Latent Space Model for Road Networks to Predict Time-Varying Traffic

Dingxiong Deng[1], Cyrus Shahabi[1], Ugur Demiryurek[1], Linhong Zhu[2], Rose Yu[1], Yan Liu[1]
Department of Computer Science, University of Southern California[1]
Information Sciences Institute, University of Southern California[2]
{dingxiod, shahabi, demiryur, qiyu, yanliu.cs}@usc.edu, linhong@isi.edu

ABSTRACT

Real-time traffic prediction from high-fidelity spatiotemporal traffic sensor datasets is an important problem for intelligent transportation systems and sustainability. However, it is challenging due to the complex topological dependencies and high dynamism associated with changing road conditions. In this paper, we propose a Latent Space Model for Road Networks (LSM-RN) to address these challenges holistically. In particular, given a series of road network snapshots, we learn the attributes of vertices in latent spaces which capture both topological and temporal properties. As these latent attributes are time-dependent, they can estimate how traffic patterns form and evolve. In addition, we present an incremental online algorithm which sequentially and adaptively learns the latent attributes from the temporal graph changes. Our framework enables real-time traffic prediction by 1) exploiting real-time sensor readings to adjust/update the existing latent spaces, and 2) training as data arrives and making predictions on-the-fly. By conducting extensive experiments with a large volume of real-world traffic sensor data, we demonstrate the superiority of our framework for real-time traffic prediction on large road networks over competitors as well as baseline graph-based LSM's.

Keywords

Latent space model, real-time traffic forecasting, road network

1. INTRODUCTION

Recent advances in traffic sensing technology have enabled the acquisition of high-fidelity spatiotemporal traffic datasets. For example, at our research center, for the past five years, we have been collecting data from 15000 loop detectors installed on the highways and arterial streets of Los Angeles County, covering 3420 miles cumulatively (see the case study in [13]). The collected data include several main traffic parameters such as occupancy, volume, and speed at the rate of 1 reading/sensor/min. These large scale data streams enable accurate traffic prediction, which in turn improves route navigation, traffic regulation, urban planning, etc.

The *traffic prediction* problem aims to predict the future travel speed of each and every edge of a road network, given the historical speed readings from the sensors on these edges. To solve the traffic prediction problem, the majority of existing techniques utilize the historical information of an edge to predict its future travel-speed using regression techniques such as Auto-regressive Integrated Moving Average (ARIMA) [18], Support Vector Regression (SVR) [20] and Gaussian Process (GP) [31]. There are also studies that leverage spatial/topological similarities to predict the readings of an edge based on its neighbors in either the Euclidean space [10] or the network space [14]. Even though there are few notable exceptions such as Hidden Markov Model (HMM) [14, 28] that predict traffic of edges by collectively inferring temporal information, these approaches simply combine the local information of neighbors with temporal information. Furthermore, existing approaches such as GP and HMM are computationally expensive and require repeated offline trainings. Therefore, it is very difficult to adapt the models to real-time traffic forecasting.

Motivated by these challenges, we propose Latent Space Modeling for Road Networks (**LSM-RN**), which enables more accurate and scalable traffic prediction by utilizing both topology similarity and temporal correlations. Specifically, with LSM-RN, vertices of dynamic road network are embedded into a latent space, where two vertices that are similar in terms of both time-series traffic behavior and the road network topology are close to each other in the latent space. Recently, Latent Space Modeling has been successfully applied to several real-world problems such as community detection [24, 29], link prediction [17, 33] and sentiment analysis [32]. Among them, the work on social networks [17, 24, 33] (hereafter called **LSM-SN**) is most related to ours because in both scenarios data are represented as graphs and each vertex of these graphs has different attributes. However, none of the approaches to LSM-SN are suitable for both identifying the edge and/or sensor latent attributes in road networks and exploiting them for real-time traffic prediction due to the following reasons.

First, road networks show significant topological (e.g., travel-speeds between two sensors on the same road segment are similar), and temporal (e.g., travel-speeds measured every 1 minute on a particular sensor are similar) correlations. These correlations can be exploited to alleviate the missing data problem, which is unique to road networks, due to the fact that some road segments may contain no sensors and any sensor may occasionally fail to report data. Second, unlike social networks, LSM-RN is fast evolving due to the time-varying traffic conditions. On the contrary, social networks evolve smoothly and frequent changes are very unlikely (e.g., one user changes its political preferences twice a day). Instead, in road networks, traffic conditions on a particular road segment can change rapidly in a short time (i.e., time-dependent) because of rush/non-rush hours and traffic incidents. Third, LSM-RN is highly dynamic where fresh data come in a streaming fashion, whereas the connections (weights) between nodes in social net-

KDD '16, August 13-17, 2016, San Francisco, CA, USA

DOI: http://dx.doi.org/10.1145/2939672.2939860

works are mostly static. The dynamic nature requires frequent model updates (e.g., per minute), which necessitates partial updates of the model as opposed to the time-consuming full updates in LSM-SN. Finally, with LSM-RN, the ground truth can be observed shortly after making the prediction (by measuring the actual speed later in future), which also provides an opportunity to improve/adjust the model incrementally (i.e., online learning).

With our proposed LSM-RN, each dimension of the embedded latent space represents a latent attribute. Thus the attribute distribution of vertices and how the attributes interact with each other jointly determine the underlying traffic pattern. To enforce the topology of road network, LSM-RN adds a graph Laplacian constraint which not only enables global graph similarity, but also completes the missing data by a set of similar edges with non-zero readings. Subsequently, we incorporate the temporal properties into our LSM-RN model by considering time-dependent latent attributes and a global transition process. With these time-dependent latent attributes and the transition matrix, we are able to better model how traffic patterns form and evolve.

To infer the time-dependent latent attributes of our LSM-RN model, a typical method is to utilize multiplicative algorithms [16] based on Non-negative Matrix Factorization, where we jointly infer the whole latent attributes via iterative updates until they become stable, termed as *global learning*. However, global learning is not only slow but also not practical for real-time traffic prediction. This is because, traffic data are of high-fidelity (i.e., updates are frequent in every one minute) and the actual ground-truth of traffic speed becomes available shortly afterwards (e.g., after making a prediction for the next five minutes, the ground truth data will be available instantly after five minutes). We thus propose an incremental online learning with which we sequentially and adaptively learn the latent attributes from the temporal traffic changes. In particular, each time when our algorithm makes a prediction with the latent attributes learned from the previous snapshot, it receives feedback from the next snapshot (i.e., the ground truth speed reading we already obtained) and subsequently modifies the latent attributes for more accurate predictions. Unlike traditional online learning which only performs one single update (e.g., update one vertex per prediction) per round, our goal is to make predictions for the entire road network, and thus the proposed online algorithm allows updating latent attributes of many correlated vertices simultaneously.

Leveraging global and incremental learning algorithms our LSM-RN model can strike a balance between accuracy and efficiency for real-time forecasting. Specifically, we consider a setting with a predefined time window where at each time window (e.g., 5 minutes), we learn our traffic model with the proposed incremental inference approach on-the-fly, and make predictions for the next time span. Meanwhile, we batch the re-computation of our traffic model at the end of one large time window (e.g., one hour). Under this setting, our LSM-RN model enables the following two properties: (1) real-time feedback information can be seamlessly incorporated into our framework to adjust for existing latent spaces, thus allowing for more accurate predictions, and (2) our algorithms perform training and predictions on-the-fly with small amount of data rather than requiring large training datasets.

We conducted extensive experiments on a large scale of real-world traffic sensor dataset. We demonstrated that the LSM-RN framework achieves better accuracy than that of both existing time series methods (e.g. ARIMA and SVR) and the LSM-SN approaches. Moreover, we show that our algorithm scales to large road networks. For example, it only takes 4 seconds to make a prediction for a network with 19,986 edges. Finally, we show that our batch window setting works perfectly for streaming data, alternating the executions of our global and incremental algorithms, which strikes a compromise between prediction accuracy and efficiency. For instance, incremental learning is one order of magnitude faster than global learning, and it requires less than 1 seconds to incorporate real-time feedback information.

The remainder of this paper is organized as follows. We discuss the related work in Section 2 and define our problem in Section 3. and explain LSM-RN in Section 4. We present the global learning and increment learning algorithms, and discuss how to adapt our algorithms for real-time traffic forecasting in Section 5. In Section 6, we report the experiment results and conclude the paper afterwards.

2. BACKGROUND AND RELATED WORKS

2.1 Traffic analysis

Many studies have been conducted to address the traffic prediction problem, but no single study so far has tackled all the challenges in a holistic manner. Some focused on missing values [19] or missing sensors [30], but not both. Some studies [18, 31] utilize temporal data which models each sensor (or edge) independently and makes predictions using time series approaches (e.g., ARIMA [18], SVR [20] and GP [31]). For instance, Pan et. al. [18] learns an enhanced ARIMA model for each edge in advance, and then performs traffic prediction on top of these models. Very few studies [14, 28] utilize spatiotemporal model with correlated time series based on Hidden Markov Model, but only for small number of time series and not always using the network space as the spatial dimension (e.g., using Euclidean space [10]). In [27], Xu et. al. consider using the newly arrived data as feedback to reward one classifier vs. the other but not for dynamically updating the model. Note that many existing studies [5, 12, 25, 28] on traffic prediction are based on GPS dataset, which is different with the sensor dataset, where we have fine-grained and steady readings from road-equipped sensors. We are not aware of any study that applies latent space modeling (considering both time and network topology) to real-time traffic prediction from incomplete (i.e., missing sensors and values) sensor datasets.

2.2 Latent space model and NMF

Recently, many real data analytic problems such as community detection [24,29], recommendation system [6], topic modeling [22], image clustering [4], and sentiment analysis [32], have been formulated as the problem of latent space learning. These studies assume that, given a graph, each vertex resides in a latent space with attributes, and vertices which are close to each other are more likely to be in the same cluster (e.g., community or topic) and form a link. In particular, the objective is to infer the latent matrix by minimizing the difference (e.g., squared loss [29, 32] or KL-divergence [4]) between observed and estimated links. However, existing methods are not designed for the highly correlated (topologically and temporally) and dynamic road networks. Few studies [21] have considered the temporal relationships in SN with the assumption that networks evolve over time. The temporal graph snapshots in [21] are treated separately and thus newly observed data are not incorporated to improve the model. Compared with existing works, we explore the feasibility of modeling road networks with time-varying latent space. The traffic speed of a road segment is determined by their latent attributes and the interaction between corresponding attributes. To tackle the sparsity of road network, we utilize the graph topology by adding a graph Laplacian constraint to impute the missing values. In addition, the latent position of each vertex, varies over time and allows for sudden movement from one timestamp to the next timestamp via a transition matrix.

Different techniques have been proposed to learn the latent properties, where Non-negative Matrix Factorization (NMF) is one of

the most popular methods thanks to ease of interpretability and flexibility. In this work, we explore the feasibility of applying dynamic NMF to traffic prediction domain. We design a global algorithm to infer the latent space based on the traditional multiplicative algorithm [9, 16]. We further propose a topology-aware incremental algorithm, which adaptively updates the latent space representation for each node in the road network with topology constraints. The proposed algorithms differ from traditional online NMF algorithms such as [3], which independently perform the online update.

Notations	Explanations
$\mathcal{N}, n$	road network, number of vertices of the road network
G	the adjacency matrix of a graph
U	latent space matrix
B	attribute interaction matrix
A	the transition matrix
k	the number of dimensions of latent attributes
T	the number of snapshots
$span$	the gap between two continuous graph snapshots
h	the prediction horizon
λ, γ	regularization parameters for graph Laplacian and transition process

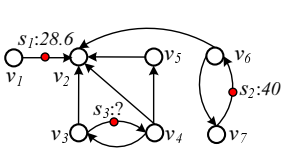
Table 1: Notations and explanations

3. PROBLEM DEFINITION

We denote a road network as a directed graph $\mathcal{N} = (V, E)$, where V is the set of vertices and $E \in V \times V$ is the set of edges, respectively. A vertex $v_i \in V$ models a road intersection or an end of road. An edge $e(v_i, v_j)$, which connects two vertices, represents a directed network segment. Each edge $e(v_i, v_j)$ is associated with a travel speed $c(v_i, v_j)$ (e.g., 40 miles/hour). In addition, $\mathcal{N}$ has a corresponding adjacency matrix representation, denoted as G, whose $(i, j)^{\text{th}}$ entry represents the edge weight between the i^{th} and j^{th} vertices.

The road network snapshots are constructed from a large-scale, high resolution traffic sensor dataset (see detailed description of sensor data in Section 6). Specifically, a sensor s (i.e., a loop detector) is located at one segment of road network $\mathcal{N}$, which provides a reading (e.g., 40 miles/hour) per sampling rate (e.g., 1 min). We divide one day into different intervals, where $span$ is the length of each time interval. For example, when $span = 5$ minutes, we have 288 time intervals per day. For each time interval t, we aggregate (i.e., average) the readings of one sensor. Subsequently, for each edge segment of network $\mathcal{N}$, we average all sensor readings located at that edge as its weight. Therefore, at each timestamp t, we have a road network snapshot G_t from traffic sensors.

Example. *Figure 1 (a) shows a simple road network with 7 vertices and 10 edges at one timestamp. Three sensors (i.e., s_1, s_2, s_3) are located in edges (v_1, v_2), (v_3, v_4) and (v_7, v_6) respectively, and each sensor provides an aggregated reading during the time interval. Figure 1(b) shows the corresponding adjacent matrix after mapping the sensor readings to the road segments. Note that the sensor dataset is incomplete with both missing values (i.e., sensor fails to report data) and missing sensors (i.e., edges without any sensors). Here sensor s_3 fails to provide reading, thus the edge weight of $c(v_3, v_4)$ is ? due to missing value. In addition, the edge weight of $c(v_3, v_2)$ is marked as $\times$ because of missing sensors.*



(a) An abstract road network $\mathcal{N}$

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	28.6	0	0	0	0	0
v_2	0	0	0	0	0	0	0
v_3	0	x	0	?	0	0	0
v_4	0	x	0	0	x	0	0
v_5	0	x	0	0	0	0	0
v_6	0	x	0	0	0	0	x
v_7	0	0	0	0	40	0	0

(b) Adjacency matrix representation G

Figure 1: An example of road network

Given a small number of road network snapshots, or a dynamic road network, our objective is to predict the future traffic conditions. Specifically, a *dynamic road network*, is a sequence of snapshots $(G_1, G_2, \cdots, G_T)$ with edge weights denoting time-dependent travel speed.

With a dynamic road network, we formally define the problem of edge traffic prediction with missing data as follows:

Problem 1 *Given a dynamic road network $(G_1, G_2, \cdots, G_T)$ with missing data at each timestamp, we aim to achieve the following two goals:*

- *complete the missing data (i.e., both missing value and sensor) of G_i , where $1 \leq i \leq T$;*
- *predict the future readings of G_{T+h}, where h is the prediction horizon. For example, when $h = 1$, we predict the traffic condition of G_{T+1} at the next timestamp.*

For ease of presentation, Table 1 lists the notations we use throughout this paper. Note that since each dimension of a latent space represents a latent attribute, we thus use latent attributes and latent positions interchangeably.

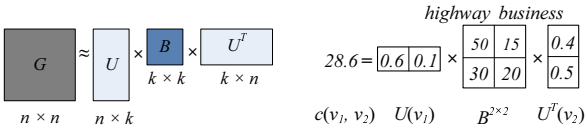
4. LATENT SPACE MODEL FOR ROAD NETWORKS (LSM-RN)

In this section, we describe our LSM-RN model in the context of traffic prediction. We first introduce the basic latent space model (Section 4.1) by considering the graph topology, and then incorporate both temporal and transition patterns (Section 4.2). Finally, we describe the complete LSM-RN model to solve the traffic prediction problem with missing data (Section 4.3).

4.1 Topology in LSM-RN

Our traffic model is built upon the latent space model of the observed road network. Basically, each vertex of road network have different attributes and each vertex has an overlapping representation of attributes. The attributes of vertices and how each attribute interacts with others jointly determine the underlying traffic patterns. Intuitively, if two highway vertices are connected, their corresponding interaction generates a higher travel speed than that of two vertices located at arterial streets. In particular, given a snapshot of road network G, we aim to learn two matrices U and B, where matrix $U \in R_+^{n \times k}$ denotes the latent attributes of vertices, and matrix $B \in R_+^{k \times k}$ denotes the attribute interaction patterns. The product of UBU^T represents the traffic speed between any two vertices, where we use to approximate G. Note that B is an asymmetric matrix since the road network G is directed. Therefore, the basic traffic model which considers the graph topology can be determined by solving the following optimization problem:

$$\arg\min_{U \geq 0, B \geq 0} J = ||G - UBU^T||_F^2 \quad (1)$$



(a) Basic model (b) Travel time of $c(v_1, v_2)$

Figure 2: An example of our traffic model, where G represents a road network, U denotes the attributes of vertices in the road network, n is number of nodes, and k is number of attributes, and B denotes how one type of attributes interacts with others.

Similar Non-negative Tri-factorization frameworks have been utilized in clustering [9], community detection [29] and sentimental analysis [32]. Figure 2 (a) illustrates the intuition of our static traffic model. As shown in Figure 2 (b), suppose we know that each

vertex is associated with two attributes (e.g., highway and business area), and the interaction pattern between two attributes is encoded in matrix B, we can accurately estimate the travel speed between vertex v_1 and v_2, using their latent attributes and the matrix B.

Overcome the sparsity of Road Network. In our road network, G is very sparse (i.e., zero entries dominate the items in G) for the following reasons: (1) the average degree of a road network is small [26], and thus the edges of road network is far from fully connected, (2) the distribution of sensors is non-uniform, and only a small number of edges are equipped with sensors; and (3) there exists missing values (for those edges equipped with sensors) due to the failure and/or maintenance of sensors.

Therefore, we define our loss function only on edges with observed readings, that is, the set of edges with travel cost $c(v_i, v_j) > 0$. In addition, we also propose an in-filling method to reduce the gap between the input road network and the estimated road network. We consider graph Laplacian dynamics, which is an effective smoothing approach for finding global structure similarity [15]. Specifically, we construct a graph Laplacian matrix L, defined as $L = D - W$, where W is a graph proximity matrix that is constructed from the network topology, and D is a diagonal matrix $D_{ii} = \sum_j (W_{ij})$. With these new constraints, our traffic model for one snapshot of road network G is expressed as follows:

$$\arg\min_{U,B} J = ||Y \odot (G - UBU^T)||_F^2 + \lambda Tr(U^T L U), \qquad (2)$$

where Y is an indication matrix for all the non-zero entries in G, i.e, $Y_{ij} = 1$ if and only if $G(i,j) > 0$; $\odot$ is the Hadamard product operator, i.e., $(X \odot Z)_{ij} = X_{ij} \times Z_{ij}$; and λ is the Laplacian regularization parameter.

4.2 Time in LSM-RN

Next, we will incorporate the temporal information, including time-dependent modeling of latent attributes and the temporal transition. With this model, each vertex is represented in a unified latent space, where each dimension either represents a spatial or temporal attribute.

4.2.1 Temporal effect of latent attributes

The behavior of the vertices of road networks may evolve quickly. For instance, the behavior of a vertex that is similar to that of a highway vertex during normal traffic condition, may become similar to that of an arterial street node during congestion hours. Because the behavior of each vertex can change over time, we must employ a time-dependent modeling for attributes of vertices for real-time traffic prediction. Therefore, we add the time-dependent effect of attributes into our traffic model. Specifically, for each $t \leq T$, we aim to learn a corresponding time-dependent latent attribute representation U_t. Although the latent attribute matrix U_t is time-dependent, we assume that the attribute interaction matrix B is an inherent property, and thus we opt to fix B for all timestamps. By incorporating this temporal effect, we obtain our model based on the following optimization problem:

$$\arg\min_{U_t,B} J = \sum_{t=1}^{T} ||Y_t \odot (G_t - U_t B U_t^T)||_F^2 + \sum_{t=1}^{T} \lambda Tr(U_t L U_t^T) \qquad (3)$$

4.2.2 Transition matrix

Due to the dynamics of traffic condition, we aim to learn not only the time-dependent latent attributes, but also a transition model to capture the evolving behavior from one snapshot to the next. The transition should capture both periodic evolving patterns (e.g., morning/afternoon rush hours) and non-recurring patterns caused by traffic incidents (e.g., accidents, road construction, or work zone closures). For example, during the interval of an accident, a vertex transition from the normal state to the congested at the beginning, then become normal again after the accident is cleared.

We thus assume a global process to capture the state transitions. Specifically, we use a matrix A that approximates the changes of U between time $t-1$ to time t, i.e., $U_t = U_{t-1}A$, where $U \in R_+^{n \times k}, A \in R_+^{k \times k}$. The transition matrix A represents how likely a vertex is to transit from attribute i to attribute j from timestamp 1 to timestamp T.

4.3 LSM-RN Model

Considering all the above discussions, the final objective function for our LSM-RN model is defined as follows:

$$\arg\min_{U_t,B,A} J = \sum_{t=1}^{T} ||Y_t \odot (G_t - U_t B U_t^T)||_F^2 + \sum_{t=1}^{T} \lambda Tr(U_t L U_t^T) + \sum_{t=2}^{T} \gamma ||U_t - U_{t-1}A||_F^2 \qquad (4)$$

where λ and γ are the regularization parameters.

By solving Eq. 4, we obtain the learned matrices of U_t, B and A from our LSM-RN model. Consequently, the task of both missing value and sensor completion can be accomplished by the following:

$$G_t = U_t B U_t^T, \qquad \text{when } 1 \leq t \leq T. \qquad (5)$$

Subsequently, the edge traffic for snapshot G_{T+h} (where h is the number of future time spans) can be predicted as follows:

$$G_{T+h} = (U_T A^h) B (U_T A^h)^T \qquad (6)$$

5. LEARNING&PREDICTION BY LSM-RN

In this section, we first present a typical global multiplicative algorithm to infer the LSM-RN model, and then discuss a fast incremental algorithm that scales to large road networks.

5.1 Global learning algorithm

We develop an iterative update algorithm to solve Eq. 4, which belongs to the category of traditional multiplicative update algorithm [16]. By adopting the methods from [16], we can derive the update rule of U_t, B and A. The details of derivation can be found in the technical report [8].

Lemma 1 *The Update rule of U_t, B and A can be expressed as follows:*

$$(U_t) \leftarrow (U_t) \odot \Big(\frac{(Y_t \odot G)(U_t B^T + U_t B) + \lambda W U_t + \gamma (U_{t-1}A + U_{t+1}A^T)}{(Y_t \odot U_t B U_t^T)(U_t B^T + U_t B) + \lambda D U_t + \gamma (U_t + U_t A A^T)} \Big)^{\frac{1}{4}} \qquad (7)$$

$$B \leftarrow B \odot \Big(\frac{\sum_{t=1}^{T} U_t^T (Y_t \odot G_t) U_t}{\sum_{t=1}^{T} U_t^T (Y_t \odot (U_t B U_t^T)) U_t} \Big) \qquad (8)$$

$$A \leftarrow A \odot \Big(\frac{\sum_{t=1}^{T} U_{t-1}^T U_t}{\sum_{t=1}^{T} U_{t-1}^T U_{t-1} A} \Big) \qquad (9)$$

Algorithm 1 outlines the process of updating each matrix using aforementioned multiplicative rules to optimize Eq. 4. The general idea is to jointly infer and cyclically update all the latent attribute matrices U_t, B and A. In particular, we first jointly learn the latent attributes for each time t from all the graph snapshots (Lines 3–4). Based on the sequence of time-dependent latent attributes ($i.e., U_1, U_2, \cdots, U_T$), we then learn the global attribute interaction pattern B and the transition matrix A (Lines 5–6).

From Algorithm 1, we now explain how our LSM-RN model jointly learns the spatial and temporal properties. Specifically, when

we update the latent attribute of one vertex $U_t(i)$, the spatial property is preserved by (1) considering the latent positions of its adjacent vertices ($Y_t \odot G_t$), and (2) incorporating the local graph Laplacian constraint (i.e., matrix W and D). Moreover, the temporal property of one vertex is then captured by leveraging its latent attribute in the previous and next timestamps (i.e., $U_{t-1}(i)$ and $U_{t+1}(i)$), as well as the transition matrix.

Algorithm 1 Global-learning($G_1, G_2, \cdots, G_T$)

Input: graph matrix $G_1, G_2, \cdots, G_T$.
Output: U_t ($1 \leq t \leq T$), A and B.

1: Initialize U_t, B and A
2: **while** Not Convergent **do**
3: **for** $t = 1$ to T **do**
4: update U_t according to Eq. 7
5: update B according to Eq. 8
6: update A according to Eq. 9

In the following, we briefly discuss the time complexity and convergence of global learning algorithm. From Lemma 1 In each iteration, the computation is dominated by matrix multiplication operations. Therefore, the worst case time complexity per iteration is dominated by $O(T(nk^2 + n^2k))$. In practice , we opt to choose a low-rank latent space representation, where k is a small number (e.g., 20). In terms of convergence, followed the proof shown in previous works [4,16,32], we can prove that Algorithm 1 converges into a local minimal and the objective value is non-increasing in each iteration.

5.2 Incremental learning algorithm

The intuition behind our incremental algorithm is based on the observation that each time when we make a prediction for the next five minutes, the ground truth reading will be available immediately after five minutes. This motivates us to adjust the latent position of each vertex so that the prediction is closer to the ground truth. On the other hand, it is not necessary to perform the latent position adjustment for each vertex. This is because during a short time interval, the overall traffic condition of the whole network tends to stay steady, and the travel cost of most edges changes at a slow pace, although certain vertices can go through obvious variations. Therefore, instead of recomputing the latent positions of all the vertices from scratch at every time stamp, we perform a "lazy" update. In particular, to learn the latent space U_t, the incremental algorithm utilizes the latent space we have already learned in the previous snapshot (i.e., U_{t-1}), makes predictions for the next snapshot (i.e., G_t), and then conditionally adjusts latent attributes of a subset of vertices based on the changes of traffic condition.

5.2.1 Framework of incremental algorithm

Algorithm 2 presents the pseudo-code of incremental learning algorithm. Initially, we learn the latent space of U_1 from our global multiplicative algorithm (Line 1). With the learned latent matrix U_{t-1}, at each time stamp t between 2 and T, our incremental update consists of the following two components: 1) identify candidate vertices based on feedbacks (Lines 3-8); 2) update their latent attributes and propagate the adjustment from one vertex to its neighbors (Line 9). As outlined in Algorithm 2, given U_{t-1} and G_t, we first make an estimation of $\widehat{G_t}$ based on U_{t-1} (Line 3). Subsequently, we use G_t as the feedback information, select the set of vertices where we make inaccurate predictions, and insert them into a candidate set $cand$ (Lines 4-8). Consequently, we update U_t based on the learned latent matrix U_{t-1}, the ground truth observation G_t and candidate set $cand$ (Line 9). After that, we learn the global transition matrix A (Line 10).

5.2.2 Topology-aware incremental update

Given U_{t-1} and G_t, we now explain how to calculate U_t incrementally from U_{t-1} with the candidate set $cand$, with which we can accurately approximate G_t. The main idea is similar to an online learning process. At each round, the algorithm predicts an outcome for the required task (i.e., predict the speed of edges). Once the algorithm makes a prediction, it receives feedback indicating the correct outcome. Then, the online algorithm can modify its prediction mechanism for better predictions on subsequent timestamps. In our scenario, we first use the latent attribute matrix U_{t-1} to predict G_t as if we do not know the observation, subsequently we adjust the model of U_t according to the true observation of G_t we already have in hand.

However, in our problem, we are making predictions for the entire road network, not for a single edge. When we predict for one edge, we only need to adjust the latent attributes of two vertices, whereas in our scenario we need to update the latent attributes for many correlated vertices. Therefore, the effect of adjusting the latent attribute of one vertex can potentially affect its neighboring vertices, and influence the convergence speed of incremental learning. Hence, the adjustment order of vertices is very important.

Algorithm 2 Incremental-Learning($G_1, G_2, \cdots, G_T$)

Input: graph matrix $G_1, G_2, \cdots, G_T$.
Output: U_t ($1 \leq t \leq T$), A and B.

1: $(U_1, B) \leftarrow$Global-learning(G_1)
2: **for** $t = 2$ to T **do**
3: $\widehat{G_t} \leftarrow U_{t-1}BU_{t-1}^T$ (prediction)
4: $cand \leftarrow \emptyset$ (a subset of vertices to be updated)
5: **for** each $i \in G$ **do**
6: **for** each $j \in out(i)$ **do**
7: **if** $|G_t(i,j) - \widehat{G_t(i,j)}| \geq \delta$ **then**
8: $cand \leftarrow cand \cup \{i, j\}$
9: $U_t \leftarrow$ Incremental-Update($U_{t-1}, G_t, cand$) (See Section 5.2.2)
10: Iteratively learn transition matrix A using Eq. 9 until A converges

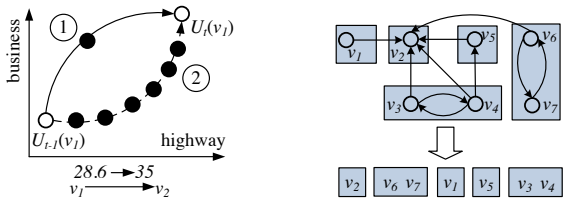
Algorithm 3 Incremental-Update($U_{t-1}, G_t, cand$)

Input: the latent matrix U_{t-1}, observed graph reading G_t, candidate set $cand$, hyper-parameters δ and τ
Output: Updated latent space U_t.

1: $U_t \leftarrow U_{t-1}$
2: **while** Not Convergent AND $cand \notin \emptyset$ **do**
3: order $cand$ from the reverse topological order
4: **for** $i \in cand$ **do**
5: $oldu \leftarrow U_t(i)$
6: **for** each $j \in out(i)$ **do**
7: adjust $U_t(i)$ with Eq. 11
8: **if** $||U_t(i) - oldu||_F^2 \leq \tau$ **then**
9: $cand \leftarrow cand \setminus \{i\}$
10: **for** each $j \in out(i)$ **do**
11: $p \leftarrow U_t(i)BU_t(j)$
12: **if** $|p - G_t(i,j)| \geq \delta$ **then**
13: $cand \leftarrow cand \cup \{j\}$

Algorithm 3 presents the details of updating U_t incrementally from U_{t-1}. For each vertex i of $cand$, we adjust its latent position so that we could make more accurate predictions (Line 7) and then examine how this adjustment would influence the candidate task set from the following two aspects: (1) if the latent attribute of i does not change much, we remove it from the set of $cand$ (Lines 8-9); (2) if the adjustment of i also affects its neighbor j, we add vertex j to $cand$ (Lines 10-13).

The remaining questions in our Incremental-Update algorithm are how to adjust the latent position of one vertex according to feedbacks, and how to decide the order of update. In the following, we address each of them.



(a) Adjustment method (b) Adjustment order

Figure 3: Challenges of adjusting the latent attribute with feedbacks.

Adjusting latent attribute of one vertex. To achieve high efficiency of adjusting the latent attribute, we propose to make the smallest changes of the latent space (as fast as possible) to predict the correct value. For example, as shown in Figure 3 (a), suppose we already know the new latent position of v_1, then fewer step movement (Option 1) is preferable than gradual adjustment (Option 2). Note that in our problem, when we move the latent position of a vertex to a new position, the objective of this movement is to produce a correct prediction for each of its outgoing edges. Specifically, given $U_{t-1}(i)$, we want to find $U_t(i)$ which could accurately predict the weight of each edge $e(v_i, v_j)$ that is adjacent to vertex v_i. We thus formulate our problem as follows:

$$\begin{aligned} U_t(i), \xi^* = \arg \min_{U(i) \in R_+^k} \frac{1}{2}||U(i) - U_{t-1}(i)||_F^2 + C\xi \\ \text{s.t.} \quad |U(i)BU^T(j) - G_t(i,j)| \leq \delta + \xi, \end{aligned} \tag{10}$$

where ξ is a non-negative slack variable, $C > 0$ is a parameter which controls the trade-off between being conservative (do not change the model too much) and corrective (satisfy the constraint), and δ is a precision parameter.

Note that we have non-negativity constraint over the latent space of $U_t(i)$. We thus adopt the approaches from [3]: When the predicted value $\widehat{y_t}$ (i.e., $U_t(i)BU_t^T(j)$) is less than the correct value y_t (i.e., $G_t(i,j)$), we use the traditional online passive-aggressive algorithm [7] because it guarantees the non-negativity of $U(i)$; Otherwise, we update $U(i)$ by solving a quadratic optimization problem. The detailed solution is as follows:

$$U_t(i) = \max(U_{t-1}(i) + (k^* - \theta^*) \cdot BU_{t-1}(j)^T, 0) \tag{11}$$

k^* and θ^* are computed as follows:

$$\begin{cases} k^* = \alpha_t, \theta^* = 0 & \text{if } \widehat{y_t} < y_t \\ k^* = 0, \theta^* = C & \text{if } \widehat{y_t} > y_t \text{ and } f(C) \geq 0 \\ k^* = 0, \theta^* = f^{-1}(0) & \text{if } \widehat{y_t} > y_t \text{ and } f(C) < 0 \end{cases} \tag{12}$$

where

$$\alpha_t = \min\left(C, \frac{\max(|\widehat{y_t} - y_t| - \delta, 0)}{||BU_{t-1}(j)^T||^2}\right)$$

$$f_t(\theta) = \max\left(U_t(i) - \theta BU_t(j)^T, 0\right) \cdot BU_t(j)^T - G_t(i,j) - \delta$$

Updating order of $cand$. As we already discussed, the update order is important because it influences the convergence speed of our incremental algorithm. Take the example of the road network shown in Figure 1, suppose our initial $cand$ contains three vertices v_7, v_6 and v_2, where we have two edges $e(v_7, v_6)$ and $e(v_6, v_2)$. If we randomly choose the update sequence as $< v_7, v_6, v_2 >$, that is, we first adjust the latent attribute of v_7 so that $c(v_7, v_6)$ has a correct reading; subsequently we adjust the latent attribute of v_6 to correct our estimation of $c(v_6, v_2)$. Unfortunately,the adjustment of v_6 could influence the correction we have already made to v_7, thus leading to an inaccurate estimation of $c(v_7, v_6)$ again. A desirable order is to update vertex v_6 before updating v_7.

Therefore, we propose to consider the reverse topology of road network when we update the latent position of each candidate vertex $v \in cand$. The general principle is that: given edge $e(v_i, v_j)$, the update of vertex v_i should be proceeded after the update of v_j, because the position of v_i is dependent on v_j. This motivates us to derive a reverse topological order in the graph of G. Unfortunately, the road network G is not a Directed Acyclic Graph (DAG), and contains cycles. To address this issue, we first generate a condensed super graph where we contract each Strongly Connected Component (SCC) of the graph G as a super node. We then derive a reverse topological order based on this condensed graph. For the vertex order in each SCC, we generate an ordering of vertices inside each SCC by random algorithms or some heuristics. Figure 3(b) shows an example of ordering for the road network of Figure 1, where each rectangle represents a SCC. After generating a reverse topological order based on the contracted graph and randomly ordering the vertices within each SCC, we obtain one final ordering $< v_2, v_6, v_7, v_1, v_5, v_4, v_3 >$. Each time when we update the latent attributes of $cand$, we follow this ordering of vertices.

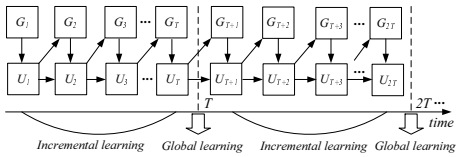


Figure 4: A batch window framework for real-time forecasting.

Time complexity. For each vertex i, the computational complexity of adjusting its latent attributes using Eq. 11 is $O(k)$, where k is number of attributes. Therefore, to compute latent attributes u, the time complexity per iteration is $O(kT(\Delta n + \Delta m))$, where Δn is number of candidate vertex in $cand$, and Δm is total number of edges incident to vertices in $cand$. In practice, $\Delta n \ll n$ and $\Delta m \ll m \ll n^2$. In addition, the SCC can be generated in linear time O(m+n) via Tarjan's algorithm [23]. Therefore, we conclude that the computational cost per iteration is significantly reduced using Algorithm 2 as compared to using the global learning approach.

5.3 Real-time forecasting

In this section, we discuss how to apply our learning algorithms to real-time traffic prediction, where the sensor reading is received in a streaming fashion. In practice, if we want to make a prediction for the current traffic, we cannot afford to apply our global learning algorithm to all the previous snapshots because it is computationally expensive. Moreover, it is not always true that more snapshots would yield a better prediction performance. The alternative method is to treat each snapshot independently: i.e., each time we only apply our incremental learning algorithm for the most recent snapshot, and then use the learned latent attribute to predict the traffic condition. Obviously, this might yield poor prediction quality as it totally ignores the temporal transitions.

To achieve a good trade-off between the above two methods, we propose to adapt a sliding window setting for the learning of our LSM-RN model, where we apply incremental algorithm at each timestamp during one time window, and only run our global learning algorithm at the end of one time window. As shown in Figure 4, we apply our global learning at timestamps T (i.e., the end of one time window), which learns the time-dependent latent attributes for the previous T timestamps. Subsequently, for each timestamp $T+i$ between [T, 2T], we apply our incremental algorithm to adjust the latent attribute and make further predictions: i.e., we use U_{T+i} to predict the traffic of $G_{T+(i+1)}$. Each time we receive the true observation of $G_{T+(i+1)}$, we calculate $U_{T+(i+1)}$ via the incremental update from Algorithm 3. The latent attributes U_{2T} will be recomputed at timestamp $2T$ (the end of one time window), and the U_{2T} would be used for the next time window $[2T, 3T]$.

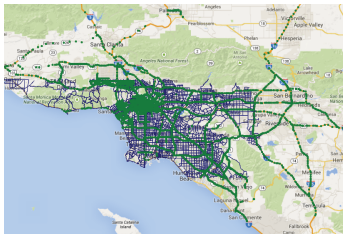


Figure 5: Sensor distribution and Los Angeles road network.

6. EXPERIMENT

6.1 Dataset

We used a large-scale high resolution (both spatial and temporal) traffic sensor (loop detector) dataset collected from Los Angeles county highways and arterial streets. This dataset includes both inventory and real-time data for 15000 traffic sensors covering approximately 3420 miles. The sampling rate of the data, which provides speed, volume (number of cars passing from sensor locations) and occupancy, is 1 reading/sensor/min. We have been collecting and archiving this sensor dataset continuously since 2010.

We chose sensor data between March and April in 2014 for our experiments, which include more than 60 million records of readings. As for the road network, we used Los Angeles road network which was obtained from HERE Map dataset [11]. We constructed two subgraphs of Los Angeles road network, termed as SMALL and LARGE. The SMALL (resp. LARGE) network contains 5984 (resp. 8242) vertices and 12538 (resp. 19986) edges. As described in Section 3, the sensor data are mapped to the road network, where 1642 (resp. 4048) sensors are mapped to SMALL (resp. LARGE). Figure 5 shows sensors locations and road network segments, where the green lines depict the sensors, and blue lines represent the road network segments. After mapping the sensor data, we have two months of network snapshots for both SMALL and LARGE.

6.2 Experimental setting

6.2.1 *Algorithms*

Our methods are termed as **LSM-RN-All** (i.e., global learning algorithm) and **LSM-RN-Inc** (i.e., incremental learning algorithm).

For edge traffic prediction, we compare with LSM-RN-Naive, where we adapted the formulations from LSM-SN ([29] and [21]) by simply combining the topology and temporal correlations. In addition, LSM-RN-Naive uses a Naive incremental learning strategy in [21], which independently learns the latent attributes of each timestamp first, then the transition matrix. We also compare our algorithms with two representative time series prediction methods: a linear model (i.e., ARIMA [18]) and a non-linear model (i.e., SVR [20]). We train each model independently for each time series with historical data. In addition, because these methods will be affected negatively due to the missing values during the prediction stages (i.e, some of the input readings for ARIMA and SVR could be zero), for fair comparison we consider ARIMA-Sp and SVR-Sp, which use the completed readings from our global learning algorithm. We also implemented the Tensor method [1, 2], however, it cannot address the sparsity problem of our dataset and thus produce meaningless results (most of the prediction values are close to 0).

For missing-value completion, we compare our algorithms with two methods: (1) KNN [10], which uses the average values of the nearby edges in Euclidean distance as the imputed value, (2) LSM-RN-Naive, which independently learns the latent attributes of each snapshot, then uses them to approximate the edge readings.

To evaluate the performance of online prediction, we consider the scenario of a batch-window setting described in Section 5.3.

Table 2: Experiment parameters

Parameters	Value range
T	$2, 4, 6, 8, \mathbf{10}, 12$
$span$	$\mathbf{5}, 10, 15, 20, 25, 30$
k	$5, 10, 15, \mathbf{20}, 25, 30$
λ	$2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, \mathbf{2^{3}}, 2^{5}$
γ	$2^{-7}, \mathbf{2^{-5}}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}$

Considering a time window $[0, 2T]$, we first batch learn the latent attributes of U_T and transition matrix A from $[0, T]$, we then sequentially predict the traffic condition for the timestamps during $[T + 1, 2T]$. Each time when we make a prediction, we receive the true observations as the feedback. We compare our Incremental algorithm (Inc), with three baseline algorithms: Old, LSM-RN-Naive and LSM-RN-All. Specifically, to predict G_{T+i}, LSM-RN-Inc utilizes the feedback of $G_{T+(i-1)}$ to adjust the time-dependent latent attributes of $U_{T+(i-1)}$, whereas Old does not consider the feedback, and always uses latent attributes U_T and transition matrix A from the previous time window. On the other hand, LSM-RN-Naive ignores the previous snapshots, and only applies the inference algorithm to the most recent snapshot $G_{T+(i-1)}$ (aka Mini-batch). Finally, LSM-RN-All applies the global learning algorithm consistently to all historical snapshots (i.e., G_1 to $G_{T+(i-1)}$) and then makes a prediction (aka Full-batch).

6.2.2 *Configurations and measures.*

We selected two different time ranges that represent rush hour (i.e., 7am-8am) and non-rush hour (i.e., 2pm-3pm), respectively. For the task of missing value completion, during each timestamps of one time range (e.g., rush hour), we randomly selected 20% of values as unobserved and manipulated them as missing [1], with the objective of completing those missing values. For each traffic prediction task at one particular timestamp (e.g., 7:30 am), we randomly selected 20% of the values as unknown and use them as ground-truth values.

We varied the parameters T and $span$: where T is the number of snapshots, and $span$ is time gap between two continuous snapshots. We also varied k, λ, and γ, which are parameters of our model. The default settings (shown with bold font) of the experiment parameter are listed in Table 2. Because of space limitations, the results of varying γ are not reported, which are similar to result of varying λ. We use Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE) to measure the accuracy. In the following we only report the experiment results based on MAPE, the experiment results based on RMSE are reported in the technical report [8]. Specifically, MAPE is defined as follows:

$$MAPE = (\frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y_i}|}{y_i})$$

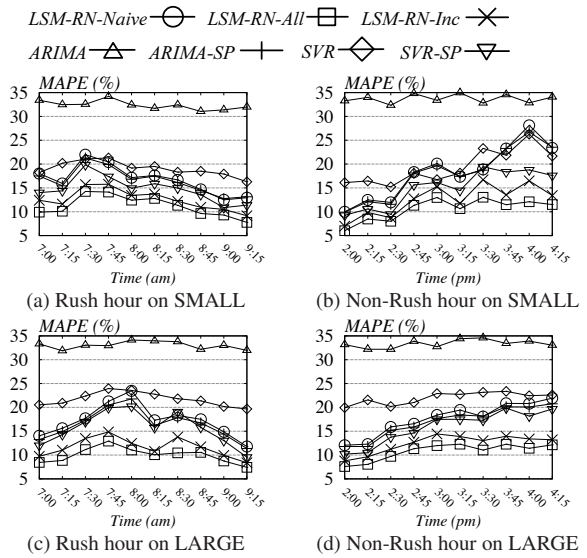
With ARIMA and SVR, we use the dataset of March to train a model for each edge, and use 5-fold cross-validation to choose the best parameters. All the tasks of missing value completion and edge traffic prediction tasks are conducted on April data. We conducted our experiments with C++ on a Linux PC with i5-2400 CPU @ 3.10G HZ and 24GB memory.

6.3 Comparison with edge traffic prediction

6.3.1 *One-step ahead prediction*

The experimental results of SMALL are shown in Figures 6 (a) and (b). Among all the methods, LSM-RN-All and LSM-RN-Inc achieve the best results, and LSM-RN-All performs slightly better

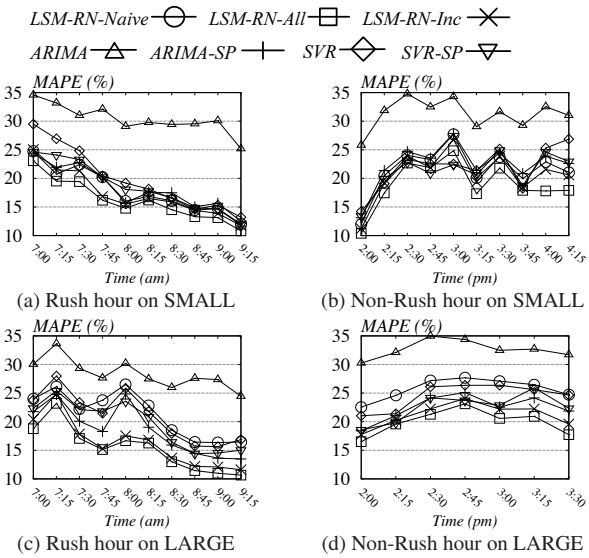
[1]Note that missing values are plenty in our dataset, especially for arterials. However, we needed ground-truth for evaluation purposes and that is why we generated missing values artificially.



(a) Rush hour on SMALL (b) Non-Rush hour on SMALL

(c) Rush hour on LARGE (d) Non-Rush hour on LARGE

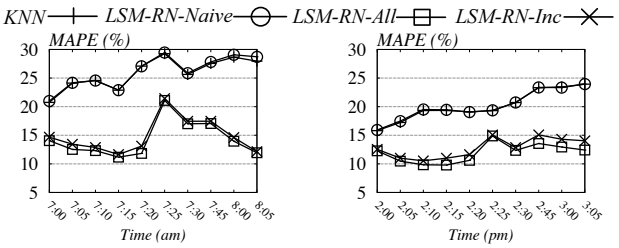
Figure 6: One-step ahead prediction MAPE



(a) Rush hour on SMALL (b) Non-Rush hour on SMALL

(c) Rush hour on LARGE (d) Non-Rush hour on LARGE

Figure 7: Six-steps ahead prediction MAPE



(a) Rush hour on LARGE (b) Non-Rush hour on LARGE

Figure 8: Missing value completion MAPE

than LSM-RN-Inc. This demonstrates the effectiveness of time-dependent latent attributes and the transition matrix. We observe that without imputing of missing values, time series prediction techniques (i.e., ARIMA and SVR) perform much worse than LSM-RN-All and LSM-RN-Inc. Meanwhile, LSM-RN-Naive, which separately learns the latent attributes of each snapshot, cannot achieve good prediction results as compared to LSM-RN-All and LSM-RN-Inc. This indicates that simply combining topology and time is not enough for accurate predictions. We note that even with completed readings, the accuracy of SVR-Sp and ARIMA-Sp is worse than that of LSM-RN-All and LSM-RN-Inc. One reason is that simply combining the spatial and temporal properties does not necessarily yield a better performance. Another reason is that both SVR-Sp and ARIMA-Sp also suffer from missing data during the training stage, which results in less accurate predictions. In the technical report [8], we show how the ratio of missing data would influence the prediction performance. Finally, we observe that SVR is more robust than ARIMA when encountering missing values: i.e., ARIMA-Sp performs significantly better than ARIMA, while the improvement of SVR-Sp over SVR is marginal. This is because ARIMA is a linear model which mainly uses the weighted average of the previous readings for prediction, while SVR is a non-linear model that utilizes a kernel function. Figures 6 (c) and (d) show the experiment results on LARGE, the trend is similar to SMALL.

6.3.2 Multi-steps ahead prediction

We now present the experiment results on long-term predictions, with which we predict the traffic conditions for the next 30 minutes (i.e., $h = 6$). The prediction accuracy of different methods on SMALL are shown in Figures 7 (a) and (b). Although LSM-RN-All and LSM-RN-Inc still outperform other methods, the margin between our methods and the baselines is narrower. The reason is that: when we make long-term predictions, we use the predicted values from the past for future prediction. This leads to the problem of error accumulation, i.e., errors incurred in the past are propagated into future predictions. We observe the similar trends on LARGE, the results are reported in Figures 7 (c) and (d).

6.4 Comparison for missing value completion

In this set of experiments, we evaluate the completion accuracy of different methods. Due to space limitation, we only report the experiment results on LARGE in Figures 8 (a) and (b), and the effects on SMALL are similar. We observe that both LSM-RN-All and LSM-RN-Inc achieve much lower errors than that of other methods. This is because LSM-RN-All and LSM-RN-Inc capture both spatial and temporal relationships, while LSM-RN-Naive and KNN only use spatial property. LSM-RN-All performs better than LSM-RN-Inc by jointly inferring all the latent attributes. On the other hand, we note that LSM-RN-Naive and KNN have similar performances, which is inferior to our methods. This also indicates that utilizing both spatial and temporal properties yields a larger gain than only utilizing the spatial property. As shown in Figure 8(b), the completion performance during the non-rush hour is better as compared to the rush hour time. This is because during rush hour range, the traffic condition is more dynamic, and the underlying pattern and transition changes frequently.

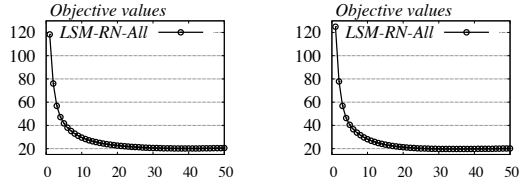
6.5 Scalability

Table 3: Running time comparisons. For ARIMA and SVR, the training time cost is the total training time for all the edges for one-step ahead prediction, and the prediction time is the average prediction time per edge per query.

data	SMALL		LARGE	
	train (s)	pred.(ms)	train (s)	pred. (ms)
LSM-RN-Naive	-	1353	-	29439
LSM-RN-All	-	869	-	14247
LSM-RN-Inc	-	407	-	4145
ARIMA	484	0.00015	987	0.00024
SVR	47420	0.00042	86093.99	0.00051

Table 3 shows the running time of different methods. Although ARIMA and SVR are fast in each prediction, they require large volume of training data and have much higher training time, which can be a problem for real systems. On the contrary, our methods do not require extra training data, i.e., our methods efficiently train and predict at the same time. Among them, LSM-RN-Inc is the most

efficient approach: it only takes less than 500 milliseconds to learn the time-dependent latent attributes and make predictions for all the edges of the road network. This is because our incremental learning algorithm conditionally adjusts the latent attributes of certain vertices, and utilizes the topological order that enables fast convergence. Even for the LARGE dataset, LSM-RN-Inc takes less than five seconds, which is acceptable considering that the span between two snapshots is at least five minutes in practice. This demonstrates that LSM-RN-Inc scales well to large road networks. Regarding LSM-RN-All and LSM-RN-Naive, they both require much longer running time than that of LSM-RN-Inc. In addition, LSM-RN-All is faster than LSM-RN-Naive. This is because LSM-RN-Naive independently runs the global learning algorithm for each snapshot T times, while LSM-RN-All only applies global learning for all the snapshots once.



(a) SMALL (b) LARGE

Figure 9: Converge rate

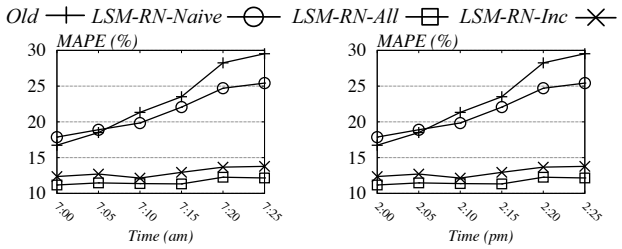
Convergence analysis. Figures 9 (a) and (b) report the convergence rate of iterative algorithm LSM-RN-All on both SMALL and LARGE. As shown in Figure 9, LSM-RN-All converges very fast: when the number of iterations is around 20, our algorithm tends to converge in terms of our objective value in Eq. 4.

6.6 Comparison for real-time forecasting

In this set of experiments, we evaluate our online setting algorithms. Due to space limitation, we only report the experiment results on LARGE. As shown in Figures 10 (a) and (b), LSM-RN-Inc achieves comparable accuracy with LSM-RN-All (Full-batch). This is because LSM-RN-Inc effectively leverages the real-time feedback to adjust the latent attributes. We observe that LSM-RN-Inc performs much better than Old and LSM-RN-Naive (Mini-batch), which ignore either the feedback information (i.e., Old) or the previous snapshots (i.e., LSM-RN-Naive). One observation is that Old performs better than LSM-RN-Naive for the initial timestamps, whereas Old surpasses Mini-batch at the later timestamps. This indicates that the latent attributes learned in the previous time-window are more reliable for predicting the near-future traffic conditions, but may not be good for long-term predictions because of the error accumulation problem.

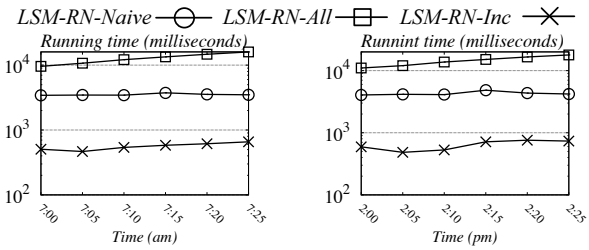
Figures 11 (a) and (b) show the running time comparisons of different methods. One important observation from this experiment is that LSM-RN-Inc is the most efficient approach, which is on average two times faster than LSM-RN-Naive and one order of magnitude faster than LSM-RN-All. This is because LSM-RN-Inc performs a conditional latent attribute update for vertices within a small portion of road network, whereas LSM-RN-Naive and LSM-RN-All both recompute the latent attributes from at least one entire road network snapshot. Since in the real-time setting, LSM-RN-All utilizes all the up-to-date snapshots and LSM-RN-Naive only considers the most recent single snapshot, LSM-RN-Naive is faster than LSM-RN-All. We observe that LSM-RN-Inc only takes less than 1 second to incorporate the real-time feedback information, while LSM-RN-Naive and LSM-RN-All take much longer.

Therefore, we conclude that LSM-RN-Inc achieves a good trade-off between prediction accuracy and efficiency, which is applicable for real-time traffic prediction applications.



(a) Rush hour on LARGE (b) Non-Rush hour on LARGE

Figure 10: Online prediction MAPE



(a) Rush hour on LARGE (b) Non-Rush hour on LARGE

Figure 11: Online Prediction time

6.7 Varying parameters of our methods

In this section, we evaluate the performance of our methods by varying the parameters of our model. Due to space limitation, we only show the experimental results on SMALL.

6.7.1 Effect of varying T

Figure 12 (a) and Figure 12 (b) show the prediction performance and the running time of varying T, respectively. We observe that with more snapshots, the prediction error decreases. In particular, when we increase T from 2 to 6, the results improve significantly. However, the performance tends to stay stable at $T \geq 6$. This indicates that fewer snapshots (i.e., two or less) are not enough to capture the traffic patterns and the evolving changes. On the other hand, more snapshots (i.e., more historical data) do not necessarily yield better gain, considering the running time increases when we have more snapshots. Therefore, to achieve a good trade-off between running time and prediction accuracy, we suggest to use at least 6 snapshots, but no more than 12 snapshots.

6.7.2 Effect of varying span

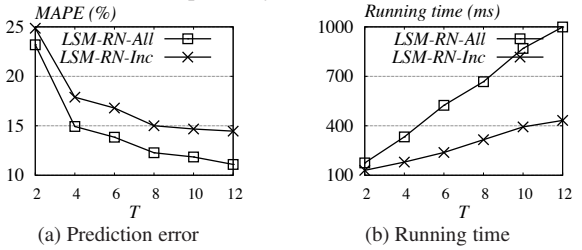
The results of varying $span$ are shown in Figure 13. Clearly, as the time gap between two snapshots increases, the performance declines. This is because when $span$ increases, the evolving process of underlying traffic may not evolve smoothly, the transition process learned in the previous snapshot is not applicable for the future. Fortunately our sensor dataset usually have high-resolution, so it is better to use smaller span to learn the latent attributes. In addition, span does not affect the running time of either algorithms.

6.7.3 Effect of varying k and λ

Figure 14 (a) shows the effect of varying k. We observe that: (1) we achieve better results with increasing number of latent attributes; (2) the performance is stable when $k \geq 20$. This indicates that a low-rank latent space representation can already capture the attributes of the traffic data. In addition, our results show that when the number of latent attributes is small (i.e., $k \leq 30$), the running time increases with k but does not change much when we vary k from 5 to 30. Therefore, setting k to 20 achieves a good balance between computational cost and accuracy.

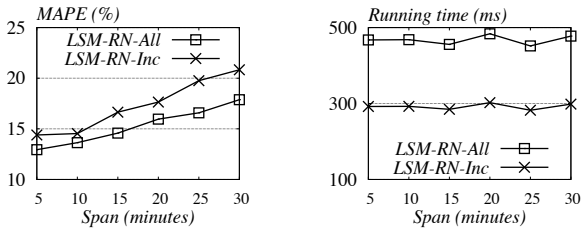
Figure 14 (b) depicts the effect of varying λ, which is the regularization parameter for our graph Laplacian dynamics. We observe that the graph Laplacian has a larger impact on LSM-RN-All al-

gorithm than on LSM-RN-Inc. This is because λ controls how the global structure similarity contributes to latent attributes and LSM-RN-All jointly learns those time-dependent latent attribute, thus λ has larger effect on LSM-RN-All. In contrast, LSM-RN-Inc adaptively updates the latent positions of a small number of changed vertices in limited localized view, and thus is less sensitive to the global structure similarity than LSM-RN-All. In terms of parameters choices, $\lambda = 2$ and $\lambda = 8$ yields best results for LSM-RN-All and LSM-RN-Inc, respectively.



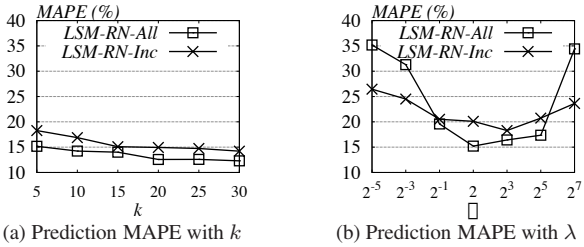
(a) Prediction error (b) Running time

Figure 12: Effect of varying T



(a) Prediction error (b) Running time

Figure 13: Effect of varying span



(a) Prediction MAPE with k (b) Prediction MAPE with λ

Figure 14: Effect of varying k and λ, where k is number of latent attributes, and λ is the graph regularization parameter.

7. CONCLUSION

In this paper, we studied the problem of real-time traffic prediction using real-world sensor data for road networks. We proposed LSM-RN, where each vertex is associated with a set of latent attributes that captures both topological and temporal properties of road networks. We showed that the latent space modeling of road networks with time-dependent weights accurately estimates the traffic patterns and their evolution over time. To efficiently infer these time-dependent latent attributes, we developed an incremental online learning algorithm which enables real-time traffic prediction for large road networks. With extensive experiments we verified the effectiveness, flexibility and scalability of our model in identifying traffic patterns and predicting future traffic conditions.

Acknowledgments. Dingxiong Deng, Cyrus Shahabi and Ugur Demiryurek were supported in part by NSF grants IIS-1115153, IIS-1320149, CNS-1461963 and Caltrans-65A0533, the USC Integrated Media Systems Center (IMSC), and unrestricted cash gifts from Google, Northrop Grumman, Microsoft, and Oracle. Linhong Zhu was supported in part by DARPA grant Number W911NF-12-1-0034. Rose Yu and Yan Liu were supported by the U. S. Army Research Office under grant Number W911NF-15-1-0491, NSF IIS-1254206 and the USC Integrated Media System Center (IMSC). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the sponsors such as NSF.

8. REFERENCES

[1] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, March 2011.

[2] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015.

[3] M. Blondel, Y. Kubo, and U. Naonori. Online passive-aggressive algorithms for non-negative matrix factorization and completion. In *AISTATS*, 2014.

[4] D. Cai, X. He, J. Han, and T. S. Huang. Graph regularized nonnegative matrix factorization for data representation. *TPAMI*, 33(8):1548–1560, 2011.

[5] H. Cheng and P.-N. Tan. Semi-supervised learning with data calibration for long-term time series forecasting. In *KDD*, pages 133–141. ACM, 2008.

[6] F. C. T. Chua, R. J. Oentaryo, and E.-P. Lim. Modeling temporal adoptions using dynamic matrix factorization. In *ICDM*, pages 91–100. IEEE, 2013.

[7] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *JMLR*, 7:551–585, 2006.

[8] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, R. Yu, and Y. Liu. Latent space model for road networks to predict time-varying traffic. *CoRR*, abs/1602.04301, 2016.

[9] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *KDD*, pages 126–135. ACM, 2006.

[10] J. Haworth and T. Cheng. Non-parametric regression for space–time forecasting under missing data. *Computers, Environment and Urban Systems*, 36(6):538–550, 2012.

[11] HERE. https://company.here.com/here/.

[12] T. Idé and M. Sugiyama. Trajectory regression on road networks. In *AAAI*, 2011.

[13] H. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94, 2014.

[14] J. Kwon and K. Murphy. Modeling freeway traffic with coupled hmms. Technical report.

[15] R. Lambiotte, J.-C. Delvenne, and M. Barahona. Laplacian dynamics and multiscale modular structure in networks. *arXiv:0812.1770*, 2008.

[16] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2001.

[17] A. K. Menon and C. Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2011.

[18] B. Pan, U. Demiryurek, and C. Shahabi. Utilizing real-world transportation data for accurate traffic prediction. ICDM, pages 595–604, 2012.

[19] L. Qu, Y. Zhang, J. Hu, L. Jia, and L. Li. A bpca based missing value imputing method for traffic flow volume data. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 985–990. IEEE, 2008.

[20] G. Ristanoski, W. Liu, and J. Bailey. Time series forecasting using distribution enhanced linear regression. In *PAKDD*, pages 484–495. 2013.

[21] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson. Modeling dynamic behavior in large evolving graphs. In *WSDM*, pages 667–676, 2013.

[22] A. Saha and V. Sindhwani. Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization. In *WSDM*, pages 693–702. ACM, 2012.

[23] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[24] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, 22(3):493–521, 2011.

[25] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories. In *KDD*, pages 25–34. ACM, 2014.

[26] L. Wu, X. Xiao, D. Deng, G. Cong, A. D. Zhu, and S. Zhou. Shortest path and distance queries on road networks: An experimental evaluation. *VLDB*, 5(5):406–417, 2012.

[27] J. Xu, D. Deng, U. Demiryurek, C. Shahabi, and M. v. d. Schaar. Mining the situation: Spatiotemporal traffic prediction with big data. *Selected Topics in Signal Processing, IEEE Journal of*, 9(4):702–715, 2015.

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

[29] Y. Zhang and D.-Y. Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *KDD*, pages 606–614. ACM, 2012.

[30] J. Zheng and L. M. Ni. Time-dependent trajectory regression on road networks via multi-task learning. In *AAAI*, 2013.

[31] J. Zhou and A. K. Tung. Smiler: A semi-lazy time series prediction system for sensors. SIGMOD '15, pages 1871–1886, 2015.

[32] L. Zhu, A. Galstyan, J. Cheng, and K. Lerman. Tripartite graph clustering for dynamic sentiment analysis on social media. In *SIGMOD'14*, pages 1531–1542.

[33] L. Zhu, G. V. Steeg, and A. Galstyan. Scalable link prediction in dynamic networks via non-negative matrix factorization. *arXiv preprint arXiv:1411.3675*, 2014.