Situation Aware Multi-Task Learning for Traffic Prediction

Dingxiong Deng[1], Cyrus Shahabi[1], Ugur Demiryurek[1], Linhong Zhu[2]
Department of Computer Science, University of Southern California[1]
Pinterest[2]
{dingxiod, shahabi, demiryur}@usc.edu, linhongzhu@pinterest.com

***Abstract*—Due to the recent vast availability of transportation traffic data, major research efforts have been devoted to traffic prediction, which is useful in many applications such as urban planning, traffic management and navigations systems. Current prediction methods that independently train a model per traffic sensor cannot accurately predict traffic in every situation (e.g., rush hours, constructions and accidents) because there may not exist sufficient training samples per sensor for all situations. To address this shortcoming, our core idea is to explore the commonalities of prediction tasks across multiple sensors who behave similarly in a specific traffic situation. Instead of building a model independently per sensor, we propose a Multi-Task Learning (MTL) framework that aims to first automatically identify the traffic situations and then simultaneously build one forecasting model for similar-behaving sensors per traffic situation. The key innovation here is that instead of the straightforward application of MTL where each "task" corresponds to a sensor, we relate each MTL's "task" to a traffic situation. Specifically, we first identify these traffic situations by running clustering algorithms on all sensors' data. Subsequently, to enforce the commonalities under each identified situation, we use the group Lasso regularization in MTL to select a common set of features for the prediction tasks, and we adapt efficient FISTA algorithm with guaranteed convergence rate. We evaluated our methods with a large volume of real-world traffic sensor data; our results show that by incorporating traffic situations, our proposed MTL framework performs consistently better than naively applying MTL per sensor. Moreover, our holistic approach, under different traffic situations, outperforms all the best traffic prediction approaches for a given situation by up to 18% and 30% in short and long term predictions, respectively.**

I. Introduction

There are now numerous ways to collect traffic data. The most traditional and widely used method to collect traffic flow data is via traffic loop detectors installed on the surface of the pavements of almost all major cities in the world. These sensors continuously measure the average speed and the number of cars passed through a certain segment of the road, which can be used to compute the average speed at that road segment [1]. For example, there are approximately 15000 loop detectors installed on highways and arterial streets of Los Angeles County (covering 3420 miles, cumulatively). The more recent data collection methods convert/aggregate data from GPS devices [2], [3] and traffic cameras [4] to the same volume/occupancy format acquired by loop detectors, sometimes simply to protect the privacy of individuals (e.g., trajectories, cars). For example, Inrix (a main provider of real-time and archived traffic data) aggregates real-time traffic information from GPS-enabled vehicles, mobile devices and traditional road sensors to estimate traffic flow on road segments. Therefore, in this paper, without loss of generality, we abstract the data collection points as traffic "sensors" and the collected data as "average speed" at that collection point (i.e., road segment). Consequently, the traffic prediction problem aims to estimate the future travel speed of each and every sensor, given the historical and real-time speed readings observed from these sensors. Recently, the problem of traffic prediction has attracted much attention from both industry and academia due to its critical utility in various applications such as estimation of the arrival time (ETA), route navigation, traffic regulation, and urban planning.

During the past three decades, many approaches have been proposed to solve the traffic prediction problem including time series methods [5]–[7], probabilistic graph approaches [8], [9], and other machine learning models [10]–[13]. The majority of the existing studies (e.g., [5], [11], [12]) treat each sensor independently, i.e., they extract the training data from one sensor and then learn a prediction model per sensor. These studies hence do not take advantage of the obvious commonalities across sensors in their models. A few studies do consider finding commonalities across multiple sensors by grouping the sensors according to their spatial proximity (in Euclidean [14] or network space [7], [8]) or similarities in a latent space [13]. However, none of these studies distinguish the underlying traffic situations. Even for one single sensor, traffic prediction can become very complex given the numerous traffic conditions that can cause traffic congestion e.g., recurring (e.g., daily rush hours), occasional (e.g., rainy day), unpredictable (e.g., accidents), and temporarily congestions (e.g., a basketball game). Hence, it can be difficult for one single model to capture all these complex traffic situations (whether for a single sensor or across sensors). Few studies built different models for different traffic situations (focusing only on two situations) such as [5] using ARIMA in normal traffic conditions but Historical Average Model (HAM) for rush hours, or [10] using two different deep learning models to predict both rush hours and post-accident congestions. However, they neglect the fact that sensors behaving the same under one situation may behave differently under another traffic situation, e.g., a sensor x may be grouped with sensor y under "normal condition" but with sensor z under "rush hour". We are not aware of a holistic traffic prediction approach that

builds a model for all sensors under all traffic situations.

Towards this end, we propose to explore the commonalities across sensors. This is because the traffic readings within one sensor can be an amalgam of multiple traffic situations, and thus difficult to build one single model to capture all these traffic situations. On the other hand, our key observation from working with large-scale traffic sensor data is that there exists so much commonalities across sensors, especially since they exhibit similar patterns under the same traffic situation, for example during the rush hour or at a rainy day. Moreover, the number of traffic situations is limited, which lends itself to building one model per traffic situation rather than per sensor. To summarize, our hypothesis is that ***building models based on the shared traffic situations across sensors can help improve the prediction accuracy***.

To verify our hypothesis, we utilize the framework of Multi-Task Learning (MTL) because the objective of MTL is to explore the commonalities across different tasks and learn multiple related tasks by extracting and utilizing shared information. To examine whether traffic situations are necessary to build the prediction models, we first ignore the underlying traffic situation, and we naively apply the traditional MTL formulation (termed as **Naive-MTL**) to jointly learn the prediction model of all sensors, i.e., *each "task" corresponds to a sensor* (see Figure 1). Thus, the objective of Naive-MTL is then to build prediction models for all sensors via exploring their commonalities. Unfortunately, it is impossible to find the commonalities across sensors without considering their traffic situations, and the performance of Naive-MTL becomes similar to that of training models per sensor independently. This is because even two nearby sensors under different traffic situations can behave very differently. For example, consider two nearby sensors on northbound I-110, where one sensor is more prone to traffic incidents than the other, although they behave almost the same during normal traffic conditions. On the other hand, the accident-prone sensor at northbound I-110 may behave similar to another sensor located at northbound I-710 (a freeway parallel to I-110) when incident happens, although they may not be correlated in other traffic situations.

Therefore, we propose a Situation-Aware Multi-Task Learning (termed as **SA-MTL**) framework: where we first identify the traffic situations across all sensors, then apply the MTL framework for each identified traffic situation, i.e., **each "task" corresponds to a traffic situation** (see Figure 3). Since the number of distinct traffic situations is small, we can apply MTL for each individual traffic situation across different sensors to examine whether the traffic situations are shared among all the sensors. Specifically, to identify the traffic situation, we can augment each training sample of one sensor with additional contextual features including road type (e.g., highway or arterial), location, weather condition, area classification (e.g., business district, residential), accident information, etc. Subsequently, we combine the training samples across all sensors and cluster them into several partitions where each partition represents one typical traffic situation and consists of different number of training samples from all sensors. Consequently, for each specific traffic situation, we use MTL that simultaneously learns the prediction model of all sensors. In particular, we employ the group Lasso regularization based on the $l_{2,1}$-norm, which ensures that for one traffic situation a small set of features are shared among all sensor prediction tasks. We utilize the FISTA [15] method to solve the proposed optimization problem with guaranteed convergence rate.

We evaluated our proposed model with extensive experimental study on the large scale Los Angeles traffic sensor data. We show that by taking into consideration all of the traffic situations, our proposed SA-MTL framework performs consistently better than not only Naive-MTL but also other state-of-the-art approaches, with up to 18% and 30% improvement for short and long term predictions, respectively, over the best approach under each traffic situation and prediction horizon.

The remaining of this paper is organized as follows. We define our problem and explain Naive-MTL in Section II. We present our proposed SA-MTL in Section III. In Section IV, we report the experiment results. We discuss the related work in Section V and conclude the paper afterwards.

II. Problem Definition And Naive MTL Formulation

In this section, we first provide our problem definition, we then describe a typical method for applying Multi-Task Learning framework to our traffic prediction problem. Finally, we show that the existing MTL formulation does not perform well for our problem. For ease of presentation, the notations used throughout this paper are presented in Table I.

TABLE I
NOTATIONS AND EXPLANATIONS

Notations	Explanations
t, T	one sensor t, total number of sensors T
$v_t(\gamma)$	travel speed of sensor t at time γ
h	prediction horizon
x_t, y_t	the training input and output of sensor t
D	the feature dimension
w_t, W	the model parameter of one sensor t, model matrix of all sensors
l, Ω	the empirical loss and regularization terms
X	combined training samples of all sensors
P_c	one traffic situation partitioning

A. Problem Definition

Given a set of road segments containing T traffic sensors, assume that at a given time interval γ (e.g., 5 minute), each sensor t provides a traffic speed reading $v_t(\gamma)$ (e.g., 40 miles/hour). Given a historical sensor dataset, the objective is to predict the future traffic speed of any given sensor. We formulate the speed prediction problem as follows:

Definition 1: Given a set of observed historical reading of every sensor t, suppose the current time is γ, the traffic prediction problem is to estimate the future travel speed $v_t(\gamma + h)$ for each sensor, where h is the prediction horizon. For example, when $h = 1$, we predict the traffic speed at the next timestamp.

Definition 2: Short-term prediction and long-term prediction refer to the scenarios when $h = 1$ and $h > 1$, respectively.

In this paper, we regard the traffic prediction problem as a regression problem. In particular, for each sensor t, suppose the current time is γ, in order to predict $v_t(\gamma + h)$, we mainly extract the previous lag readings (i.e., $v_t(\gamma-1), v_t(\gamma-2), \cdots, v_t(\gamma - lag)$) as the training features (The details of other features can be found in Section IV). For each sensor t, we construct the training input x_t and output y_t, where $x_t \in R^{n_t \times D}$, $y_t \in R^{n_t}$ and D is the feature number, and we want to learn a function such that $y_t = f_t(x_t)$.

Many machine learning approaches have been utilized to solve traffic prediction problem, the majority of them treat each sensor independently and train a model per sensor. For simplicity, we consider linear prediction models[1]: i.e., for each sensor t, our goal is to infer a linear function f where $f_t(x_t) = x_t w_t$ and $w_t \in R^{D\times 1}$. To learn the parameter vector w_t, we solve the following optimization problem:

$$\arg\min_{w_t} l(f_t(x_t), y_t) + \Omega(w_t), \tag{1}$$

where $l(f_t(x_t), y_t)$ defines the loss function and $\Omega(w_t)$ is the regularization term of each sensor. For example, we can employ squared loss and l_2 regularization, respectively.

Therefore, our objective is to learn T linear models of each sensor. We denote $W = [w_1, \cdots, w_T] \in R^{D\times T}$ as the parameter matrix to be estimated.

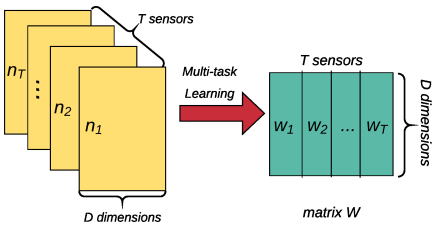


Fig. 1. Illustration of Naive Multiple-Task Learning

B. Naive Multi-Task Learning Formulation

In this paper, we are interested in exploring the commonalities across different sensors and applying them to reduce the error of prediction. Instead of solving each task independently, we first employ the typical multi-task learning formulation, where we can learn all prediction tasks simultaneously by extracting and utilizing the common information across these sensors. As shown in Figure 1, one typical formulation of MTL that estimates W is as follows:

$$\arg\min_{W} L(W) + \Omega(W) \tag{2}$$

where $L(W) = \sum_{t=1}^{n} \frac{1}{n_t} l(f_t(x_t), y_t)$ and $\Omega(W)$ encodes our prior knowledge to constrain the sharing of information between sensors.

Many different MTL approaches have been proposed to enforce the commonalities across tasks by defining the appropriate penalty functions Ω. For example in [16], each w_t is constrained to be close to each other; and in [17], a common set of features are selected across the tasks by employing the $l_{2,1}$ norm of W matrix.

[1]Our method can be generalized to non-linear models.

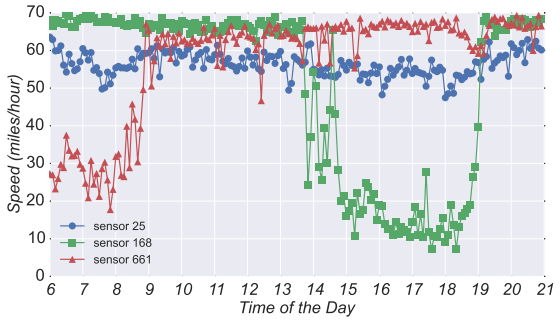


Fig. 2. Traffic readings of three different highway sensors at the same day.

However, simply applying the existing MTL framework does not perform well for our traffic prediction problem. The reason is that the evolving patterns of sensors can be a combination of different traffic situations, and the prior knowledge we have enforced in MTL no longer hold any more. We use the following example to illustrate the intuition:

Example 1: *Figure 2 shows one day traffic readings of three highway sensors. Obviously the three sensors exhibit different morning/afternoon rush hour patterns. For instance, the speed of sensor 25 does not fluctuate too much throughout the whole day, while sensor 168 and 661 clearly have one afternoon and morning rush hour patterns, respectively. Because the prediction model for rush hour is typically different with the model for normal traffic condition and each sensor has its unique mixture of normal and rush hour traffic conditions, simply enforcing the model similarities (e.g., w_t should be close to each other) across all sensors do not perform well.*

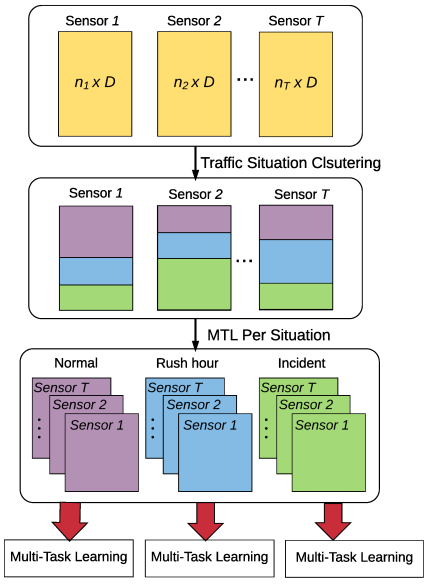


Fig. 3. Overview of SA-MTL Framework. SA-MTL first combines the training samples of all sensors together, and clusters them into several partitions, where each partition represents one typical traffic situation and consists of different number of training samples from all sensors. Consequently, for each partition (i.e., traffic situation), we utilize multi-task feature learning that simultaneously learns the prediction model of all sensors.

III. SA-MTL Framework

In Section III-A, we first describe the general process of our proposed SA-MTL framework. Subsequently we explain each component of SA-MTL in Sections III-B, III-C and III-D.

A. Overall Framework

Although each sensor has its unique traffic patterns, they exhibit similar performance under the same traffic situation (e.g., normal or rush hour condition), as shown in Figure 2. Our hypothesis is that sensor behave similarly under the common traffic situation, which motivates us to explore the commonalities among sensors under each traffic situation.

Towards this end, we propose Situation-Aware Multi-Task-Learning framework (SA-MTL), which is shown in Figure 3. With SA-MTL, we first identify the traffic situations across all sensors via clustering algorithms, then apply tradition multi-task learning framework for each traffic situation. The three steps are also illustrated in Algorithm 1: (1) Combine the training samples from all sensors together (Lines 1-3); (2) Cluster the combined training data into k partitions, where each partition constitutes training samples from different sensors (Lines 4); (3) Apply multi-task learning for each partition P_c, where $c \in [1, k]$, and learn a parameter matrix W_c for each traffic situation (Lines 5-7).

With the trained models W_c of each partition P_c, we are able to make the traffic prediction for any sensor under any situations. Given a prediction request from one sensor with input features, we first determine the underlying traffic situation (i.e., cluster label c) of this request. Subsequently, we can utilize the trained model from W^c to predict its future travel speed.

B. Combining and Augmenting Training Data

Our main hypothesis is that traffic situations are shared among all sensors. Therefore, to identify them, we combine the training data x_t from each sensor t into a matrix X, where $X \in R^{N \times D}$ and $N = \sum_{t=1}^{T} n_t$. This addresses the issue that one sensor may have insufficient training samples under one traffic situation.

Besides utilizing the sensor dataset, we can integrate other sources of information that relates to traffic situation, such as sensor attributes (i.e., sensor direction, category, etc), road and weather condition, and events information to generate better traffic situations. Currently, road construction and weather information are readily available for most areas of United States; and other factors such as incident are becoming available at certain areas. For example, three years of Los Angeles accident data from different agencies including California Highway Patrol (CHP) and California Transportation Agencies (CalTrans) are public available. With all these extra information, we can augment the training data (i.e., adding more features) before feeding into the clustering algorithm to better identify the traffic situation.

C. Traffic Situation Clustering

With the combined matrix X, the next step is to identify the hidden traffic situations. That is, we want to partition the training data into different groups, where each group corresponds to one traffic situation and contains the training data from different sensors. Hence we consider unsupervised learning (i.e., clustering) methods.

Algorithm 1: SA-MTL Framework

Input: $\forall t \in [1, T], x_t \in R^{n_t \times D}, y_t \in R^{n_t}$, number of cluster k
Output: $\forall c \in [1, k], W^c$

1 **for** $t = 1$ *to* T **do**
2 $\quad X \leftarrow \bigcup x_t$ (See Section III-B)
3 **end**
4 Partition X into clusters P_c $(c \leq k)$ (See Section III-C)
5 **for** $c = 1$ *to* k **do**
6 $\quad$ Multi-Task Learning W_c for each traffic situation P_c (See Section III-D)
7 **end**

To identify the traffic situation, a straightforward idea is to use traditional clustering methods such as K-means clustering [18] and specify the number of cluster. In our traffic prediction problem, our objective is to discover the underlying similarities across sensors, i.e., the latent traffic patterns. In addition, since the density of different traffic situations can be skewed (e.g., normal traffic condition dominates the training sample), K-means clustering may not perform well. Therefore, we propose to utilize Non-Negative Matrix Factorization (NMF) [19] to discover the traffic situations. NMF has been shown to have the ability to extract the underlying hidden features, which has been applied to document clustering [20] and traffic predictions [13]. In [20], given a document corpus, NMF can discover the basic topics of these documents, and each document will be represented as an addictive combination of these basic topics. Subsequently, the cluster label of each document can then be determined by finding the base topic with which the document has the largest projection value. We use the similar idea to discover the hidden situations and assign each training sample into the corresponding one.

Formally, given the combined input matrix $X \in R^{N \times D}$ and the cluster number k, where $k < D$, the objective of NMF is to find two non-negative matrices $V \in R_+^{N \times k}$ and $H \in R_+^{k \times d}$ to approximate the original matrix X, the formulation is given as follows:

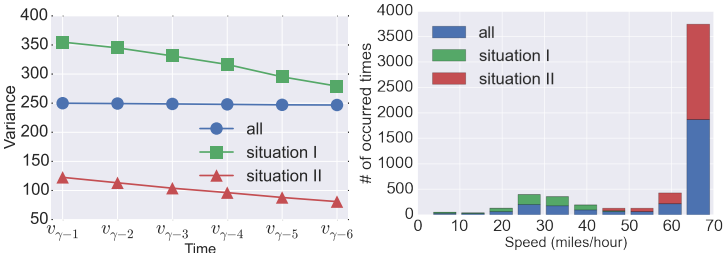
$$\arg\min_{V,H} \frac{1}{2} ||X - VH||_F^2 \tag{3}$$

By solving the above equation, we have two non-negative matrices $V \in R_+^{N \times k}$ and $H \in R_+^{k \times D}$, where each row of H represents a basic traffic situation, and matrix V contains the weights of linear combination of each basic traffic situation, and we can derive the cluster label from matrix V. Specifically, for one training row X_i of X, similar as [20], we assign the cluster label by examining V_i. More precisely, X_i is assigned to traffic situation c if $c = \arg\max_j V_{ij}$. Because NMF is NP-hard problem, we use projected gradient descent [21] to solve Equation 3.

Through these steps, we can generate k partitions of X, and each partition $P_c (c \in [1, k])$ contains the training data across T different sensors, i.e.,

$$P_c = \bigcup_{t=1}^{T} \{x_t^c, y_t^c\}, \tag{4}$$

where x_t^c and y_t^c is the corresponding training input and output of sensor t that belongs to cluster c. Note that it is possible that one situation group may not contain the training samples from each and every sensor, because the training samples of one sensor may only belong to a subset of all traffic situations.



(a) Variance of training features (b) Distribution of travel speed

Fig. 4. Traffic situations of one sensor.

Example 2: *Figure 4 shows two identified traffic situations of one sensor and their properties after the clustering steps, where Situations I and II can be treated as congested scenario and normal condition, respectively. We observe clear differences between the two traffic situations, where we can build a model for each situation instead of training one model for the mixture of the two situations. Specifically, Figure 4 (a) and (b) show the variance of one sensor's six continuous readings (i.e., training features) and the speed distributions (i.e., prediction targets) of this sensor under each traffic situation for a two month's period, respectively. Comparing with the scenario that traffic situations have not been considered (i.e., situation all), we observe that the variance of training features tends to be larger in situation I, while smaller in situation II; on the other hand, the sensor contains smaller speed values (i.e., less than 50) in situation I, while larger values (i.e., larger than 40) in situation II. Hence, the traffic conditions can be naturally divided into two subgroups. Intuitively speed reading varies significantly (i.e., larger variance) under the congested scenario (i.e., smaller speed values) and vice versa, we thus regard situation I and II as congested scenario and normal condition, respectively.*

D. Multi-Task Learning Per Traffic Situation

Within each partition c, our objective is to learn the parameter matrix of W^c of the corresponding sensors, where each column of W^c (i.e., W_t^c) is the model parameter for sensor t. To explore the commonalities under one traffic situation, we propose to simultaneously build forecasting models for all sensors with Multi-Task Learning. This is because, under each traffic situation, sensors exhibit similar behaviors and transition patterns. For example, during the rush hour, the process of transitioning from normal condition to congested state, and then back to normal condition should be similar across different sensors. Hence the prediction models of sensors within each situation can share some commonalities, and exploring these commonalities should help improve the prediction performance. In addition, learning different sensors together can increase the sample size of each sensor, which addresses the issue that some sensors only have limited number of training samples for one specific traffic situation.

We therefore utilize multi-task feature learning [17], [22] that selects a common set of features across multiple related models. That is, within each traffic situation we restrict that the regression model of each sensor shares the same subset of features. For example, the prediction models under rush hour situation can mainly rely on features such as time of the day, day of the week, whether or not in the boundary of rush hour and the previous reading. This can be achieved by employing the group lasso regularization, i.e., penalizing the sum of the l_2 norms of each row in W^c. In addition, we also include the l_2 norm regularization for W^c to increase the robustness of the model. For each traffic situation partition P_c, we formulate our MTL problem as follows:

$$\arg\min_{W^c} \sum_{t=1}^{T} ||y_t^c - x_i^c W_t^c||_F^2 + \rho_1 ||W^c||_{2,1} + \rho_2 ||W^c||_F^2 \tag{5}$$

where the first term is the data fitting term for all sensors (i.e., Frobenius norm), ρ_1 and ρ_2 are the regularization parameters for group lasso and l_2 norm penalty, respectively.

In Equation 5, the $l_{2,1}$ norm is given by $||W^c||_{2,1} = \sum_{d=1}^{D} \sqrt{\sum_t {W_{dt}^c}^2}$, where it computes the 2-norm of parameter values in each feature dimension across all sensors. Consequently, for any feature d, this regularization achieves the minimal value when the corresponding parameters of one row are all zeros, i.e., W_{dt}^c for all t. This guarantees that l_{21} norm would choose the W_c with the smallest number of non-zero rows, which is the same as finding a subset of sharing features that are used for all sensors.

1) Algorithm of Solving Equation 5: We adapt FISTA [15] algorithm to solve the convex optimization problem. FISTA was a Proximal Gradient method that can be used to solve the composite optimization problem. FISTA combines the idea of ISTA (Iterative Shrinkage-Thresholding Algorithm) [23] with Nestorov's Accelerated Gradient Descent [24], and converges faster than ISTA. In particular, our objective is to minimize $F(W) = f(W) + g(W)$, where $f(W)$ and $g(W)$ are defined as follows (we ignore the cluster label c for x_t^c, y_t^c and W^c hereafter):

$$\begin{aligned} f(W) &= \sum_{t=1}^{T} ||y_t - x_t W||_F^2 + \rho_2 ||W||_F^2 \\ g(W) &= \rho_1 ||W||_{2,1} \end{aligned} \tag{6}$$

As an iterative algorithm, given the current search point W_i, similar as ISTA, FISTA generates the next point W_{i+1} by first finding $F(W)$'s quadratic approximation Q at point W_i. The

quadratic approximation function $Q(W, W_i)$ at W_i is defined as follows,

$$Q(W, W_i) = f(W_i) + < W - W_i, \nabla f(W_i) > + \frac{\eta}{2}||W - W_i||^2 + g(W) \quad (7)$$

The next point $W_{i+1} = \arg\min_W Q(W, W_i)$, is simplified as follows after ignoring constant terms of Q:

$$W_{i+1} = \arg\min_W g(W) + \frac{\eta}{2}||W - (W_i - \frac{1}{\eta}\nabla f(W_i))||^2 \quad (8)$$

Besides using one sequence $\{W_i\}$, similar with Nestorov's algorithm, FISTA utilizes another sequence $\{Z_i\}$, where $\{W_i\}$ is the sequence for approximation solutions, and $\{Z_i\}$ is the sequence of search points. The approximate solution of W_{i+1} is generated at the search point Z_i, and Z_i is the affine combination of W_{i-1} and W_i.

The details of FISTA algorithm is presented in Algorithm 2. At each iteration, we first generate the search point of Z_i from W_i and W_{i-1} (Line 4), and calculate W_{i+1} from the search point of Z_i (Line 7). Finally, the step size η at each step is calculated via line search according to Armijo rule [25].

Algorithm 2: FISTA Algorithm

Input: $\forall t, x_t, y_t$ and W_0
Output: W

1 Initialize $W_1 = W_0, \lambda_{-1} = 0, \lambda_0 = 1, \eta = 1$;
2 **for** $i = 1, 2, \cdots$, **do**
3 $\alpha_i \leftarrow \frac{\lambda_{i-2}-1}{\lambda_{i-1}}$;
4 $Z_i \leftarrow (1 - \alpha_i)W_i + \alpha_i W_{i-1}$;
5 **for** $j \leftarrow 0, 1, \cdots$ **do**
6 $\eta \leftarrow 2^j \eta_{i-1}$;
7 $W_i \leftarrow \arg\min_Z g(Z) + \frac{\eta}{2}||Z - (Z_i - \frac{1}{\eta}\nabla f(Z_i))||_F^2$;
8 **if** $F(W_i) \leq Q(W_i, Z_i)$ **then**
9 $\eta_i \leftarrow \eta$;
10 break;
11 **end**
12 **end**
13 $\lambda_i \leftarrow \frac{1+\sqrt{1+4\lambda_{i-1}^2}}{2}$;
14 **if** *Convergence* **then**
15 $W \leftarrow W_i$;
16 break;
17 **end**
18 **end**

2) Time Complexity and Convergence: We first analyze the time complexity of Algorithm 2. For each iteration, it costs $O(mD)$ to evaluate $F(w)$ and $O(DT)$ to find the approximation point, where m is the total number of training samples, D is the number of feature dimension, and T is the number of sensors (i.e., tasks). As shown in [15], FISTA has a convergence rate of $O(\frac{1}{\epsilon^2})$, while gradient and subgradient descent have convergence rates of $O(\frac{1}{\epsilon})$ and $O(\frac{1}{\sqrt{\epsilon}})$, where ϵ denotes the number of iteration.

IV. Experiment

We performed extensive empirical evaluation of our approach to evaluate its effectiveness and efficiency. We first compare our methods with various categories of baseline methods. We then show the effect of varying different parameters. Finally, we provide several case studies to demonstrate the advantage of our methods.

A. Experiment Design

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 include 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. The sensor data are aggregated into 5 minutes interval. We chose four months data with 880 sensors in total from January/2014 to April/2014 for our experiment.

2) Algorithms: We compared our proposed algorithms **SA-MTL** with the following baseline approaches:

- Straightforward methods require little training effort: Historical Average Model (**HAM**), Random Walk (**RW**) (i.e., using the most recent readings as the predicted speed),.
- Single sensor models: we independently train a model for each sensor with the following different models: Ridge Regression Model (**Ridge**) [26], Support Vector Regression (**SVR**) [12], Multi-level Neural Network (**Neural**) [27], Random Forest (**Forest**) [26] and time series approach Autoregressive Integrated Moving Average (**ARIMA**) [5]. Note that RW is a special case of ARIMA, i.e., ARIMA(0, 1, 0).[2]
- Multi sensor models: Multi-Task Feature Learning without considering traffic situations (i.e., **Naive-MTL** [17]), Clustered Multi-Task Learning (i.e., **CMTL** [28], [29]) that simply groups certain sensors together. We use the implementation from [30].

3) Configurations and Measures: For each sensor t, we generated the training and testing samples from our historical sensor dataset as follows: suppose the current time is γ, in order to predict $v(\gamma + h)$, we mainly utilized the knowledge of previous lag readings, i.e., $v(\gamma-1), v(\gamma-2), \cdots, v(\gamma-lag)$ as our features. We tested different values of lag, and find that fixing $lag = 6$ achieves the good trade-off between accuracy and training complexity. In addition, we generated historical features, i.e, for each weekday time (e.g., Monday 4pm) of one sensor, we aggregated the previous readings of this sensor at that specific time from the past three months. Besides features from sensor readings, basic contextual information (e.g., time of day, day of week) and sensor attributes (e.g.,

[2]We did not compare our methods with latent space modeling (LSM) approach [13] because we have cleaned the sensors with missing values, and in [13], LSM achieves similar performance with SVR when the missing value issue has been addressed. We did not compare our methods with the latest deep learning approach [10] either, because the post-accident scenario has not been explicitly introduced in our experiments.

TABLE II
SHORT TERM PREDICTION PERFORMANCE (RMSE)

	Rush Hour	Non Rush-Hour
HAM	10.65	7.59
RW	5.32	4.50
Ridge	5.54	4.38
Neural	5.80	4.63
Forest	6.13	4.90
SVR	5.64	4.49
ARIMA	5.71	4.75
Naive-MTL	5.49	4.37
C-MTL	8.24	7.27
SA-MTL	**4.54**	**3.72**

TABLE III
LONG TERM PREDICTION PERFORMANCE (RMSE)

	Rush Hour	Non-Rush Hour
HAM	11.34	8.46
RW	10.22	8.06
Ridge	8.91	6.86
Neural	9.11	7.2
Forest	9.58	7.38
SVR	8.86	7.03
ARIMA	8.89	7.23
Naive-MTL	8.91	6.85
C-MTL	15.82	13.28
SA-MTL	**6.4**	**4.83**

sensor category, direction) were also included as our training features. We utilized two months data, where March data are used for generating training samples and April for testing purpose, respectively. The historical features were generated from three months sensor data (i.e., from January to March). In our experiments, we did not incorporate weather and incident data as the training features and we applied the same features across all our models except for ARIMA because ARIMA is a single variate time series approach. For ARIMA, we only used March data for training and April data for testing.

We varied different parameters of our SA-MTL model, namely, the number of features (i.e., whether or not to include historical feature), the clustering algorithm and the number of clusters for our traffic situation identification, and the regularization parameter ρ_1 for multi-task learning. We used 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 RMSE, because the results based on MAPE are similar to those of RMSE. We manually distinguished rush hour (i.e., 7am-9am, 4pm-7pm) and non-rush hour time ranges, and we reported RMSE for both of them. We conducted both short term and long term traffic prediction, where $h = 1$ (i.e., 5 minutes) for short term and $h = 6$ (i.e., 30 minutes) for long term, respectively.

The definition of RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

For each tested method, we use 5-fold cross-validation to choose the best parameters, and report the corresponding results. We conducted our experiments on a Linux PC with i5-2400 CPU @ 3.10G HZ and 24GB memory.

B. Performance Comparisons

1) Short-term Traffic Prediction: Table II shows the performance comparisons when $h = 1$, i.e., we predict the traffic for the next five minutes. Among all methods, SA-MTL achieves the best performance, with over 18% and 13% improvement compared with the baseline methods for rush and non-rush hour, respectively. The prediction error in rush hour is higher than that of non-rush hour, because the traffic condition in rush hour is more complex for prediction. We clearly observe that by incorporating traffic situations, SA-MTL performs significantly better than Naive-MTL. Even for non-rush hour, 13% decrease of error is significant given the already accurate prediction of short term traffic in non rush-hour due to its repetitive pattern. On the other hand, Naive-MTL performs similarly to other single sensor models (e.g., Ridge, SVR) that train a model for each sensor independently, and C-MTL that simply groups sensors without considering traffic situations even yields worse performance than Naive-MTL. These observations support our hypothesis that grouping sensors together without considering their commonalties under different traffic situations, will have no effect or even negative effect in traffic prediction. For other baselines, RW achieves good performance (even better than other single sensor models) and HAM performs worse, which shows that the most recent reading is a strong indicator for short term prediction, while historical feature dose not help much. Among single sensor models, Ridge, Neural and SVR achieve similar results, while Forest performs the worst among them.

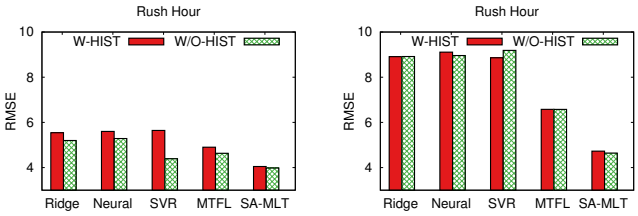
2) Long-term Traffic Prediction: We now present the experiment results for long-term prediction in Table III, with which we predict the traffic conditions for the next 30 minutes (i.e., $h = 6$). Basically long term prediction is harder than short term prediction considering many factors can affect the traffic in the next 30 minutes. In practice, typically there exist two categories of methods for long term prediction, i.e., 1) training a model for each horizon or 2) iteratively applying the previously predicted values as the input for the next prediction task. In our experiments, we chose the first method where we train a model for each prediction horizon, because the iterative methods can suffer from the error accumulation problem. As shown in Table III, SA-MTL achieves the best accuracy with more than 30% improvement over the best baseline for each situation, which is more significant than that of short term prediction. Meanwhile, SA-MTL performs much better than Naive-MTL and C-MTL, which shows the effectiveness of exploring commonalities after identifying the traffic situations. Interestingly, simple methods such as HAM and RW no longer work well for long term predictions, with much worse performance than single sensor models. For single sensor models, SVR performs better than Ridge, Neural and Forest.

TABLE IV
RUNNING TIME COMPARISON OF DIFFERENT METHODS

	HAM	RW	Ridge	Neural	Forest	SVR	ARIMA	MTFL	CMTL	SA-MTL
Training Time (sec)	1256.37	0.008	15.9	413.11	7719.15	4122.86	161.24	19.03	207.83	68.25
Testing Time (sec)	1.21	0.86	0.91	12.92	3.42	168.46	4.946	0.206	0.17	1.34

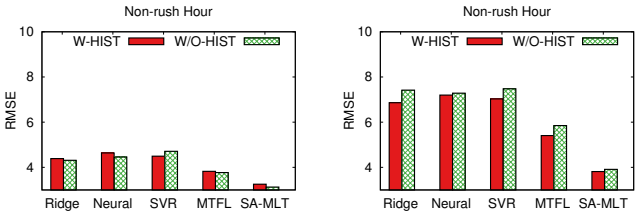
3) Running Time comparisons: Table IV shows the running time of different methods at both training and testing phases. Among all single-sensor models, Ridge is the most efficient because it is a linear model, whereas other non-linear models such as Neural, Forest, and SVR require significantly more training time. On the other hand, as multi-sensor models, Naive-MTL and C-MTL require similar training time as Ridge. Compared with Ridge and Naive-MTL, our method SA-MTL takes a little longer time because we require extra steps to identify the traffic situations, and then apply MTL for each identified situation. However, even including extra steps, the training time of SA-MTL is still efficient and takes less than 70 seconds, thus it can easily scale to large number of sensors. The testing time includes the prediction time for all testing samples of all sensors. We note that the testing time is mostly negligible, except for Neural and SVR.

C. Varying Parameters



(a) Short term prediction (b) Long term prediction

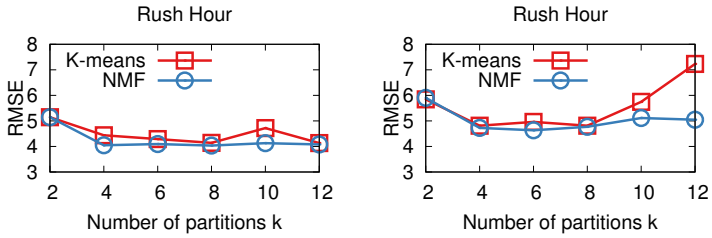
Fig. 5. Effect of using historical feature on Rush hour



(a) Short term prediction (b) Long term prediction

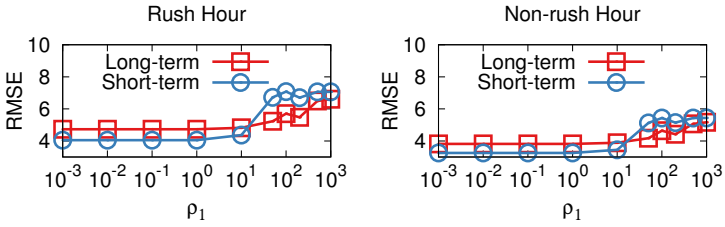
Fig. 6. Effect of using historical feature on Non-Rush hour

1) Effect of Using Historical Features: We evaluated the effect of using different set of training features, here we presented the results of traffic prediction with/without historical average speed readings in Figure 5. As shown in Figure 5(a), the historical aggregated averaged readings does not help for short term prediction, this conforms with the results in Table II: HAM achieves the worst accuracy, whereas using the previous reading (RW) has relatively good performance. However, as shown in Figure 5(b), the historical feature does not help for long term prediction, either. This is a bit contradictory with our common knowledge that historical average can be a good predictor for long term prediction. The reason is that from our observations, the historical values reflect the trend of ground truth value mostly at the transitioning time (i.e., the boundary of rush hour), but would not closely correlate with the readings under other traffic conditions. Moreover, the traffic prediction tasks are dominated by those non-transitioning time instances, hence historical feature does not help much in long term prediction. Figures 6 (a) and (b) show similar effect during the non-rush hour time.



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

Fig. 7. Effect of varying k on Rush hour and Non-Rush hour



(a) Short term prediction (b) Long term prediction

Fig. 8. Effect of varying ρ_1 on Rush hour

2) Effect of Varying k: The results of varying different clustering algorithms (i.e., NMF v.s. K-means) and the number of cluster k (i.e, traffic situations) are shown in Figures 7 (a) and (b). We observe that NMF and K-means achieve similar results, and NMF performs slightly better than K-means. This shows the superiority of NMF of discovering the latent traffic situations. For the number cluster k, we note that $k = 4$ achieves the best performance for both short and long term traffic predictions. This indicates that even though traffic conditions are chaotic and complex, we can distinguish its underlying latent grouping with limited number of traffic situations. This is one of the main reasons that our SA-MTL framework is effective and efficient in traffic prediction tasks.

3) Effect of Varying Hyper-parameters ρ_1 in Equation 5: Figures 8 (a) and (b) show the effect of varying ρ_1 from 0.0001 to 1000. We observe that ρ_1 achieves the best RMSE value when $\rho_1 = 1$. When ρ_1 becomes larger, the performances become worse. Therefore, setting k to smaller values (e.g., 1) can yield better accuracy.

D. Case Studies

Here we present two case studies of predicted values versus observed ground truth to demonstrates the superiority of

SA-MTL under various traffic situations such as rush hour transition which mimics the patterns of accidents and other abrupt situations and is the most difficult to predict. Figure 9 plots the prediction values for sensor 168. Compared with the best baseline method, we notice that our prediction has less error at the transition time, especially at the range of [12pm, 13pm] and [18pm, 19pm], whereas the best baseline cannot capture these abrupt changes. Similar results occur for sensor 661 in the transitioning periods of morning rush hour.

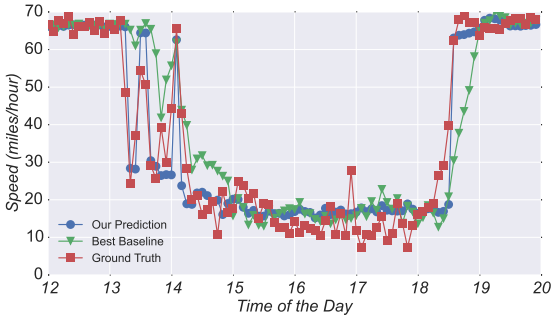


Fig. 9. Case study for sensor 168 with $h = 6$.

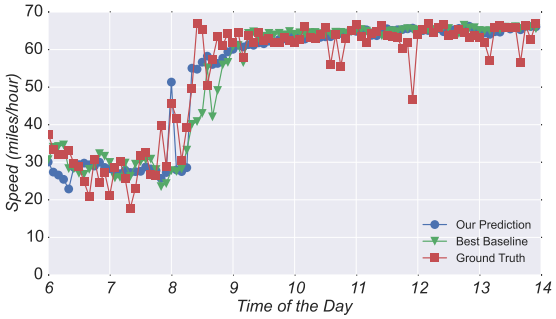


Fig. 10. Case study for sensor 661 with $h = 6$.

V. Related Work

A. Traffic Prediction Analysis

With the increasing demand for traffic prediction, various approaches have been proposed to predict traffic. Some representative techniques include Historical Average Model (HAM) [31], ARIMA [5]–[7], Support Vector Regression (SVR) [12], Gaussian Process (GP) [11], [32] and Deep Learning [10], [33], [34]. A recent survey of traffic prediction techniques can be found in [35], [36]. Most existing studies train a prediction model for each sensor independently, and they aim at predicting in specific traffic situation, e.g., either typical condition or when accident occurs. For instance, ARIMA [6] and SVR [12] have been applied in this manner. Yu et. al. [10] recently applied deep learning methods for traffic prediction for either rush hour and post-accident scenarios. One disadvantage of training models independently is that the commonalities across different sensors have been neglected. In addition, training a model per sensor and per traffic situation usually incurs high computational complexity for scenarios with thousands of sensors.

Some studies utilize multivariate (e.g., VARIMA [7]) or spatiotemporal models (e.g., HMM [8]) by considering multiple loop detectors together, but for small number of sensors. Similar as ARIMA [5], VARIMA [7], [37] has difficulty to capture near-boundary and transitional behaviors (e.g., the beginning and end of rush hour) in traffic flows, i.e., they focus on one specific traffic situations. Deng et. al. [13] proposed Latent Space Model for Road Network (LSM-RN) to estimate how traffic patterns form and evolve. Their model performs training and prediction on-the-fly by only utilizing the most recent traffic data, but does not consider traffic situations. In addition, LSM-RN does not perform well for long-term predictions because LSM-RN ignores the large amount of historical traffic data. Xu et. al. [38] train several base predictors from the representative traffic situations, and make the prediction by matching the current traffic situation to the most effective model. The focus of [38] is to dynamically update and identify the traffic situations, where they also independently train the base models for several chosen sensors. Different with existing studies, we identify the traffic situations via unsupervised learning (i.e., clustering methods), then apply supervised learning (i.e., multi-task learning) to simultaneously train the prediction models for all sensors per traffic situation, we thus can find the best prediction model for each traffic situation.

B. Multi-Task Learning and Mixture of Experts

Multi-task learning (MTL) [16], [39] learns multiple related tasks simultaneously by extracting and utilizing shared information, thus resulting in better generalization performance than independently learning each task. A key aspect of MTL is about how to extract and exploit the commonality among different tasks, i.e., the task relatedness. In [16], Evgeniou et. al. proposed the regularized MTL which constrained the models of all tasks to be close to each other. Another formulations of task relatedness include constraining multiple tasks to share a common set of features [17], or a common subspace [40], [41]. MTL has been applied in various domains such as computer vision [42], natural language processing [40], social event prediction [43], etc. Recently, MTL has been incorporated into the deep learning architecture [33] to provide better traffic prediction. Different with [33], our proposed SA-MTL framework applies MTL for each identified traffic situation.

Mixture of experts (ME), proposed by Jacobs et. al. [44], [45], is a learning paradigm that divides one task into a subset of distinct tasks (i.e., expert), and then utilizes a gating network to select the best expert. In our traffic prediction scenario, for each sensor, we learn an individual model for every traffic situation, which can be regarded as a special case of ME. Different with traditional ME, we explore the commonalities across all sensors, i.e., we assume that the traffic situations are shared among sensors, and MTL can thus be applied for each identified traffic situation to improve the prediction accuracy. This also addresses the issue that one sensor does not contain enough training samples for a specific traffic situation.

VI. Conclusion

In this paper, we presented a novel Multi-Task Learning framework for the problem of traffic prediction. Different with

existing MTL formulation that trains a model per sensor, our framework automatically identifies the basic traffic situations, and simultaneously trains a model per traffic situation. By extensive experimental evaluation, we showed that our proposed model can accurately capture traffic situations and achieve significant improvement for both short and long term predictions, especially for long term prediction.

Acknowledgments. This research has been funded in part by NSF grants IIS-1320149, and CNS-1461963, Caltrans-65A0533, the USC Integrated Media Systems Center (IMSC), and the USC METRANS Transportation Center. 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.

REFERENCES

[1] K. F. Petty, P. Bickel, M. Ostland, J. Rice, F. Schoenberg, J. Jiang, and Y. Ritov, "Accurate estimation of travel times from single-loop detectors," *Transportation Research Part A: Policy and Practice*, vol. 32, no. 1, pp. 1–17, 1998.

[2] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *KDD*, 2014, pp. 25–34.

[3] T. Idé and M. Sugiyama, "Trajectory regression on road networks." in *AAAI*, 2011.

[4] S. H. Kim, J. Shi, A. Alfarrarjeh, D. Xu, Y. Tan, and C. Shahabi, "Real-time traffic video analysis using intel viewmont coprocessor," in *International Workshop on Databases in Networked Information Systems*. Springer, 2013, pp. 150–160.

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

[6] B. L. Smith, B. M. Williams, and R. K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 4, pp. 303–321, 2002.

[7] W. Min and L. Wynter, "Real-time road traffic prediction with spatio-temporal correlations," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 4, pp. 606–616, 2011.

[8] 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*, vol. 6, no. 9, pp. 769–780, 2013.

[9] B. Shahsavari and P. Abbeel, "Short-term traffic forecasting: Modeling and learning spatio-temporal relations in transportation networks using graph neural networks," Master's thesis, EECS Department, University of California, Berkeley, Dec 2015.

[10] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017.

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

[12] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE transactions on intelligent transportation systems*, vol. 5, no. 4, pp. 276–281, 2004.

[13] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, R. Yu, and Y. Liu, "Latent space model for road networks to predict time-varying traffic," in *SIGKDD*, 2016, pp. 1525–1534.

[14] J. Haworth and T. Cheng, "Non-parametric regression for space–time forecasting under missing data," *Computers, Environment and Urban Systems*, vol. 36, no. 6, pp. 538–550, 2012.

[15] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[16] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *KDD*, 2004, pp. 109–117.

[17] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Mach. Learn.*, vol. 73, no. 3, pp. 243–272, Dec. 2008.

[18] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

[19] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *NIPS*, 2001, pp. 556–562.

[20] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *SIGIR*, 2003, pp. 267–273.

[21] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[22] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient l 2, 1-norm minimization," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 339–348.

[23] K. Bredies and D. A. Lorenz, "Linear convergence of iterative soft-thresholding," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 813–837, 2008.

[24] Y. Nesterov, "Introductory lectures on convex programming volume i: Basic course," 1998.

[25] D. Bertsekas, "On the goldstein-levitin-polyak gradient projection method," *IEEE Transactions on automatic control*, vol. 21, no. 2, pp. 174–184, 1976.

[26] F. Guo, "Short-term traffic prediction under normal and abnormal conditions," Ph.D. dissertation, Imperial College London, United Kingdom, 2013.

[27] R. Yasdi, "Prediction of road traffic using a neural network approach," *Neural computing & applications*, vol. 8, no. 2, pp. 135–142, 1999.

[28] L. Jacob, J.-p. Vert, and F. R. Bach, "Clustered multi-task learning: A convex formulation," in *Advances in neural information processing systems*, 2009, pp. 745–752.

[29] J. Zhou, J. Chen, and J. Ye, "Clustered multi-task learning via alternating structure optimization," in *Advances in neural information processing systems*, 2011, pp. 702–710.

[30] ——, "Malsar: Multi-task learning via structural regularization," *Arizona State University*, 2011.

[31] I. Kaysi, M. Ben-Akiva, and H. Koutsopoulos, "Integrated approach to vehicle routing and congestion prediction for real-time driver guidance," *Transportation Research Record*, no. 1408, 1993.

[32] A. Girard, C. E. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," in *Advances in Neural Information Processing Systems*, 2003, pp. 545–552.

[33] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, 2014.

[34] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic speed prediction and congestion source exploration: A deep learning method," in *ICDM*, Dec 2016, pp. 499–508.

[35] E. I. Vlahogianni, J. C. Golias, and M. G. Karlaftis, "Short-term traffic forecasting: Overview of objectives and methods," *Transport reviews*, vol. 24, no. 5, pp. 533–557, 2004.

[36] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 3–19, 2014.

[37] Y. Kamarianakis and P. Prastacos, "Space–time modeling of traffic flow," *Computers & Geosciences*, vol. 31, no. 2, pp. 119–133, 2005.

[38] 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*, vol. 9, no. 4, pp. 702–715, 2015.

[39] R. Caruana, "Multitask learning," in *Learning to learn*. Springer, 1998, pp. 95–133.

[40] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *Journal of Machine Learning Research*, vol. 6, no. Nov, pp. 1817–1853, 2005.

[41] A. Agarwal, S. Gerber, and H. Daume, "Learning multiple tasks using manifold regularization," in *Advances in neural information processing systems*, 2010, pp. 46–54.

[42] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing visual features for multiclass and multiview object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 5, 2007.

[43] L. Zhao, Q. Sun, J. Ye, F. Chen, C.-T. Lu, and N. Ramakrishnan, "Multi-task learning for spatio-temporal event forecasting," in *KDD*, 2015, pp. 1503–1512.

[44] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[45] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.