Efficient Traffic Speed Forecasting Based on Massive Heterogeneous Historical Data

Xing-Yu Chen*, Hsing-Kuo Pao†, Yuh-Jye Lee‡
Department of Computer Science, National Taiwan University of Science and Technology
Taipei, Taiwan
*Email: *star30834@hotmail.com, †pao@mail.ntust.edu.tw, ‡yuh-jye@mail.ntust.edu.tw*

***Abstract*—Drivers dream of foreseeing traffic condition to enjoy efficient driving experience at all times. Given the historical patterns for different locations and different time, people should be able to guess the possible traffic speed in a near future moment. What is difficult and interesting for this task is that we need to filter the useful data that could help us for the next moment traffic speed prediction from a massive amount of historical data. On the other hand, the traffic condition could be highly dynamic and we can only give a reliable traffic prediction by using the most updated model for prediction. This implies that frequent retraining is necessary. To conquer the task, we propose a lazy learning approach for traffic speed prediction given massive historical data. The approach integrates the kNN and Gaussian process regression for efficient and robust traffic speed prediction. kNN can help us to select the most informative data for Gaussian process Regression using a big data framework. Thanks for the most recent progress of big data research, the processing of massive data for prediction in close to real time has become possible now compared to any time in the past. We aim at using a `Hadoop` framework for the prediction given heterogeneous data including traffic data such as speed, flow, occupancy, and weather data.**

***Keywords*-Big data; Gaussian Process Regression; Hadoop, Intelligent Transportation Systems (ITS); K Nearest Neighbors (KNN); MapReduce;**

I. Introduction

Letting drivers know how long to drive from one place to another is as important as any other daily tasks. Based on the data driven approach [1], we can use the past data to guide us to estimate the possible time to take in a journey. To estimate the travel time, we can focus on predicting traffic speed for a given time and location and compute the total elapsed time in the journey. Experienced drivers know how to avoid rush hours and bottleneck routes to shorten his traveling time. Clearly, how a driver is judged as an experienced one is highly related to how much past experience the driver has. One of the ultimate questions to ask is: how well can we dynamically predict the traffic speed in real time or close to real time given *all* the past data?

In an Intelligent Transportation System (ITS), we continuously collect data such as *speed*, *flow* and *occupancy* to monitor traffic conditions at all times. Based on the ITS information, decision makers can study traffic congestion, bottleneck routes, and various types of incidents to judge how a specially designed route topology, signal control, ramp control, etc. can achieve high traffic efficiency. Most of the time, the decision is made time dependent. Between rush hours and non-rush hours, between weekdays and weekends, the traffic patterns are generally different. To design a smart ITS, researchers and technicians around the world have started to focus on collecting data for analysis and continue on improving traffic efficiency. In this work, given a location, we aim to predict traffic speed efficiently using all available historical data related to the prediction. To have a better prediction result, we would like to dynamically change the prediction model so that the prediction can be as updated as possible given different rush hour conditions at different time. Moreover, we would like to integrate various factors that can influence traffic patterns to provide as accurate prediction result as possible. We manage the massive historical ITS data using the `MapReduce`-based `Hadoop` framework.

To master the massive historical data for efficient speed prediction, we have to carefully select limited informative data for training given limited computing resources. Some issues must be considered for this problem, including:

1) As mentioned above, traffic speed is highly dependent on *time*. People all know that driving experience between rush hours and non-rush, also between weekdays and weekends are different. We should be able to predict traffic speed for different time. Given that need, we build different profiles for both rush hours and non-rush hours, for weekdays, weekends, even special holidays to make the prediction reliable.
2) Due to the above reason, we have to collect data as many as possible for many different kinds of occasions. Robust prediction is usually based on an as complete as possible feature set and by using more features, in general, we need more data to make learning possible [2]. After all, we have to deal with large-scale data, large in the sense of data number and the number of features. Intuitively speaking, when the learning problem is too "case-by-case", we have to collect a large amount of data to make the learning robust.
3) The data necessary for speed prediction have their natural spatial and temporal relationships. Traffic data in this moment and in the previous moment are usually

correlated. On the other hand, traffic data in nearby locations are also usually correlated. Traffic congestion in the highway may also introduce traffic congestion in the ramps that lead to this highway, or even the local area near the highway entrance. Knowing the above facts, a model that can perform well for speed prediction must be able to consider these two relationships.

4) Both ITS and non-ITS data can influence the traffic speed. How the traffic is at a time of a day depends on not just the ITS-related factors such as how many cars on roads or how fast drivers drive, etc. It also depends on weather conditions such as whether it is raining or snowing and how fast local officers try to deal with the weather conditions, if the conditions become severe enough to introduce danger. After all, to effectively predict traffic speed, one must collect not only the ITS data, but also non-ITS data such as weather data, local special event schedules, policemen patrol records, traffic broadcasting or smartphone auto-routing information, etc. All the information makes up the database that is needed for speed prediction to be heterogeneous.
5) It is favored if we can predict traffic speed with confidence level. The confidence level is useful for people who are in charge of traffic control and transportation management. For instance, speed prediction can be used for incident detection and the prediction can give a possible incident warning to policemen. The confidence level in this case can suggest policemen which incident owns a higher priority than others if more than one incident occurs at a time.

We need a traffic speed forecasting model to address the above issues. Other than that, we hope the model can be efficient enough so that the forecasting can be used in real cases. We propose a method to predict traffic speed based on a k Nearest Neighbors (kNN) and Gaussian Process Regression (GPR) combined approach given massive data which have spatial and temporal relationships. Gaussian process is known to be good at dealing with spatio-temporal relationships among data and can provide confidence level for the prediction result. Combining kNN and GPR is mainly an computation issue. Gaussian Process cannot be scaled up easily for its cubic learning computation and quadratic space requirement [3]. To look for a *structured* Gaussian process, i.e., a Gaussian process with a structured covariance matrix [4], we need to analyze how data are distributed in the space which may not be possible for massively generated data and for online prediction tasks. To work around the problem, we combine kNN and GPR where we adopt kNN to select informative data in the aspect of GPR prediction performance. When the prediction performance remains similar with or without the data, kNN may choose to ignore them to make the GPR learning and inference operated afterwards more efficiently.

We organize the rest of the paper as follows. In Section II, we discuss some related work in the past. After that, we introduce the proposed method in Section III. To evaluate the proposed method, we first present a case study in Section IV and show how well the performance is for the proposed method on the case study dataset in Section V. In the end, we conclude our contribution in Section VI.

II. Past Work

A. Large-scale Learning via kNN and Gaussian Process

k-nearest neighbors is a simple non-parametric algorithm which can solve both classification and regression problems. Apart from classification and regression, searching for nearest neighbors is the key procedure of many useful algorithms such as recommendation [5], dimensionality reduction [6], computer networking [7], and so on. In the most naïve implementation, kNN, as a lazy approach needs $O(nD)$ to search for closest neighbors given a new data for prediction where n is the number of data points and D is the dimensionality. To scale up to large-scale problems, one has to use techniques such as kd-tree [8] for indexing, or to find "landmarks" in the dataset so that any new data can be categorized as one close to one of a few landmarks for classification or regression. There are also some hardware efforts for kNN computation acceleration [9], [10] for different applications. One has to know that both kd-tree and the landmark approach rely on indexing or building structures on the dataset, which may not be trivial for the kNN query on a massive dataset and on a dataset of high dimensionality.

Gaussian process needs $O(n^3)$ time for learning and $O(n^2)$ space for storage [3]. That makes GPR not practical to large-scale problems. Many GP or GPR related approaches for large-scale data basically look for "blocky" structures in the dataset [4] so that we can compute matrix inverse for smaller matrices instead for computation accerleration. One may also find an efficient lazy approach for real-time computation based on a set of Gaussian processes [11].

Our method also belongs to a lazy approach. What is different from the above work is that we first use kNN to select the most influential neighbors for GPR computation. Moreover, we take advantage of `MapReduce` framework and use it to efficiently select kNN for large-scale data. In the `Hadoop` environment, one input data will be split into many 64-MB blocks, and each block is processed by a `Map` task. To implement kNN in `MapReduce` architecture, our idea is to find the local k neighbors based on each data block. After that, the `Reduce` task may associate these local neighbors from each `Map` task and extract the global k neighbors of the input data from different `Map` tasks.

B. Big Data Framework

The `MapReduce` framework was proposed by Dean and Ghemawat in 2004 [12]. The framework is well known for its ability to deal with large-scale data in a distributed manner on computing clusters. `Map` and `Reduce` are the two key functions to implement the distributed computation and developers can design their own functions according to different applications. In the `Map` phase, we map large scale data to a set of output `key/value` pairs as intermediate data. Then, the intermediate data will be combined to the same group if they share the same key. Afterwards, in the `Reduce` phase, after receiving the sorted intermediate data, the `Reduce` function merges the data of the same key and compute an integrated value based on developers' requirement. Our forecasting method is implemented on the `MapReduce` platform.

C. Vehicle Traffic Prediction

The vehicle traffic prediction has a long history for its practical purpose. A rich set of research was devoted in traffic flow modeling, by various methods such as autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA) [13], Kalman filtering [14], particle filtering [15]. Clearly, those methods use a generative approach to model traffic data, which relies on strong assumptions such as Markovian assumptions between data of consecutive moments, or the stationarity of model parameters. The model can fail for anomalous events. Especially, the above methods cannot easily deal with sudden state or phase changes such as incidental events on roads, severe weather conditions, or abnormal rush hour patterns. In our approach, thanks to the big data framework, referencing historical data of similar patterns becomes possible and we have very little chance to face unfamiliar patterns for prediction given a massive historical data for learning. Second, we adopt GPR as part of the prediction components which allows long-term interactions between data. Moreover, we focus on modeling the spatial and temporal relationships between different ITS or non-ITS sensory data. To apply the above models for spatial-temporal modeling other than GPR could be theoretically intractable. We also need to point out that the above methods may not scale up to large-scale data easily for their high complexity computation. Recently, researchers have started to use large-scale framework for ITS data analysis, such as using distributed computation for acceleration [16].

III. Proposed Method

We propose a method to forecast traffic speed based on a `MapReduce` framework. The proposed approach has a few ingredients: kNN, Gaussian process regression and the `MapReduce` module for computation performance enhancement. Ideally, we can adopt either kNN or GPR to predict traffic speed on specific location and time given past data. Apparently, the probabilistic Gaussian process is more robust than kNN in its prediction performance. For instance, to predict the traffic speed at certain location and time, a few noisy data in the kNN neighborhood region can affect the prediction result. Moreover, GPR is more useful than kNN for its rich confidence level information. However, GPR is generally computationally intensive. Given the Gaussian process computation barrier, we have to solve a small-scale problem by selecting the limited influential data for GPR so that we can predict traffic speed efficiently. In this work, we combine kNN and GPR for efficient traffic speed prediction. We utilize kNN to select the "closest" data, assumed to be the most informative data of the predicted case to build a relatively small-scale Gaussian process and use the Gaussian process for traffic forecasting. After all, we use a `MapReduce` framework to find kNN and schedule tasks wisely to enhance the prediction performance. We discuss each of the components in the following subsections. Before doing that, we introduce the notations that are used in this work in Table I.

Table I
Notations used in this work.

Notation	**Definition**
h	an instance of historical data
$\mathcal{H}$	historical dataset $\mathcal{H} \equiv \{h_i : i = 1, 2, \cdots, n\}$ where n is the number of historical data
t	test data, i.e., the next moment traffic data for testing
$h.\mathbf{x}, h.y$ and $t.\mathbf{x}, t.y$	the feature vector and target value for h and t, where $\mathbf{x}$ denotes the feature vector and y denotes the target value
$\mathbf{X} = \{h_i.\mathbf{x}\}, Y = \{h_i.y\}$ and $\mathbf{x}^* = t.\mathbf{x}, y^* = t.y$	the whole historical and test feature vector sets and target value sets
k	the number of neighbors
$KTable$	an array for storing k neighbors
d	the distance two data or between h and t
Sensor ID	the unique sensor identifier of test data

A. Gaussian Process Regression

Given a historical dataset $\mathcal{H} \equiv \{h_i = (\mathbf{x}_i, y_i) : i = 1, 2, \ldots, n\}$ where $\mathbf{x}_i$ denotes an input vector and $y_i \in \mathbb{R}$ denotes the target value that is associated with $\mathbf{x}_i$, we would like to predict the traffic speed in the next moment (such as next five minutes) by GPR. The goal of GPR is to find a function f which can describe the relationship between $\mathbf{x}_i$ and y_i. Gaussian process has good properties where a Gaussian process can be completely specified by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$ or $\mathbf{x}'$ represents two input feature vectors. We use a Gaussian process with mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ to describe the relation between $\mathbf{x}$ and y as follows:

$$f \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \,. \tag{1}$$

In the real world, the data are often noisy. To allow a small difference between the true value and the observed data y, we consider a Gaussian noise model, which is written below:

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)\,, \tag{2}$$

where σ_n^2 controls the noise variance. To find the function f in GPR, we have to decide an appropriate kernel function k, that is, to define the pairwise relation between two feature vectors $\mathbf{x}$ and $\mathbf{x}'$. In this work, we focus on the traffic data that consist of various ITS and non-ITS attributes such as speed, occurrence, flow, visibility, as well as time and location of the predicted data. To describe the relation between two traffic data $\mathbf{x}$ and $\mathbf{x}'$, we take the common approach which is the squared exponential kernel function. That is, the kernel function that contains the noise term is written as follows:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}\|\mathbf{x} - \mathbf{x}'\|^2\right) + \sigma_n^2 \delta(\mathbf{x}, \mathbf{x}')\,, \tag{3}$$

which is to describe covariance between $\mathbf{x}$ and $\mathbf{x}'$, where $\delta(\mathbf{x}, \mathbf{x}')$ is the Kronecker delta function. Given the next moment traffic conditions $\mathbf{x}_*$, its traffic speed y_* is predicted by GPR as:

$$\begin{aligned} &y_* \mid \mathbf{x}_*, \mathbf{X}, Y \sim GP(k(\mathbf{x}_*, \mathbf{X})\left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I\right]^{-1} Y, \\ &k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})\left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I\right]^{-1} k(\mathbf{X}, \mathbf{x}_*))\,. \end{aligned} \tag{4}$$

As we can see, the prediction provides both the predicted value (mean) and the confidence level of the prediction (variance).

Note that we aim at developing a method that can predict the next moment traffic given *all* the past data. Why is it necessary to do so? The historical data that is related to the next moment prediction status could belong to a very old moment. For instance, we want to predict a traffic condition on a Christmas holiday and we may look for the traffic condition at the same time last year. In this sense, we have to store data for at least one year long so that data of similar patterns can be extracted from the database. On the other hand, the traffic condition is usually highly dynamic as time goes by. To give the most reliable prediction, one way is to frequently retrain the model and use the most recent model for prediction. Given the two above thoughts, we propose a lazy approach that can consider as many data as possible and retrain the model as frequently as possible for the prediction. One may prefer a wiser approach such as retraining the model when it is really necessary to do so. What we want to argue is that, given available computation resources in these days, as simple as the proposed method can easily solve the problem that may not be solvable before.

To make the prediction scalable to large-scale data, the approach is generally focused on solving the matrix inverse problem for smaller matrices. Alternatively, in our proposed method, instead of working on GPR directly, we select the most informative neighbors of $\mathbf{x}^*$ by kNN for GPR to obtain an approximated version of GPR result. In the end, we can output the result efficiently without sacrificing too much the prediction accuracy. Below we also discuss some implementation details. After that, we show how the proposed method is realized on the `MapReduce` framework.

1) Dataset and Attributes: We study the traffic data for traffic speed prediction. An efficient prediction is based on how we can catch the patterns for different time or different locations from the historical data. Given the next moment traffic condition, the kNN algorithm helps us to find the data that are close to the condition. The traffic condition is represented as a feature vector $\mathbf{x}$ which includes both of ITS and non-ITS data as follows:

$$\mathbf{x} = (\text{weekday?}, \text{speed}, \text{flow}, \text{occupancy}, \text{visibility})\,, \tag{5}$$

and the target value y is the speed data in the next moment. The feature weekday? is a binary indicator feature which is 1 if the traffic condition occurs at a weekday moment and 0 otherwise. Note that we obtain the visibility information from a weather database while other information is obtained from highway sensors. As an alternative choice, we may also include: 1) the location index, where we want to predict; 2) the time index, when we want to predict, or 3) both as the extra attributes in GPR, that we call spatial and temporal information respectively. That is, including the extra spatial and temporal information shall encourage kNN to select the data closer to the predicted location and time for GPR.

2) Weighted Euclidean Distance: Given the traffic data that consist of heterogeneous ITS and non-ITS attributes, we have to normalize them to appropriate scales; after that, we can comfortably define the metric for kNN and the covariance of attributes. For each pair of historical data h and test data t, we define their distance as:

$$d(h, t) = \sqrt{\sum_{i=1}^{D} w_i (h_i - t_i)^2} \tag{6}$$

where w_i is the normalizing factor for the i-th attribute and D is the number of features or dimensionality of the data space.

B. MapReduce Framework for Massive kNN Computation

We use the `MapReduce` framework to select the most important kNNs as the input of GPR. Given the framework, we divide the kNN selection task into several tiny tasks for distributed computation. Moreover, to further enhance the performance, we use Support Vector Regression (SVR) [17], [18] to learn how the `MapReduce` framework can be improved by assigning different `Map` and `Reduce` slots to different machines, further described below.

1) Map Phase: In the `Map` task, the `MapReduce` framework will keep retrieving a pair of `key` and `value` from the corresponding data block until all data have been exhausted. The `key` is the file offset of data block and the `value` is an instance from the data block.

In our proposed Map procedure, the first step is to load next moment traffic condition t as the test data which contains all the features from HDFS. After that, we initialize a $KTable$ for storing the k neighbors. For each instance h from $\mathcal{H}$, we calculate the distance d between h and t based on traffic condition data (speed, flow, occupancy, visibility, etc). After obtaining the distance, we decide whether or not d and h will be inserted into $KTable$ and key/value is set as one of the intermediate data. We set the key as the sensor ID of test data, and the value as (d, h).

2) Reduce Phase: In Reduce phase, we can get local k neighbors from each Map task. Here, we also initialize $KTable$ for storing the global k neighbors. According to local k neighbors, we aggregate these data and use the same decision method in the Map phase to insert instances into $KTable$. After processing all intermediate data from Map tasks, $KTable$ will store the global k neighbors. We choose k neighbors based on three different criteria for the prediction: with location index (spatial information), with time index (temporal information), with both indices (spatio-temporal information). Based on the different approaches, we recalculate all the distances d in $KTable$ and sort the $KTable$ again. The top k data in $KTable$ may be different for different approaches. Finally, we take top k data in $KTable$ to train a Gaussian process [3] and use the process to predict traffic speed in the next moment.

IV. Case Study

We evaluate the proposed method on a dataset that was provided by Research Data Exchange (RDE) [19]. RDE is a platform for ITS data sharing. Researchers and developers can obtain many kinds of source data such as freeway data, weather report, GPS, and so on. RDE contains nine data groups that were collected in different areas such as San Diego, Portland, etc. Each data group consists of multiple datasets from different organizations. In this paper, we choose the San Diego dataset for our experiments. Next, we will introduce two main source data: freeway data and weather report. We shall study the data based on the proposed method.

Freeway Data: Freeway data contain freeway traffic condition monitoring data. The data were obtained from the California Freeway Performance Measurement System (PeMS) [20]. The traffic data were collected by loop detectors deployed in the California state. The loop detectors collect the traffic data per 30 seconds. These data are also aggregated to be five-minute, hourly, and daily data. In our work, we use five-minute dataset and we extract the latitude, longitude, speed, flow, and occupancy data from each sensor. In Figure 1, we show different sensor readings: speed, flow and occupancy collected on sensors of one location, but in different days. The top two days are weekdays, the bottom one belongs to a weekend case. We can observe that they have different patterns in different days, especially between the weekday cases and the weekend case. Many people live in suburbs and work in urban districts at weekdays. They drive to return home after they get off from their work. That is the reason why the flow values in the afternoon are higher than the flow values in other time. The two weekday cases own similar patterns. In the weekend, people have different behavior such as they do not need to wake up early for work, so the flow show different patterns from the weekday cases. Different sensor readings have not just temporal relationship but also spatial relationship. We can observe that the data collected in similar time and nearby locations are likely to own similar patterns (figure not shown).

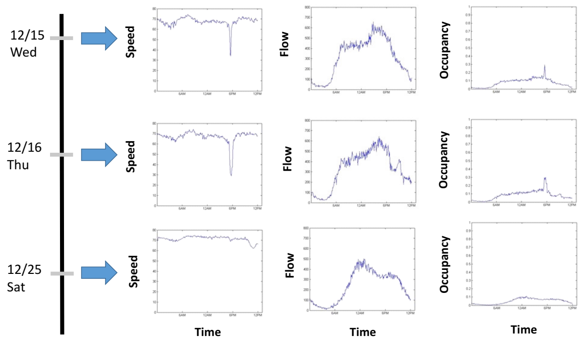

Figure 1. Traffic data collected in three different days: two from weekdays and one from Saturday.

Weather Report: The data is for weather monitoring and forecasting. Obviously, weather conditions affect not just weather itself, but also many things in our daily lives including freeway traffic. The data were obtained from the United States Government's National Oceanic and Atmospheric Administration (NOAA). The RDE dataset consists of one year weather data in 2010. The weather data contain multiple factors such as temperature, dew point, visibility, and so on. In our work, we extract only visibility data, from seven weather stations around San Diego.

V. Experiment Result

We conduct the experiments and organize them by a few different scenarios to demonstrate that the proposed method can predict traffic speed with high accuracy. For all the experiments, the computing clusters is based on Hadoop version 1.2.1 installed on one master node and four slave nodes with CentOS 6.4 operating system. Further details of the node information are shown in Table II.

A. Data Preparation

To evaluate the proposed method, we use San Diego data from RDE dataset. We focus on I5N road section in San Diego. The I5N road section consists of 92 sensors and these sensors are deployed in both of the north bound and south bound. In addition, there are seven weather sensors near the region in this dataset. The ITS data were continuously

Table II
DETAILS OF THE COMPUTING CLUSTERS.

	Master Node	Slave Nodes			
		1	2	3	4
# CPUs	2	8	8	6	6
memory (GB)	10	5	8	5	6
disk (TB)	2	1	1	1	1
CPU	Intel(R) Xeon(R) E5-2650 0 2.00GHz				

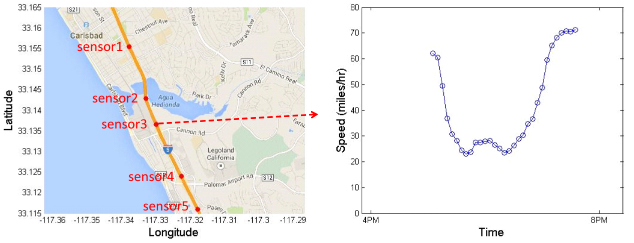

Figure 2. The left figure shows the five test sensors on I5N road section. The right figure shows the sensor 3 speed data from 16:40 to 19:35 at December 28, 2010.

collected all year long in 2010. For the weather information, we choose the closest weather sensor to extract the visibility data. We focus on the prediction accuracy of five test sensors of I5N road section for all experiments, as shown in Fig 2 unless otherwise specified.

B. *Experiment Result and Analysis*

We show how to evaluate the proposed method by several series of experiments. In the proposed kNN and GPR combined approach, after finding the k neighbors from historical traffic data, we use GPR to predict traffic speed with either spatial, temporal, or spatio-temporal information. We have several scenarios that consider different parameter settings, different location/time characteristics, and different data size, etc for the evaluation.

The basic experimental settings are as follows (if not specified otherwise): we choose the k in kNN to be 100, and we extract 5 to 70 instances for GPR by different approaches in the `Reduce` function. Regarding to GPR, we set the covariance function parameters as $\ell = 1000$, $\sigma_f = 10$, and $\sigma_n = 2$ as in Eq. 3. (further discussed in the next paragraph.) To evaluate the proposed method, we used Mean Absolute Error (MAE) which measures the average absolute error between real speed values (S_i) and forecasting result (F_i):

$$MAE = \frac{1}{n_t} \sum_{i=1}^{n_t} |S_i - F_i| \ , \tag{7}$$

where n_t is the number of values that we want to predict. We discuss different scenarios as follows.

1) Parameter Settings: The first question is to find as best parameter set as we can for GPR. We extract the data in all I5N sensors and weather sensors data from January 1, 2010 to December 27, 2010, then we choose sensor 3 (Fig 2) and focus on the traffic prediction of sensor 3 at December 28, 2010, starting from 16:40 to 19:35. We study four parameter settings. In this series, we set the k for kNN to be 100 and extract 25 instances in all experiments. Table III shows the result. First of all, among all different choices, $(\ell, \sigma_f, \sigma_n) = (1000, 10, 2)$ gives the best performance. In this case, we expect some noise in the data and the prediction may not perfectly match the input real data. To compare between different approaches, the approach with both spatial and temporal information is superior to others. We further discussed this topic in the next few paragraphs.

Table III
PARAMETER TUNING, THE RIGHTMOST SETTING GIVES THE BEST PERFORMANCE.

	Mean Absolute Error			
	$\ell = 1000$ $\sigma_f = 1$ $\sigma_n = 1$	$\ell = 1000$ $\sigma_f = 10$ $\sigma_n = 1$	$\ell = 1000$ $\sigma_f = 1$ $\sigma_n = 2$	$\ell = 1000$ $\sigma_f = 10$ $\sigma_n = 2$
w. Spatial	2.99	2.27	15.65	**2.14***
w. Temporal	2.76	2.75	15.47	**2.6***
w. Spatio-temporal	2.94	2.11	15.62	**1.99***

*=The minimum value per row

2) Between Different Sensors and Time Periods: In this scenario, we focus on five sensors shown in Fig 2 as the test sensors. We compare the forecasting result between five sensors and between different time periods. To do that, we choose two days: Dec. 25, 2010 (weekend) and Dec. 28, 2010 (weekday), and partition a day into three periods: 00:00-08:00, 08:00-16:00, and 16:00-24:00 for the comparison. The result is shown in Table IV. First of all, we observe that the MAE are basically similar between different nearby sensors and between different time periods when they are close. That explains why the approach including both spatial and temporal information is usually the best choice. However, it is not always true for all cases. The sensor speed values are generally influenced by traffic congestion happened in the weekday afternoons. The traffic congestion causes the speed values to be unstable. We can observe that the forecasting result on the period of 16:00 to 24:00 are worse than that in other time periods in the Tuesday weekday (Dec. 28). In the weekend (Dec. 25), the speed values of the whole day are relatively stable. The forecasting results of all time periods are similar in this day.

3) Different Data Size for GPR: Following the same experimental setting of the first scenario, we study the performance of the proposed method using different data size for GPR. Basically, we want to confirm that more data can really improve the forecasting result. The answer is true. In Fig 3, we see that more training data for GPR indeed give better performance. The result is going to be stable as the number of input instances goes larger than 65. To compare between different approaches, considering only the spatial information or both the spatial and temporal information at

Table IV
COMPARISON BETWEEN DIFFERENT SENSORS AND TIME PERIODS.

	Mean Absolute Error				
	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5
Dec. 25 Sat					
00:00-08:00	0.55	0.53	0.49	0.67	0.67
08:00-16:00	0.41	0.53	0.5	0.42	0.46
16:00-24:00	0.57	0.75	0.63	0.61	0.73
Dec. 28 Tue					
00:00-08:00	0.49	0.84	0.85	0.45	0.47
08:00-16:00	0.71	0.9	0.81	0.58	0.7
16:00-24:00	1.13	1.55	1.56	1.95	1.37

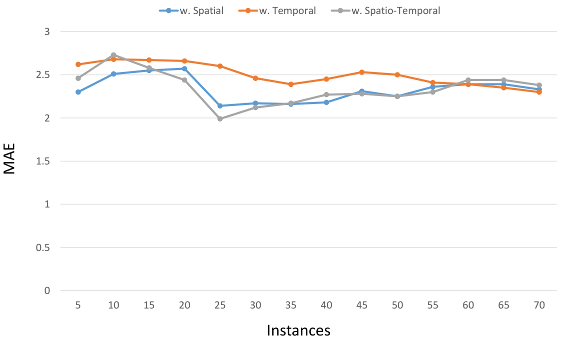

Figure 3. Different input instances for Gaussian process.

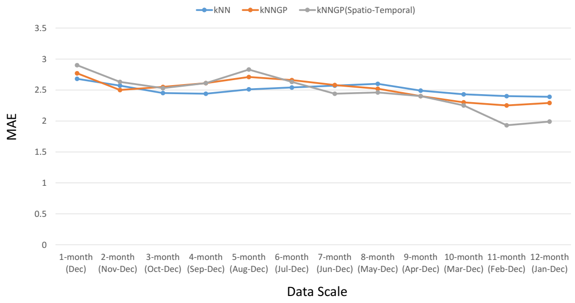

Figure 4. Different Training Periods.

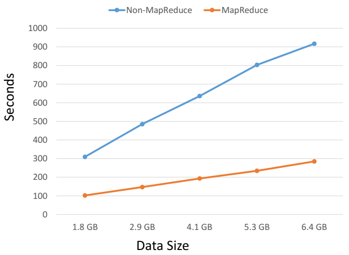

Figure 5. Compare the running time between Non-`MapReduce` and `MapReduce` environments.

the same time outputs the best result in most cases. The reason is again due to the usual traffic congestion in the weekday afternoon. That makes the temporal information not so reliable for prediction. One traffic condition now may not imply a similar condition in the next moment.

4) Different Training Periods: Following the same experimental settings from the first scenario, we test the proposed method given different portions of the historical data (or we call different training periods): from a couple of months to nearly the whole year data. According to different portions of the dataset, we use the same test sensor just like in the first scenario to forecast the traffic speed. In this scenario, we also compare three different approaches, which are 1) using kNN only, 2) using the integrated kNN and GPR, and 3) using the integrated kNN and GPR with both spatial and temporal information as the attributes to be the prediction method. Fig 4 shows the complete result. In the original kNN, the result is based on the average value of k neighbors and it does not consider the distance between the next moment's traffic condition and all its k neighbors. After applying the GPR, it considers the distance in the covariance, so the prediction can be more reliable in general. Moreover, if we consider the spatial and temporal information as the features in the kNN computation, the result can be further improved.

5) Running Time Result: In this experiment, we prepare different size of historical data. These historical data are extracted from different number of sensors. Our kNN with GPR method is executed using either non-`MapReduce` Java code run in a single machine or the `Hadoop MapReduce` architecture. The single machine for running non-`MapReduce` Java code is booted in Windows Server 2012 R2, and it contains 32 CPU and 128 GB memory. The other one is the computing cluster (Table II). In each system environment, we run the proposed kNN and GPR combined method five times and calculate the average running time in each test data. Fig 5 shows the running time result. In this result, `Hadoop` can dramatically reduce the running time. If the input data is larger, performance will be more significant. The maximum reducing time is around 69%. In fact, we can further reduce the running time by tuning the `Hadoop` scheduling procedure because what we face here are very similar prediction tasks through time. We shall be able to take advantage of knowing what is coming for the computation in the `Hadoop` environment.

C. *Further Discussion*

To predict traffic speed, traffic congestion is not the only event on the freeway that we should consider. Many incidents frequently happened on the freeway such as traffic collision, vehicle fire, animal in traffic, and so on may also significantly degrade traffic capacity. The RDE dataset contains the incident report from California Highway Patrol (CHP). Those incidents may block the traffic or some people can get trouble on the freeway. Given the same traffic data, we can also detect incidents that is considered a side product of the proposed speed prediction method. To give a concrete example, we observe a collision accident happened between a sensor s_i and another sensor s_{i+1} at 13:10 of February 15, 2010. The driving direction follows the order of sensors $s_1, s_2, \dots, s_i, s_{i+1} \dots$, with the increasing indices. We do observe the influence such as the upstream sensors, $s_1, s_2, \dots, s_{i+1}$ detected significant decreasing speed due to the traffic blockage while the downstream sensors $s_{i+2}, s_{i+3}, \dots$, remained the similar speed on the freeway (figure not shown). We believe that further systematical study can provide a reliable solution for incident detection for policemen patrols.

VI. Conclusion

We proposed a big data approach for traffic speed forecasting on an Intelligent Transportation System. We combine kNN and Gaussian process regression based on the `MapReduce` architecture. First, we select k nearest neighbors from the historical data to the next moment predicted traffic data. After that, GPR provides reliable speed prediction with confidence level for possible decision making done afterwards. The proposed method can keep updating the forecasting model based on newly acquired data and can give the most up-to-date prediction. We implemented all the above algorithms on the `MapReduce` framework with a mechanism that can learn to improve the `MapReduce` computation as time goes by. The experiments show that the average forecasting error is smaller than 2 miles/hr at most cases. Compared to a non-`MapReduce` method, we can achieve as much as 69% reduction in the computation time by the `Hadoop` architecture.

Acknowledgment

This work received grant from National Science Council under the grant number NSC 102-2221-E-011-122; also from National Science Council, National Taiwan University and Intel Corporation under grant numbers NSC 102-2911-I-002-001 and NTU 103R7501. This work is also conducted under the "III Innovative and Prospective Technologies Project" of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of the Republic of China. The authors also acknowledge anonymous referees for their constructive criticisms.

References

[1] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 4, pp. 1624–1639, 2011.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, 1 2006.

[4] Y. Saatçi, "Scalable Inference for Structured Gaussian Process Models," Ph.D. dissertation, St. Edmund's College, University of Cambridge, 2011.

[5] A. M. Rashid, S. K. Lam, A. LaPitz, G. Karypis, and J. Riedl, "Towards a scalable kNN CF algorithm: Exploring effective applications of clustering," in *Advances in Web Mining and Web Usage Analysis*. Springer, 2007, pp. 147–166.

[6] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction." *Science*, vol. 290, no. 5500, pp. 2319–2323, December 2000.

[7] M. Batko, C. Gennaro, and P. Zezula, "A scalable nearest neighbor search in p2p systems," in *Databases, information systems, and peer-to-peer computing*. Springer, 2005, pp. 79–92.

[8] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, Sep. 1977.

[9] V. Garcia, E. Debreuve, and M. Barlaud, "Fast k nearest neighbor search using gpu," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, June 2008, pp. 1–6.

[10] K. Zhou, Q. Hou, R. Wang, and B. Guo, "Real-time kd-tree construction on graphics hardware," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5. ACM, 2008, p. 126.

[11] H. Xiao and C. Eckert, "Lazy gaussian process committee for real-time online regression." in *AAAI*, 2013.

[12] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[13] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 4th ed. Wiley, 2008.

[14] Y. Wang and M. Papageorgiou, "Real-time freeway traffic state estimation based on extended kalman filter: a general approach," *Transportation Research Part B: Methodological*, vol. 39, no. 2, pp. 141–167, 2005.

[15] L. Mihaylova, R. Boel, and A. Hegyi, "Freeway traffic estimation within particle filtering framework," *Automatica*, vol. 43, no. 2, pp. 290–300, 2007.

[16] C. Chen, Z. Liu, W.-H. Lin, S. Li, and K. Wang, "Distributed modeling in a mapreduce framework for data-driven traffic flow forecasting," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, no. 1, pp. 22–33, 2013.

[17] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[18] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *NIPS*, 1996, pp. 155–161.

[19] "Research data exchange," https://www.its-rde.net/home.

[20] "Caltrans performance measurement system," http://pems.dot.ca.gov/.