1$ and obtain p_{t+2}. Iteratively, we obtain the prediction after m time intervals p_{t+m}. In Fig. 7, we show the performance of PR-Tree in predicting the traffic condition in the next 0 to 60 min and comparing with the Avg method. As m increases, the performance becomes worse, but it is still better than Avg.

Effects of Time and Road Length. Figure 8 shows the effectiveness of our prediction across time. We plot the average mean squared error of travel speed

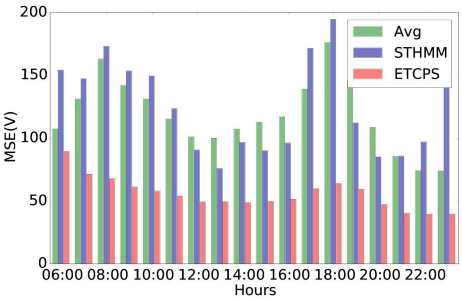


Fig. 8. MSE varies over day

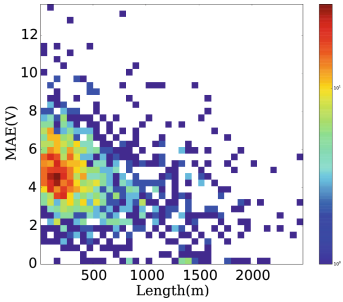


Fig. 9. RMAE varies over road length

(MSE) for the baseline Avg, STHMM and ETCPS respectively during different hours for all days. The result shows that our system outperforms both the baseline and STHMM.

To illustrate the effectiveness of the road length, in the Fig. 9, we show the relation between MAE and the length of road segments. The result shows that the road segments with longer length tend to have smaller MAE, i.e., our prediction performs better for the road segments with longer lengths.

Running Time. Since the predictions of both PR-Tree and STPGM are simple which can be done in real time, we only present the running time for training our models in Fig. 10 and Table 2. From Table 2, we can see that the training time cost of PR-Tree is very small. It takes only 3.26 min to process 10^5 roads. However, STPGM takes a much longer time to train as shown in Table 2. Especially for the state formulation phase, clustering the traffic conditions is time costing. It takes 176.6 min to process the state formulation phase for 10^5 roads. We stress that SHTMM applies a complicated state formulation algorithm and the state space is much larger than STPGM. In our data set, the time consuming of SHTMM is 1718 ms per road whereas even for STPGM, it only takes 13.3 ms per road to train the model.

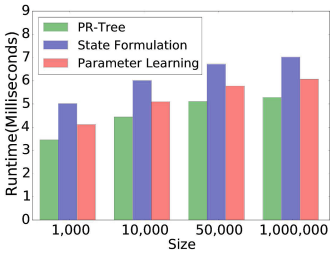


Fig. 10. Time cost

Table 2. Time cost (Minutes)

Size	PR-Tree	State Formulation	Parameter learning
10^3	0.04	1.72	0.22
10^4	0.46	17.46	2.09
$5 * 10^4$	2.14	88.1	10.09
10^5	3.26	176.6	19.61

9 Related Work

In this section, we review the related existing works. Most of prior works use the probabilistic models to predict the traffic conditions. Hunter et al. [4] formulated the traffic condition prediction in the arterial network to a maximum likelihood problem and estimated the travel time distributions based on the observed route travel times. Yeon et al. [12] 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 [9].

To capture the correlations between road segments, Hofleitner et al. [3] 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. [13] 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. [11]. 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 (Sect. 8), we show that our model outperform STHMM in both the efficiency and accuracy. We stress that Chu et al. [2] 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 [1,10]. Wang et al. [10] 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. [1] 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.

10 Conclusion

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.

Acknowledgment. This work was supported in part by the National Basic Research Program of China grants 2015CB358700, 2011CBA00300, 2011CBA00301, and the National NSFC grants 61033001, 61361136003.

References

1. Asghari, M., Emrich, T., Demiryurek, U., Shahabi, C.: Probabilistic estimation of link travel times in dynamic road networks. In: ACM SIGSPATIAL (2015)
2. Chu, V.W., Wong, R.K., Liu, W., Chen, F.: Causal structure discovery for spatio-temporal data. In: Bhowmick, S.S., Dyreson, C.E., Jensen, C.S., Lee, M.L., Muliantara, A., Thalheim, B. (eds.) DASFAA 2014, Part I. LNCS, vol. 8421, pp. 236–250. Springer, Heidelberg (2014)
3. Hofleitner, A., Herring, R., Abbeel, P., Bayen, A.: Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network. IEEE Trans. Intell. Transp. Syst. **13**(4), 1679–1693 (2012)
4. Hunter, T., Herring, R., Abbeel, P., Bayen, A.: Path and travel time inference from GPS probe vehicle data. NIPS Anal. Netw. Learn. Graphs **12**(1) (2009)
5. Kwon, J., Murphy, K.: Modeling freeway traffic with coupled HMMs. Technical report, University of California, Berkeley (2000)
6. Leontiadis, I., Marfia, G., Mack, D., Pau, G., Mascolo, C., Gerla, M.: On the effectiveness of an opportunistic traffic management system for vehicular networks. IEEE Trans. Intell. Transp. Syst. **12**(4), 1537–1548 (2011)
7. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate GPS trajectories. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 352–361. ACM (2009)
8. Ma, S., Zheng, Y., Wolfson, O.: T-share: a large-scale dynamic taxi ridesharing service. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 410–421. IEEE (2013)
9. Ramezani, M., Geroliminis, N.: On the estimation of arterial route travel time distribution with Markov chains. Transp. Res. Part B: Methodol. **46**(10), 1576–1590 (2012)
10. Wang, Y., Zheng, Y., Xue, Y.: Travel time estimation of a path using sparse trajectories. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 25–34. ACM (2014)

11. Yang, B., Guo, C., Jensen, C.S.: Travel cost inference from sparse, spatio temporally correlated time series using Markov models. Proc. VLDB Endow. **6**(9), 769–780 (2013)
12. Yeon, J., Elefteriadou, L., Lawphongpanich, S.: Travel time estimation on a freeway using discrete time Markov chains. Transp. Res. Part B: Methodol. **42**(4), 325–338 (2008)
13. Yuan, J., Zheng, Y., Xie, X., Sun, G.: T-drive: enhancing driving directions with taxi drivers' intelligence. IEEE Trans. Knowl. Data Eng. **25**(1), 220–232 (2013)
14. Yuan, J., Zheng, Y., Zhang, C., Xie, X., Sun, G.Z.: An interactive-voting based map matching algorithm. In: Proceedings of the 2010 Eleventh International Conference on Mobile Data Management, pp. 43–52. IEEE Computer Society (2010)
15. Zheng, W., Lee, D.H., Shi, Q.: Short-term freeway traffic flow prediction: Bayesian combined neural network approach. J. Transp. Eng. **132**(2), 114–121 (2006)