Multi-models machine learning methods for traffic flow estimation from Floating Car Data

Jinjian Li, Guillaume Lozenguez, Jacques Boonaert, Arnaud Doniec

Available online at www.sciencedirect.com
ScienceDirect
Transportation Research Part C 00 (2021) 1–24
Journal Logo

Multi-models Machine Learning Methods for Traffic Flow Estimation from Floating Car Data

Jinjian Li, Jacques Boonaert, Arnaud Doniec, Guillaume Lozenguez
IMT Nord Europe, Institut Mines-Télécom, Univ. Lille, Centre for Digital Systems F-59000 Lille, France.
jinjian.li@{imt-lille-douai.fr, qq.com}, {arnaud.doniec, jacques.boonaert, guillaume.lozenguez}@imt-nord-europe.fr

Abstract

Traffic flow measurement is very important for traffic management systems. However, the existing traditional measurement approaches are highly time-consuming and expensive to continuously gather the required data and to maintain the corresponding equipment, such as loop detectors and video cameras. On the other hand, many services on the web propose to estimate automobile travel time taking into account traffic conditions thanks to crowd sourced data (Floating Car Data). This work proposes to reconstruct, from estimated travel time, traffic flows using machine learning method. In particular, we evaluate the capacity of Gaussian Process Regressor (GPR) to address this issue. After obtaining estimated travel time on a given route, a clustering process shows that travel duration profiles in each day can be associated to different "types of day". Then, different regressors are trained in order to estimate traffic flows from travel duration. In the "multi-model" variant, we trained a Regressor for each type of day. Conversely, in the "single model" variant, only one Regressor is trained (the type of day is not taken into account). This is an innovative work to estimate and reconstruct the traffic flow in transportation networks with machine learning method from aggregated Floating Car Data (FCD). A series of experiments are conducted to compare the estimated traffic flows, obtained by the proposed single model and multi-model, and the real ones from actual sensors. The obtained results show that both single model and multi-models can capture the tendency of real traffic flows. Furthermore, the performance can be improved by regulating parameters in GPR machine learning model, such as half width of sample window and sample size (a whole week or only weekdays), and multi-models can highly increase the performance compared with the single model. Therefore, the proposed GPR machine learning and FCD based new method can replace those traditional loop detectors for the measurement of traffic flow.

Keywords:
Estimation of traffic flows, simulation and modeling of transportation systems, Gaussian Process Regression (GPR), big data, machine learning, Floating Car Data (FCD)

1. Introduction

With the rapid growth of urban centers during the last decades, the development of efficient urban transportation services has become a central issue to reduce the high wasted time during the daily commute. The resulting increasing demand in terms of transportation flows has to cope with the difficulty to adapt existing or create new transportation networks.

In this context, simulate daily transportation behaviors allows operators to experiment and to visualize decisions about infrastructure and regulation policies. One of the major basics on efficient simulation relies on the ability to produce models representing the way that transportation flows evolve with time, depending on the traffic demands and events that impact the transportation network. The estimation of traffic flow is one of the core requirements in those simulation. One of the costless solution would be to reconstruct the traffic flow from aggregated information (travel duration estimations), available on web services.

Previous work validates that machine learning based approach is a promising way to reconstruct "sensor like traffic flow data" from aggregated information like the ones proposed by Google services [1, 2]. Applied such an approach would permit, for instance, to infer on a realistic traffic demand at each entrance of a city (i.e. the flow of incoming vehicles).

Machine learning over aggregated information approach is based on accessible databases that can provide information regarding the transportation condition (travel duration) at a given location and at a given time. In 2007, the Google company has extended Google Maps by adding Google Live Traffic, the visualization of traffic information in real time[3][4]. Here, the notion of real time means the current state and is applied to qualify the service of FCD provided. In more detail, Google exploits users' position data of Android smart-phones, in order to get a significantly fast and accurate mapping of the traffic. This data is called Floating Car Data (FCD), which can also be collected by any localization system embedded in a car and sent to the service provider via a mobile connection. Generally, these raw Floating Car Data are aggregated to provide more intelligible and relevant information regarding traffic condition. For example, in Google Maps, FCD are used to give a real-time traffic information using colored road section[1] which is determined by the navigation system or, as in the case of Google Live Traffic, by the smart phone and is sent to the service provider via a mobile phone connection. Therefore this allows the generation of real-time traffic information, which is visualized by the colors on Google Maps: red road points are related to a traffic jam or stop-and-go traffic, orange indicates heavy traffic and green points correspond to clear roads. However, those platforms generally provide only aggregated data like average travel duration more than the initial raw data.

Such an approach would permit operators to provide efficient global information while limiting the effort in continuously measuring road traffic flows based on physical sensors (radars, induction loops, etc.). The number of sensors, even in mid-sized cities, can increase very quickly. For example, to measure the input and output flows of a simple 4-points roundabout, at least 8 physical sensors are required. Therefore, such an expensive traffic flow measurement method makes the mentioned simulation process out of reach.

Preliminary result was published on the 15th World Conference on Transport Research 2019 [1] and on the 6th International Conference on Control Decision and Information Technologies 2019 [2] where traffic flows is estimated according to Floating Car Data (FCD) from Google Maps only on the basis of regressors trained using machine learning techniques instead of using stationary physical equipment (such as loop detectors [5] or video cameras [6]). In this paper, we present an extension of the previous works with an increased experiment setup that permits to automatize the model definition. Firstly, we show experimentally that, among 19 types of regression methods (including Linear Regression Models, Regression Trees, Support Vector Machines and Ensemble of Trees), the Gaussian Process Regression (GPR) is the most suitable machine learning method to obtain the best fitting criterion with respect to our dataset. Secondly, a selection of the adequate regressor is computed from a set of regressors (multi-model) to estimate traffic flows from FCD, based on the different types of travel duration profiles. This multi-model approach can greatly reduce the estimation error, by precisely clustering days presenting different types of travel duration profiles. Experiments are conducted by comparing estimated flows with real ones provided from induction loop sensors. The results we obtain seem promising enough to say that correct transportation flows models could be obtained with a very light use of real traffic sensors.

This paper is organized as follows: the next section describes related works and the different usages of aggregated FCD in the context of transportation networks modeling. The third section focuses on the problem we choose to address that is building sensor like data flow measurements from aggregated data. In this section, we also provide details regarding our problem formulation on the standpoint we took for solving it. The fourth section presents the single and multi-GPR machine learning method for the estimation of traffic volume. The fifth section deals with the experimental site, the results we obtained, the comparison between estimated traffic flow and real observed data prior to the discussion. The last section concludes this paper and presents some perspectives and further works based on it.

2. Related Work

The successful wide scale deployment of the Advanced Traveler Information Systems (ATIS) and Advanced Traffic Management Systems (ATMS) highly relies on the capability to perform accurate estimation of the real traffic

[1] 4 colors are available: green for normal speed of traffic, orange for slower conditions, red for congestion and dark red for stopped traffic

states on road networks. Therefore, the use of real-time Floating Car Data (FCD), based on traces of Global Positioning System (GPS) positions of vehicles, is emerging as a reliable and cost-effective way to collect accurate traffic data for a wide area road network. Unlike other traffic data collection techniques (e.g. traffic cameras, induction loops embedded in the roadway, radar-based sensors), floating cars act as moving sensors traveling in a traffic stream and do not require additional instrumentation to be set up on the roadway.

The main communication architecture for FCD is based on the exchange of information between a fleet of floating cars traveling on a road network and a central data operation system. The floating cars periodically send their positions (latitude, longitude and altitude) and instantaneous velocity thanks to GPS receiver and Global System for Mobile communications (GSM) or General Packet Radio Service (GPRS) transmitter. While the central data operation system tracks the received FCD along the traveled path by matching the related trajectory data to the corresponding real road network. The frequency of sending/reporting is generally determined by the required resolution of the data and the performances of the available communication channels, for example, bandwidth. Therefore, FCD is an effective approach to determine the real traffic speed on the road network, based on gathering of data such as localization, speed, direction of travel and time information from the mobile phones of drivers and passengers of the vehicle. That is to say, every vehicle with an active mobile phone acts as a sensor for the network.

Using FCD for estimating travel duration and traffic states has received high attention over the years from the scientific community. The most common and useful information provided by FCD is travel-duration and speeds along road links or paths [7, 8, 9]. Works such as [10, 11, 12] calculate or forecast the travel duration on roadside according to the speed of the floating car on that road. Event detection, such as congestion and accidents has been discussed by using the trajectories or speed of floating cars [13, 14, 15]. The percentage of floating cars required for the estimation of travel duration is presented in the works like [16, 17, 18]. Reconstructing the traffic states from FCD has been previously addressed in papers such as [17, 19, 20, 21].

In [19], the authors attempt to estimate and predict the travel speed with the real-time FCD based on traces of GPS positions. Another work like [22], both spatial and temporal characteristics to the domain of floating car sampling is introduced. And the analysis of this work can provide an insight for floating car based traffic state system designers on the transmitting period, sampling interval and penetration of floating car that are desirable in a traffic network in order to get certain coverage and accuracy in traffic state estimation. An interesting method is proposed in [20], whose purpose is to obtain a high quality on the reconstruction of travel times in the net with a smaller percentage of FCD vehicles and number of FCD messages. The most recent research about traffic state reconstruction using FCD is presented in work [21]. The speed is estimated based on FCD. Then the authors make use of the R^2 *Statistic* to build the function between average speed ($\overline{V}$) and density (ρ). Finally, the traffic flow is calculated according to the Fundamental diagram ($Q = \rho \cdot \overline{V}$).

In summary, the above works mainly deal with the following subjects: 1) strategy to collect FCD, 2) estimation of traffic state by building traditional function model. However, in this work, on the one hand, it is a pragmatic choice to apply the aggregated FCD in Google Maps, because it is the most widespread and best-known system from a more practical and industrial point of view, and it can provide lots of data in a complete transportation network considering scientific aspects. Other FCD providers (such as Uber) could have been employed, but the quality of the collected data is very similar while offering a much lower coverage than the Google-based services[23]. Moreover, accessing the data sources (programming constraints) from the Google API is almost straight straightforward and free of charge (for a given number of requests per month). On the other hand, traffic flows are estimated according to FCD from Google Maps only, on the basis of regressors trained using machine learning techniques, instead of using traditional stationary devices (such as loop detectors [5] or video cameras [6]). To the best of our knowledge, this is an innovative approach that could help minimize the use of stationary sensors while providing a good enough estimation of traffic flows.

3. Problem Description and Proposed Mathematical Model

This section firstly describes the problem addressed in the work. Next step presents the proposed system structure, where two types of regressors based models are introduced, including single model and multi-model. Then the feature extraction method is shown. At last, the criteria used to evaluate the proposed system's performance are presented.

3.1. Problem description

The problem is the estimation of traffic flow on a lane based on travel duration estimated by Google Maps thanks to FCD, as illustrated in Fig. 1. We consider a road section of a certain length (1.2 km in the example of Fig. 1), we measure the traffic flow on this section thanks to a physical sensor (in the example of Fig. 1 data were collected using a Doppler radar (24.165 GHz / 100mw EIRP)) located in the middle of the section and in parallel we estimate the travel time on this section thanks to Google Maps. The objective is to find the relationship between this travel time and the vehicle flow measured on the ground.

According to the Fundamental diagram of traffic flow theory, travels duration and traffic flows are linked with each other, and their nonlinear relationship can be traditionally formulated with some special statistical model-based methods based on many assumptions, with lots of parameters to be tuned[24][25]. Several examples of such method are Van-Aerde-Function and exponential model [26]. The parameters to be tuned can be mean travel time at zero and real flow, some other parameters to be estimated from empirical data, etc. However, it would be too complicated with too many parameters to be tuned to use a general statistical model-based method, with no guarantee of efficiency. Therefore, this work tries to directly establish such a relationship on the basis of machine learning methods. Because machine-learning based approach can learn a general black model without the knowledge of a 'Physique Model' of the dynamic system, making it much more convenient to be propagated on all over 'similar' road sections.

Figure 1: Estimation of traffic flow based on FCD (travel duration) from Google with machine learning method. It is assumed in this article that traffic flows should be uniform along the chosen road

3.2. Proposed system structure

The proposed system structure consists of three parts: data collection, machine learning model training and trained model testing, as illustrated in Fig.2. For data collection, traffic flows are measured by a sensor over a specified and limited period of time , which is 26 weeks in this work. Over this same period of time, We retrieve directly travels durations, which are calculated by Google map servers with a complicated algorithm based on the collected floating car data. In other words, through the Google maps API, the travel duration can be obtained by giving the GPS for origin and destination of the chosen road segment, and the time-stamp, as shown in the Tab. 1. The GPS for origin and destination is got manually from the Google Map on the chosen road. The time-stamp is a serial of numbers to define a certain instantaneous time elapsed since 1970, January 1_{st}, 0 hours, 0 minutes, 0 seconds. The request can be done either on a Web browser or by Python programming. The travel duration data is captured each ten minutes from Google-map. In total, 26208 data are retrieved. Then the obtained traffic flows data and their corresponding travels durations are split into two sets: a training dataset with 50 percent for the machine learning model training, and a validation dataset with another 50 percent for the trained model testing. For the training of machine learning model, the estimation of traffic flows from travels durations is treated using regression models. Three types of models are proposed: single model, manual multi-model, and K-means multi-model (refers to 3.3). The machine learning algorithm applied is the training of a Gaussian Processes Regressors (GPR) (refers to 4.5). This corresponds to the red rectangle part in Fig.2, that is presented more detailedly in the next section (refers to 3.3). At the end of the learning process, trained GPR models are obtained, which can take travels durations as input to estimate traffic flows as output, without requiring any additional "in site" flow sensors anymore. The estimated traffic flows are compared with the one measured by real sensors to evaluate the performance of trained GPR models.

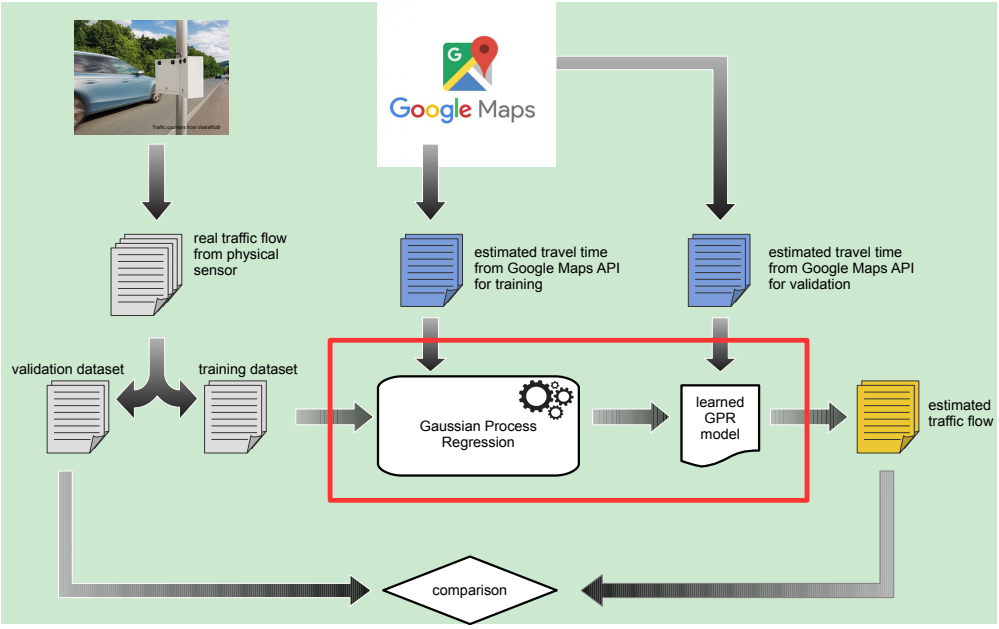

Figure 2: Proposed system structure

Table 1: Input and output data structure for requesting FCD with Google Maps API. For example, the 310 seconds in the output column means that it takes average 310 seconds for cars to pass from origin point GPS to the destination point GPS on the road at the given time stamp.

Example	Input			Output
	GPS of origin	GPS of destination	Time-stamp	Traffic duration (s)
1	(50.372329,3.070058)	(50.380205,3.082129)	1520677640	310
	Road name: Boulevard de la Republique,Douai,France		(10/03/2018;10:27:20)	
2	(50.380163,3.079186)	(50.390721,3.081124)	1521306610	370
	Road name: Boulevard Lahure,Douai,France		(17/03/2018;17:10:10)	

3.3. Single model and multi-models proposed for traffic flows estimation

The traffic flows evolve over the days of a week, because of the difference of traffic demand and supply between working day and weekend. Therefore, using days' specific GPR models (multi-models approach) instead of a "generic" one (single model approach) may provide better estimations of the traffic flows along the week. In the manual multi-models, three GPR models are trained for weekdays, Saturday and Sunday in training data set.

As a prior step, we also have to verify that days of the week can be clustered on the basis of the travel durations profiles collected by Google Map. Here, the profile associated to a day is simply the sequence of travel durations measurements along this day. To do so, we apply a Principal Components Analysis (PCA), which is a method to reduce the dimension of a feature vector while keeping a significant part of its original information (refers to 4.2). The classical k-means algorithm is further applied to this reduced feature space (refers to 4.3) to build clusters that are interpreted as different types of days. On the one hand, each of these clusters will, in turn, be used to train its own GPR model. On the other hand, the label results from k-means are applied to train a Support Vector Machine (SVM) model, which is used to cluster a new day (testing data sets) to choose the suitable trained GPR model from its corresponding FCD profile. Three provided strategies to build models are illustrated in Figs.3 and 4:

1. Single model: only one GPR model is trained using all the training data (variations of FCD profiles between the days are not taken into account during the GPR's training process, namely, all the training days are applied to the same regression model)[1].

2. Manual multi-model: three different GPR models are trained (one for days of the week except Saturday and

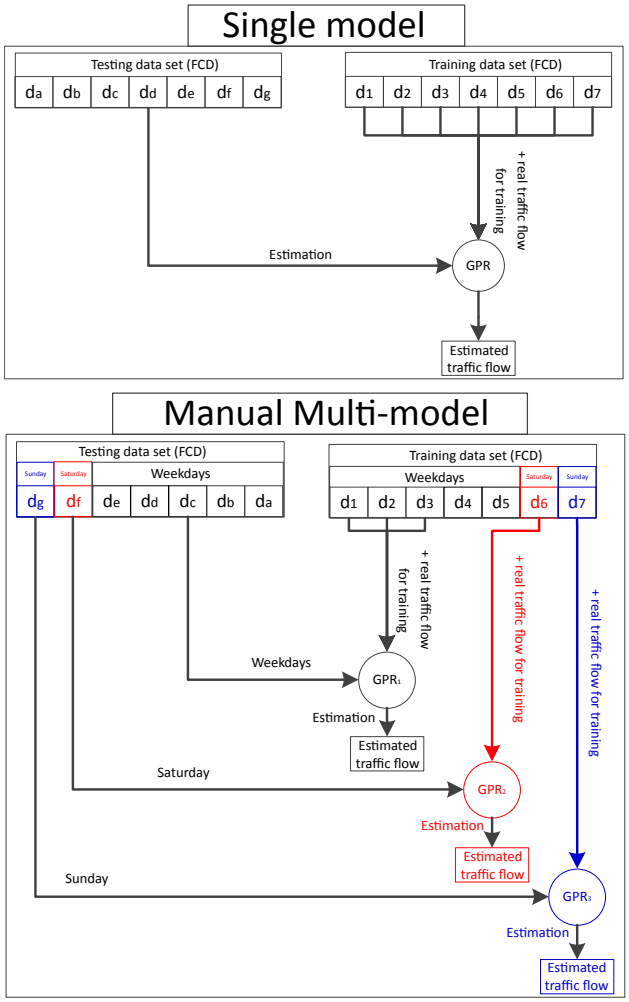

Figure 3: Flowchart for Single model and manual multi-models (an example for three clusters on training and testing data sets for a week, respectively)

Sunday, one for Saturdays and one for Sundays, respectively), to model the fact that, generally speaking, the traffic flows seem to be similar during ”regular” days of the week, and are different during Saturday and during Sunday. The term ”manual selection” stands for the fact this partition is only based on the name of the day, without statistical analysis[2].

3. K-means multi-models: several different GPR models are trained based on the clustering results from k-means, in order to model the fact that, generally speaking, the FCD profiles clustered by k-means should have similar traffic flows. This K-means multi-models are new content compared with the above two types of models published on two International Conferences. Therefore, performing the clustering process using FCD or using flow data leads to two compatible partitions of the type of days.

3.4. Features extraction method

The feature extraction method is illustrated in Fig.5. This paper address a regression problem but not a prediction problem. Then, for estimating traffic flow at time step k, ”past” and ”future” travel duration samples can be taken into account. Consequently, the feature extraction process is based on the assumption that the traffic flow f_k at the given time step k can be approximated from a samples window with $(2 \cdot n + 1)$ width centered on travel durations d_k at time step k, where, the notations are defined as follows:

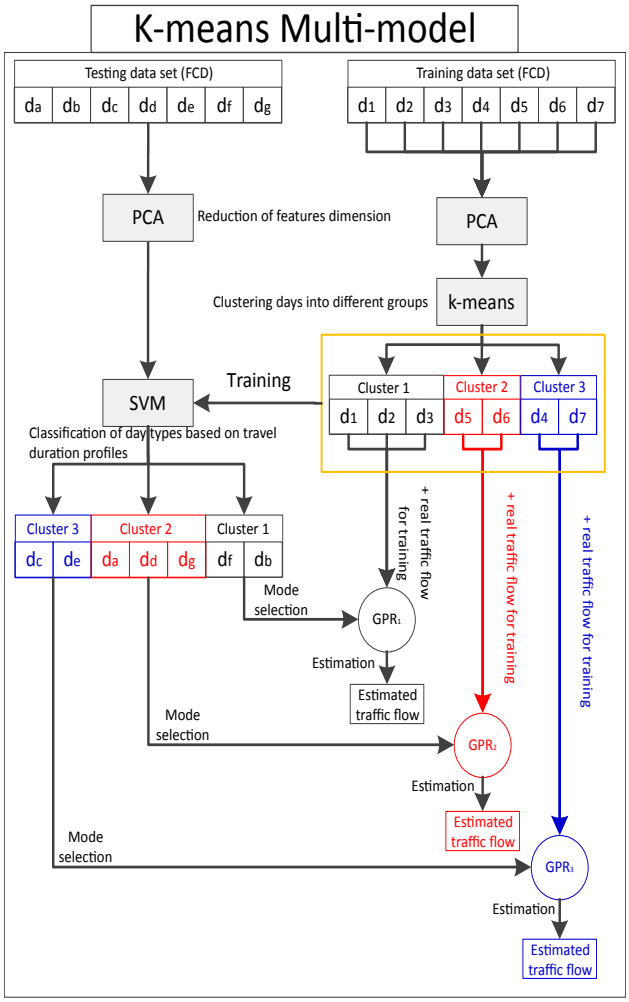

Figure 4: Flowchart for K-means multi-models (an example for three clusters on training and testing data sets for a week)

1. f_k and d_k: The variable $f(t)$ is defined as the actual traffic flow (in *veh/h*) measured by a traditional sensor (such as an inductive loop detector) on a given lane at time t, while $d(t)$ is the corresponding travel durations (in s) estimated by Google for passing the same given lane at the same instant t. In practice, traffic flow data $f(t)$ is sampled at a period T_{flow} corresponding to one hour with traditional sensors by the urban terrestrial transportation network management service of the city of Douai in France. However the travel duration data $d(t)$ is sampled from Google Maps at a period T_e, which is equal to ten minutes, in order to get more information about the travel duration. To cope with these different sampling periods, traffic flow data are linearly interpolated between two conservative values of the traffic flows, as shown in the Fig.5. As a result, both $d(t)$ and $f(t)$ are considered to be sampled with sample period T_e. Therefore, for the sake of simplicity, in the following, we denote $d(k \cdot T_e). \equiv d_k$ and $f(k \cdot T_e). \equiv f_k$, which means that the travel duration and travel flow can be represented as d_k and f_k, respectively, for k^{th} sample at time $k \cdot T_e$.

2. n: The variable n ($n \in N^*$) refers to the half width of the samples window acquired from $d(t)$, centered at time step k, as illustrated by the purple rectangle drawn in Fig.5.

Therefore, the input vector (or extracted features) X_k of the GPR is $X_k = \{d_{k-n} \ldots d_{k-1}\ d_k\ d_{k+1} \ldots d_{k+n}\}$. The output value is the traffic flow f_k. Alternatively stated, the traffic flow f_k should be estimated by the GPR model from the extracted features X_k.

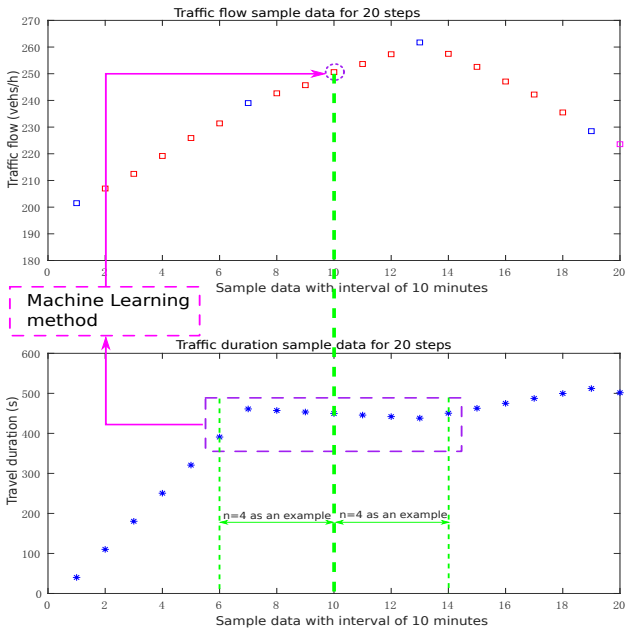

Figure 5: Features extraction model

4. Theory of Machine Learning Methods Applied

This section firstly presents the total flowchart of proposed traffic flow estimation system. Then the theory is researched for the Machine Learning related methods applied in this work, as shown in the Figs.3 and 4.

4.1. Innovation and main steps of the proposed traffic flow estimation model

The innovation of the proposed model is that we apply a Machine Learning based model to rebuild the relationship between easily accessible travel duration and costly measured traffic flow, instead of applying the traditional formulation with lots of parameters to be tuned. Moreover, the proposed model can automatically classify the training data set into different types to train several Regression models to improve the performance, compared to single Regression model, and the testing day can also be classified into the most suitable Regression model because of a trained classifier, in order to reduce the estimation error percentage. This is the first proposal compared to all the existed literatures for the problem of traffic flow estimation from FCD. Therefore, the following three subsections present the main steps of the proposed traffic flow estimation model by combining the proposed system structure in the Fig.2 and the different models flowchart in the Figs.3 and 4.

4.1.1. Main steps for traffic flow estimation with single model

- Request of FCD from Google Maps API and corresponding traffic flow from transportation management;
- Construction of input vector X_k based on the selected half width of sample window n, as presented in the subsection 3.4;
- Division of all FCD and traffic flow data into training data set and testing data set according to a certain proportion;
- Application of training data set to train 19 types of Machine Learning methods, as shown in the Fig.6;
- Application of testing data set on the trained Machine Learning models to get estimated traffic flows;
- Comparison of the estimated traffic flow and the real one.

4.1.2. Main steps for traffic flow estimation with manual multi-models

- Request of FCD from Google Maps API and corresponding traffic flow from transportation management;

- Division of all FCD and traffic flow data into three groups of training data set and testing data set according to weekdays, Saturday and Sunday;
- Construction of input vector X_k based on the selected half width of sample window n, as presented in the subsection 3.4, for training and testing data set respectively;
- Application of three groups of training data set to three different GPR models respectively, as shown in Fig.6;
- Application of three groups of testing data set on the corresponding trained Machine Learning models according to weekdays, Saturday and Sunday, in order to get estimated traffic flows;
- Comparison of the estimated traffic flow and real one.

4.1.3. Main steps for traffic flow estimation with K-means multi-models

- Request of FCD from Google Maps API and corresponding traffic flow from transportation management;
- Division of all FCD and traffic flow data into training data set and testing data set with day as unit;
- Combination of all the FCD data within the same day in the training data set as input vector for PCA;
- Application of K-means methods to groups all days in the training data set into three different groups;
- Construction of input vector X_k based on the selected half width of sample window n, as presented in the subsection 3.4, in the three different groups separately;
- Application of three groups of training data set to three different GPR models respectively, as shown in Fig.6;
- Combination of all the FCD data within the same day in the testing data set as input vector for PCA;
- A classifier SVM trained based on the K-means results in the above step 6, in order to classify the testing data into suitable group;
- Application of corresponding trained Machine Learning models in each group on testing data set to get estimated traffic flow;
- Comparison of the estimated traffic flow and real one.

4.2. Principal Components Analysis (PCA) for reduction of the selected features space dimension

Principal Component Analysis (PCA) is the general name for a technique in multivariate data analysis aimed to reduce the number of dimensions, while keeping as much as possible of the data's variation [27, 28]. Instead of researching thousands of original variables, the first few components built from a linear combination of the original features and containing the majority of the data's variation are explored. The statistical analysis and visualization of these new variables, named the principal components, can assist to find similarities and differences between samples. Important original variables that are the major contributors to the first few components can also be discovered. More precisely, PCA applies a vector space transformation to reduce the dimensionality of large data sets. By using mathematical projections, the original data set, which may have involved a great deal of variables, can often be interpreted in just a few variables (named the principal components). Therefore, it is often the case that an examination of the reduced dimensional data set will allow the user to spot trends, patterns and outliers in the data, much more easily than without performing this principal component analysis. Therefore, in this paper, PCA is applied to reduce the features dimensions of the FCD profiles in order to facilitate the k-means clustering.

4.3. Clustering method K-means to dispatch all tested days into different clusters

The k-means is an unsupervised clustering algorithm applied to find groups within the data [29, 30]. Given a set of observations (x_1, x_2, ..., x_n), where each observation is a d-dimensional vector, k-means clustering algorithm aims to divide the n observations into a set of k groups ($k \leq n$), such as $G = G_1$, G_2, ..., G_k, in order to minimize the within-group sum of distance squares, which is defined as the sum of distance functions of each point in the group to the corresponding center. The objective function of k-means is the following, where, c_i means the centroid of points in group G_i. Therefore, the k-means clustering algorithm can be used to cluster the days based on the FCD profiles. In this work, we choose the value of k based on the daily life's observation of our city in France. Typically, we have three types of day: working days, Saturdays and Sundays. During classical working days, people need to go to work, etc. Saturdays are characterized by less work and open shops. Sundays, generally, few people go to work and the possible activities are more restricted (most of the shops are closed for instance).

$$\underset{G}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in G_i} \|x_i - c_i\|^2 \tag{1}$$

4.4. Classification method SVM

SVM is a typical machine learning algorithm for classification problem, which was originally introduced by Vapnik and co-workers [31][32] and successively extended by plenty of other researchers. It can have a remarkably robust performance with respect to sparse and noisy data, which makes it useful in a good deal of applications from text categorization to protein function prediction. In particular, for the classification problem, it separates a given set of binary labeled training data with a hyperplane, which is maximally distant from training data sets (also known as 'the maximal margin hyperplane'). If no linear separation is possible, it also can work by combining with the technique of 'kernels' function, which can automatically realize a non-linear mapping to a feature space. In the end, the hyperplane found by the SVM in feature space corresponds to a non-linear decision boundary in the input space. For a more detailed presentation about SVM, interested readers can refer to [32] for more information.

4.5. Regression machine learning methods for traffic volume estimation based on the FCD

This section presents the description of regression machine learning methods for the estimation of traffic flow based on travel duration from Google Maps, as shown in Fig.6.

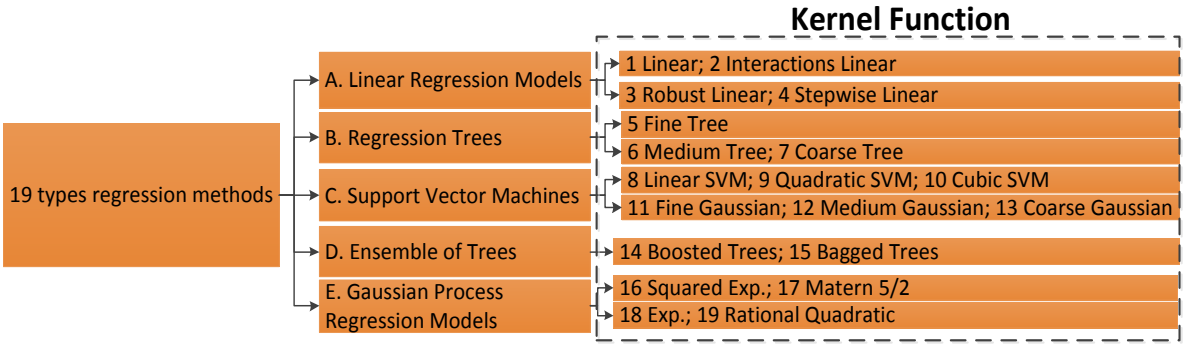

Figure 6: Machine learning models for regression with different kernel functions tested in this article (in total 19 types)

4.5.1. Linear regression models

These models describe a linear relationship between an output and one or more input. Such model has the following characteristics: 1)the response (output) has a normal distribution with mean Y, for a set of predictors or inputs, which is named as X; 2)a coefficient vector b is defined and linearly combined with the predictors X; 3)the linear regression model is Y = Xb. The classical hyperparameters to tune when using linear regression models are the learning rate and the number of interactions. We can also choose to perform robust fitting using a set of weighting functions to cope with outliers or to consider non-Gaussian noise affecting the data.

4.5.2. Regression trees models

These models can give numeric responses based on input data. In order to predict a response, the decisions in the tree from the root node down to a leaf node should be followed, because each leaf node contains a response. The trees applied are binary, which means that only one predictor (variable) is checked in each step of prediction. The hyperparameters for regression trees allow to obtain different trees forms restricting the maximum depth of trees, defining the minimum data points to split a node, fixing the number of samples required to consider node as a leaf, etc.

4.5.3. Support Vector Machine (SVM) for regression

This model is identified by Vladimir Vapnik and his colleagues [33], is a very popular machine learning tool for regression. The SVM regression is a nonparametric technology, since it relies on kernel functions. In the Matlab Regression learner APP, the linear epsilon-insensitive SVM (ϵ-SVM) regression is applied. In ϵ-SVM regression, predictor variables (X) and observed response values (Y) are included in the training data set, and the goal is to find a function f(X) that deviates from Y (observed response values) by a value no greater than ϵ (error or deviation) for each training point X, as shown in Eq.(2). And the function f(X) should be as flat as possible.

$$|f(X) - Y| \leq \epsilon \tag{2}$$

In addition to the choice of the kernel function (linear, quadratic, cubic, gaussian, etc.) two hyperparameters are mainly used to improve performance of SVM: ϵ parameter defining the width (margin) of the zone used to fit the training data and a factor, corresponding to a "box constraint" that tunes the cost of deviations larger than ϵ.

4.5.4. An ensemble of trees model

This model is a predictive model composed of a weighted combination of multiple regression trees. In general, combining multiple regression trees increases predictive performance. This means that results from many weak learners can be melded into one high-quality ensemble predictor. Here we find the same parameters as for the decision trees (maximum depth of each tree, the minimum data to split a node, etc) and some extra hyperparameters like the number of trees or the weighted combination of trees.

4.5.5. Gaussian Process Regression (GPR)

As we can see in the experimental results in the section 5 that Gaussian Process Regression (GPR) can help us to have good performance for the estimation of traffic flow. Therefore the GPR is introduced more detailedly as following. Firstly, the background of GPR machine learning method is introduced[34]. Then the algorithm of applying the GPR for the estimation of traffic flow is explained.

The GPR [34][35] is a supervised machine learning method that offers mapping function between input and (continuous) output data. The Gaussian Process framework is used in different areas extended from classification to regression problems, including speed estimation [36], travel duration estimation [37], time-varying systems [38] etc. In this work, a GPR model was adopted to model and to estimate traffic volume from the Google aggregated FCD. Because the traffic volume is expected to have some complex relationship with the travel duration on the same road, simple parametric models such as linear or polynomial functions is inappropriate for this task[24][25]. Therefore, in this subsection, firstly, Gaussian Processes (GPs) formulation is presented. Then, based on the GPs formulation, the Gaussian Process Regression machine learning method is introduced.

Generally, GPs offer a Bayesian paradigm to learn an implicit functional relationship $\hat{y} = \hat{f}(x)$, according to a given training data set, $D = \{(X_i^d, y_i)|i \in N\}$. The variable N is the size of the data set. The symbol X_i^d represents a vector for the i^{th} observed input variable (also named predictor, regressor, control, or independent) in a d-dimensional feature space. And y_i is a one-dimensional observed target value (also named predicted, regresse, response, or dependent), which is either continuous or discrete. However, unlike most classical Bayesian models [39], GPs directly infer a prior distribution on the whole function $\hat{f}(X)$. Thus, function $\hat{f}(X)$ is treated as a random field and is assumed to be a GP a prior, as the Eq.(3) shows.

$$p(\hat{f}(X)|\theta) \propto GP(m(X), k(X, X')) \tag{3}$$

where, the prior GP is fully defined by a mean function $m(X)$ and a covariance function $k(X, X')$. The notation θ means the prior's hyperparameters applied to parameterize the covariance function, as follows:

$$K(X, X') = K(X, X'; \theta) \tag{4}$$

Strictly speaking, a GP model can also be treated as a probability distribution, which is defined over the following functions:

$$E[\hat{f}(X)] = m(X) \tag{5}$$

$$Cov[\hat{f}(X), \hat{f}(X')] = k(X, X') \tag{6}$$

where $\hat{f}(X), \hat{f}(X')$ are random variables that are indexed by any pair of X and X'. Then, a GP prior can be roughly considered as a probability distribution for an infinite number of random variables. Furthermore, a collection of function values, which are indexed by any finite number of $X = [x_1, x_2, ..., x^n]^T$, e.g., $F(X) = [f(x_1), f(x_2), ..., f(x_n)]$, supposes a multivariate normal distribution in Eq.(7)

$$p(\hat{f}(X)|) = N(m(X), K(X, X')) \tag{7}$$

where the average vector $m(X)$ and covariance matrix $K(X, X')$ are determined directly based on $m(\bullet)$ and $k(\bullet, \bullet)$, as following:

$$m(X) = [m(x_1), m(x_2), ..., m(x_n)]^T \tag{8}$$

$$K_{i,j} = k(x_i, x_j) \qquad i, j = 1, ..., n \tag{9}$$

For the sake of simplicity and without loss of generality, $m = (x) = 0$ is assumed, since the data can always be centered by the sample mean.

With the machine learning term, $k(x, x')$ is often named as a **kernel function** or simply a kernel instead of a covariance function. As detailed later, kernel functions generally take certain forms which are parameterized by one or several parameters θ. Therefore, a GP prior can be specified by determining a specific type of kernel function (also named covariance function) and the associated θ values.

Once a GP prior $p(f|\theta)$ and a "noise" model $p(y|f)$ are determined, $p(f|D, \theta)$ the posterior distribution of f can be easily obtained by updating the prior $p(f|\theta)$ based on the Bayes theorem with the training data set D, as shown in Eq.(10)

$$p(f|D, \theta) = \frac{p(y|f)p(f|X, \theta)}{p(D|\theta)} \tag{10}$$

where the input variables X should be made explicit in the prior and term $p(D|\theta)$ is called Marginal Likelihood, since it is a function of variable θ and given data set D. The noise model $p(y|f)$ is also a likelihood, for the reason that it is a function of f for a fixed set of observations y. Here, the $p(y|f)$ is introduced, since y_i is a corrupted version of $f(x_i)$. Therefore, the estimation distribution for a new input x_{new} is achieved by using the Eq.(11) with the posterior $p(f|D, \theta)$

$$p(f_{new}|x_{new}, D, \theta) = \int p(f_{new}, f|D, \theta)df \tag{11}$$

By the combination with Eq.(11) and the noise model, the predictive distribution for y_{new} is achieved in the Eq.(12)

$$p(y_{new}|x_{new}, D, \theta) = \int p(y_{new}, f_{new}|D, \theta)df_{new} \tag{12}$$

From the Eq.(12), not only the estimated average values but also the associated uncertainty (error-bar) could be calculated. In the GP modeling, it is as collection of function values $f(x)$ needed to be Gaussian instead of variables x itself, which are assumed to be distribution-free. Therefore, the GP model theoretically can handle data with any kinds of distributions. For a more detailed presentation, interested readers can refer to [34][40] [41] for more information.

Gaussian Process Regression machine learning method is introduced based on the GPs formulation. The GP model presented in the above subsection can solve non-linear regression problems, if the observed target value y_i is continuous, and the noise model $p(y|f)$ is assumed as a normal distribution. Then the GPR model can be expressed in the Eq.(13).

$$y_i = f(x_i) + \theta_i \qquad \theta_i \sim N(0, \sigma^2) \tag{13}$$

In this case, the inference of GPR model becomes analytically tractable, as a result of the Gaussianity of $p(y|f)$. Accordingly, for a new input x_{new}, the predictive mean and variance associated with $\hat{f}_{new} = f(x_{new}) = f_{new}$ are defined in Eq.(14)-(15), respectively[34].

$$\mu(f_{new}) = k(x_{new}, X)[K(X, X') + \sigma^2 I]^{-1} y \tag{14}$$

and

$$Var(f_{new}) = k(x_{new}, x_{new}) - K(x_{new}, X)[K(X, X) + \sigma^2 I]^{-1} k(X, x_{new}) \tag{15}$$

where X and y mean the observed predictors and observe target value. I is defined as the identity matrix.

The main hyperparameter of GPR is σ the initial value for the noise standard deviation of the Gaussian process model.

5. Experiment and validation from the real data for single model

In this section, we conduct a series of experiments over two road segments to evaluate the proposed algorithm and compare the results with real data. Firstly, the performance is compared between estimated traffic flow and real data in the single model regarding data of a first road segment. Then the results between single model and multi-models are compared on another road segment.

5.1. Single model simulation case

The experiments are executed on a 1.2 km long road segment named "Boulevard de la République, Douai, 59500, France" with GPS of origin (50.372329,3.070058) and destination (50.380205,3.082129) for a duration of 26 weeks. Firstly, the protocol of our experiments are presented. Then, the comparison of RMSE among different machine learning regression methods is shown for n (half width of the sample window) from 2 to 24. Next, experimental performance during a whole week is presented in detail for the kernel function, named rational quadratic in GPR, because it can help us to achieve the lowest RMSE value, compared with others. Finally, an experiment with only weekdays, which performs better than the above case, is executed, for the reason that the profile of traffic flow exists big difference between weekdays and weekends.

5.1.1. Protocol and experiments description

The data of travel duration and real traffic flow on this road for a whole week from Monday to Sunday is respectively acquired from Google Maps, as shown in Fig.7 and town council of Douai in France, as Fig.8 shows. In total, 19 types of regression models are applied to find the best machine learning regression methods for such problem of estimating traffic flow from travel duration. The set of all the sample data is divided into two groups: 50 percent of sample data is used to train and validate the regression model; the remainder is extracted as new data to test the performance of the trained regression model. The 5-fold Cross Validation method is applied for the 19 types of regression machine learning algorithms to avoid overfitting problem. The n (half width of the sample window as shown in the Fig.5) locates in the zone [2,24]. Specifically, the sample time interval is from 20 to 240 minutes, since the sample step is 10 minutes. The unit for traffic flow is veh/h, travel duration s. Note that all the hyperparameters described in the presentation of each regression machine learning models have been automatically tuned by Matlab.

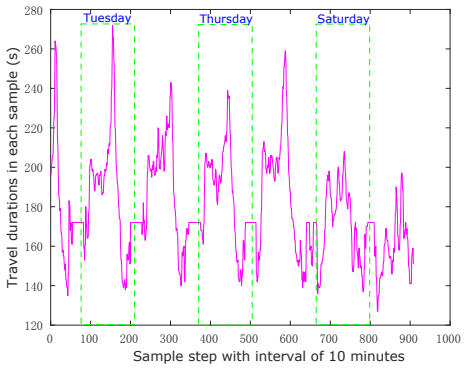

Figure 7: Travels durations from Google FCD along a week

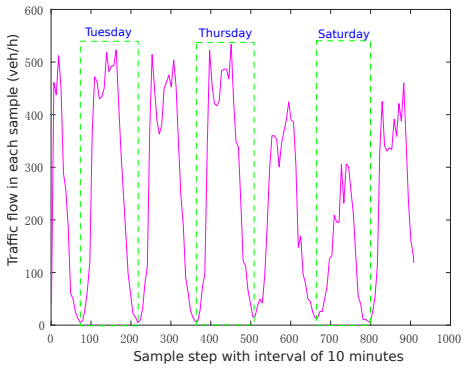

Figure 8: Traffic flows from actual sensors measurements along a week

5.1.2. *Evaluation criteria*

The evaluation criteria, containing Root Mean Square Error (RMSE) and Root-Mean-Square Deviation (RMSD), are applied to evaluate and to compare the performances of the two proposed strategies (single model and multi-model), because they are the classical criteria and the most used evaluation criteria for the sequences data regression problem while some authors suggest the joint use of the MAE ('Mean absolute Error')[42]. In addition, the estimation error distribution looks near from being Gaussian (refer to histogram in pages 16-17), which is a 'classical' use case for comparing models with MSE. Here, they are defined as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(f(k) - \hat{f}(k))^2} \tag{16}$$

$$AM = \frac{1}{N}\sum_{k=1}^{N}(f(k) - \hat{f}(k)) \tag{17}$$

$$RMSD = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N}((f(k) - \hat{f}(k)) - AM)^2} \tag{18}$$

where $f(k)$ is the observed traffic flow at time step k and $\hat{f}(k)$ is the corresponding estimated traffic flow. AM is the arithmetic mean. N is the total number of samples.

5.1.3. *Discussion about Machine Learning methods*

This subsection presents the comparison of RMSE for all the 19 machine learning regression models, as shown in Fig.10 and Tab. 2-3, in order to find the best solution, which can estimate the traffic flow from travel duration with lowest RMSE. The 19 regression modules are grouped in the Tab. 2-3 into five families:

A. Linear regression models. These models always get a very high RMSE, because the relationship between travel duration and traffic flow is nonlinear [24][25].

B. Regression trees. These models can achieve better performance than linear regression models with lower RMSE, for the regression trees can deal with nonliear system. The best result with lowest RMSE 70.206 happens when the n equals 24 under the Fine Tree (number 5 in the Tab. 2).

C. Support vector machines. The lowest RMSE for SVM is 59.811, which is obtained by the Fine Gaussian at n=24, and is 14.81 percent lower than that in the Fine Tree.

D. Ensemble of trees. Such models are better than the above regression trees by combining several trees together. The lowest RMSE (57.69) is acquired by the bagged trees when n is 24, and is 17.83 percent lower than that in the Fine Tree.

E. Gaussian process regression models. The global best result is the lowest RMSE with value of 18.938, which is achieved by Rational quadratic kernel function with n=23, is 67.17 percent lower than that in the bagged trees.

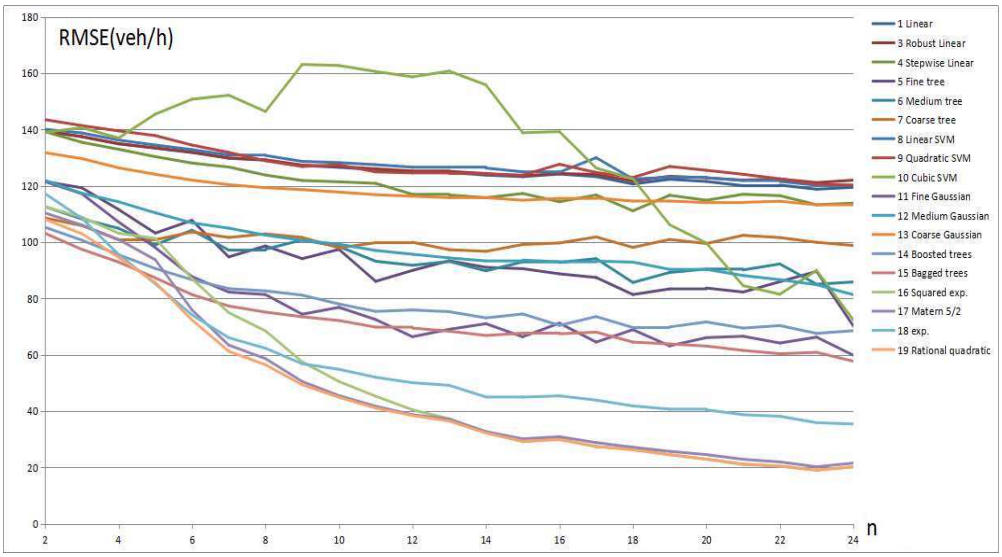

Figure 9: RMSE profile for regression models under n from 2 to 24

Table 2: RMSE value for types 1-10 of machine learning regression models under n from 2 to 24 (PE is an abbreviation of Percentage Error)

n	A. Linear Regression Models								B. Regression Trees						C. Support Vector Machines					
	1	PE(%)	2	PE(%)	3	PE(%)	4	PE(%)	5	PE(%)	6	PE(%)	7	PE(%)	8	PE(%)	9	PE(%)	10	PE(%)
2	139.34	57.89	139.82	58.09	139.4	57.92	139.2	57.83	121.49	50.48	112.54	46.76	108.56	45.10	140.04	58.18	143.47	59.61	138.91	57.71
3	137.39	57.04	137.86	57.24	137.47	57.07	135.43	56.23	119.27	49.52	108.24	44.94	105.85	43.95	138.75	57.60	141.35	58.68	140.63	58.39
4	134.86	55.97	138.73	57.57	134.96	56.01	132.95	55.18	111.58	46.31	104.88	43.53	100.8	41.83	136.23	56.54	139.48	57.89	136.94	56.83
5	133.35	55.34	137.39	57.01	133.43	55.37	130.33	54.09	103.18	42.82	99.005	41.09	100.81	41.83	134.44	55.79	137.8	57.19	145.42	60.35
6	131.79	54.71	135.9	56.41	131.95	54.77	128.05	53.15	107.71	44.71	104.24	43.27	103.61	43.01	132.83	55.14	134.44	55.81	150.72	62.57
7	129.77	53.90	137.11	56.95	129.87	53.95	126.64	52.60	94.734	39.35	97.147	40.35	101.63	42.21	130.89	54.37	131.83	54.76	152.14	63.20
8	129.05	53.64	135.48	56.31	129.21	53.71	123.79	51.46	98.612	40.99	97.456	40.51	102.91	42.78	130.74	54.34	128.86	53.56	146.37	60.84
9	127.19	52.91	140.3	58.36	127.33	52.97	121.86	50.69	94.104	39.14	100.76	41.91	101.65	42.28	128.64	53.51	126.83	52.76	163.07	67.83
10	126.54	52.68	148.69	61.90	126.85	52.81	121.4	50.54	97.403	40.55	98.407	40.97	97.951	40.78	128.21	53.37	127.45	53.06	162.7	67.73
11	125.7	52.37	178.28	74.28	126.01	52.50	120.84	50.35	86.043	35.85	93.192	38.83	99.77	41.57	127.47	53.11	124.91	52.04	160.55	66.89
12	124.97	52.11	179.63	74.90	125.23	52.22	116.96	48.77	89.94	37.50	91.758	38.26	99.918	41.66	126.58	52.78	124.56	51.94	158.66	66.16
13	124.74	52.06	210.81	87.98	125.12	52.22	116.72	48.71	93.361	38.96	93.12	38.86	97.285	40.60	126.6	52.84	124.37	51.91	160.71	67.07
14	123.77	51.71	241.99	101.1	124.32	51.94	115.76	48.36	91.025	38.03	89.777	37.51	96.699	40.40	126.29	52.76	124.19	51.89	155.8	65.09
15	123.22	51.54	282.11	118.1	123.5	51.66	117.24	49.04	90.555	37.88	92.967	38.89	99.135	41.47	124.93	52.26	123.77	51.77	138.82	58.07
16	124.15	52.00	332.36	139.2	124.35	52.09	114.32	47.89	88.712	37.16	92.762	38.86	99.672	41.75	124.81	52.28	127.62	53.46	139.21	58.31
17	123.24	51.70	582.82	244.5	124.01	52.02	116.72	48.97	87.416	36.67	94.123	39.49	101.81	42.71	129.96	54.52	124.71	52.32	126.29	52.98
18	120.62	50.69	3320.7	1395	121.66	51.12	111.1	46.69	81.363	34.19	85.653	35.99	98.107	41.23	122.39	51.43	122.95	51.67	122.49	51.47
19	122.29	51.48	2766	1164	123.36	51.94	116.61	49.09	83.37	35.10	89.206	37.56	100.93	42.49	123.03	51.80	126.85	53.40	106.17	44.70
20	121.52	51.26	1984.5	837.1	122.8	51.80	114.85	48.44	83.675	35.29	90.415	38.14	99.475	41.96	122.84	51.81	125.48	52.93	99.649	42.03
21	120	50.72	1854.7	783.9	121.99	51.56	116.97	49.44	82.25	34.76	90.11	38.08	102.39	43.27	121.84	51.49	124.07	52.44	84.521	35.72
22	120.22	50.91	1559.9	660.6	122.41	51.84	116.47	49.32	85.913	36.38	92.222	39.05	101.59	43.02	121.54	51.47	122.44	51.85	81.469	34.50
23	118.77	50.40	1748.6	742.1	121.11	51.39	113.25	48.06	89.603	38.02	85.03	36.08	99.911	42.40	120.17	51.00	120.58	51.17	89.94	38.17
24	119.44	50.79	1647.2	700.5	121.98	51.87	113.83	48.41	70.206	29.86	85.831	36.50	98.789	42.01	120.16	51.10	120.2	51.12	72.312	30.75

In summary, the estimation performance can be improved either by changing machine learner regression model or by increasing the value of n. Indeed, for most models, the RMSE decreases when n increases. The performance rank for different types of regression models for the problem of estimating traffic flow from travel duration is as following: Gaussian Process Regression models > Ensemble of Trees > Support Vector Machines > Regression Trees > Linear Regression Models. Therefore, this work chooses Gaussian Process Regression with Rational quadratic kernel function as regression model.

5.1.4. Discussion about half width of sample window n

Globally, for the GPR with kernel functions of rational quadratic, when n increases from 4 to 23, the RMSE value decreases accordingly, and the lowest one achieved at the point (n=23) is 18.9 veh/h. Then RMSE value lightly increases when n is 24, for the reason that when the n is too big, some data far away from the center point is not so related and give some extra noise influence to the estimation. However, the rate for reducing the RMSE is extremely different when n is augmented. For n from 4 to 16, the RMSE is reduced rapidly with an improvement of 65.3 veh/h. Then the RMSE is reduced slowly with an improvement of 10.3 veh/h for n from 16 to 24. Therefore, the best choice for n equals to 23 when the RMSE performance is the only criterion. Nevertheless n with 16 is the best choice if the

Table 3: RMSE value for types 11-19 of machine learning regression models under n from 2 to 24 (PE is an abbreviation of Percentage Error)

n	C. Support Vector Machines						D. Ensemble of Trees				E. Gaussian Process Regression Models							
	11	PE(%)	12	PE(%)	13	PE(%)	14	PE(%)	15	PE(%)	16	PE(%)	17	PE(%)	18	PE(%)	19	PE(%)
2	121.47	50.47	121.93	50.66	131.73	54.73	105.24	43.72	103.11	42.84	112.51	46.75	110.33	45.84	117.19	48.69	108.1	44.91
3	117.17	48.65	117.32	48.71	129.65	53.83	100.68	41.80	97.405	40.44	108.85	45.19	106.05	44.03	108.45	45.03	102.96	42.75
4	107.14	44.46	114.3	47.44	126.36	52.44	95.515	39.64	92.91	38.56	103.17	42.82	100.92	41.88	95.616	39.68	94.589	39.26
5	98.045	40.69	110.38	45.81	124.06	51.48	90.612	37.60	87.285	36.22	101.27	42.03	93.715	38.89	85.182	35.35	85.625	35.53
6	87.702	36.41	106.82	44.34	121.99	50.64	86.639	35.96	81.334	33.76	87.001	36.11	75.897	31.51	74.147	30.78	72.321	30.02
7	82.244	34.16	104.93	43.59	120.37	50.00	83.458	34.67	77.281	32.10	74.919	31.12	63.406	26.34	66.032	27.43	61.221	25.43
8	81.351	33.81	102.47	42.59	119.3	49.59	82.659	34.36	75.11	31.22	68.456	28.45	58.613	24.36	62.284	25.89	56.414	23.45
9	74.383	30.94	100.6	41.85	118.62	49.34	81.15	33.76	73.467	30.56	57.456	23.90	50.483	21.00	56.786	23.62	49.325	20.52
10	76.824	31.98	99.243	41.31	117.75	49.02	78.016	32.48	72.134	30.03	50.495	21.02	45.461	18.92	54.807	22.82	44.856	18.67
11	72.479	30.20	97.02	40.42	116.81	48.67	75.379	31.40	69.785	29.07	45.249	18.85	41.7	17.37	51.979	21.66	41.032	17.10
12	66.382	27.68	95.666	39.89	116.29	48.49	75.877	31.64	69.455	28.96	40.445	16.86	38.691	16.13	50.049	20.87	38.394	16.01
13	69.009	28.80	94.385	39.39	115.77	48.32	75.278	31.42	68.36	28.53	37.241	15.54	37.108	15.49	49.13	20.50	36.458	15.22
14	71.017	29.67	93.295	38.98	115.76	48.36	73.077	30.53	66.832	27.92	32.498	13.58	32.696	13.66	44.989	18.80	32.179	13.44
15	66.37	27.76	93.601	39.15	114.85	48.04	74.462	31.15	67.647	28.30	29.112	12.18	30.136	12.61	44.952	18.80	29.272	12.24
16	71.199	29.82	92.992	38.95	115.5	48.38	70.419	29.50	67.434	28.25	30.068	12.59	30.875	12.93	45.374	19.01	29.831	12.50
17	64.483	27.05	93.214	39.10	115.63	48.51	73.546	30.85	67.996	28.53	27.302	11.45	28.789	12.08	43.885	18.41	27.449	11.52
18	68.883	28.95	92.875	39.03	114.56	48.14	69.63	29.26	64.482	27.10	26.534	11.15	27.136	11.40	41.809	17.57	26.212	11.01
19	63.142	26.58	90.304	38.02	114.56	48.23	69.87	29.42	63.823	26.87	24.464	10.30	25.677	10.81	40.669	17.12	24.498	10.31
20	66.065	27.87	90.291	38.09	114.02	48.09	71.631	30.21	63.074	26.61	22.907	9.66	24.535	10.35	40.417	17.05	22.934	9.67
21	66.552	28.13	88.121	37.24	114.11	48.23	69.455	29.35	61.529	26.00	21.019	8.88	22.85	9.66	38.7	16.36	21.198	8.96
22	64.148	27.17	86.628	36.69	114.48	48.48	70.341	29.79	60.342	25.55	20.455	8.66	21.922	9.28	38.134	16.15	20.43	8.65
23	66.24	28.11	84.859	36.01	113.22	48.05	67.567	28.67	60.842	25.82	19.105	8.11	20.202	8.57	35.857	15.22	18.938	8.04
24	59.811	25.44	81.297	34.57	113.2	48.14	68.533	29.14	57.69	24.53	20.198	8.59	21.519	9.15	35.418	15.06	20.18	8.58

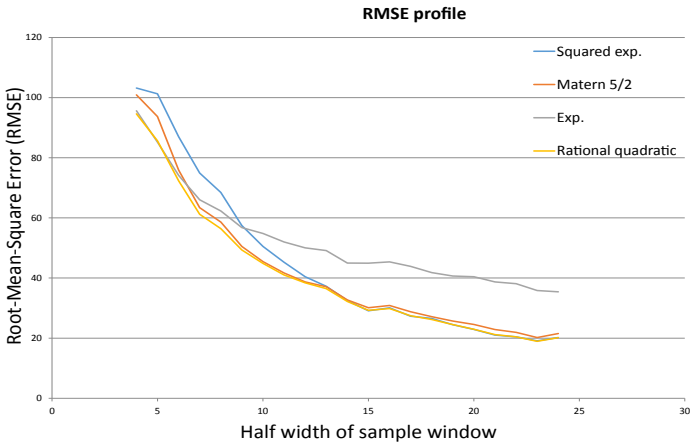

Figure 10: RMSE profile for GPR under different n

RMSE performance and length of n should be considered together, because the bigger n is, the more data is needed to estimate the traffic flow, as shown in the Fig. 5.

5.1.5. Experimental performance for a whole week

The comparison between real and estimated traffic flow and the distribution of errors is shown in detail with n as: 4, 8, 16, 24, as shown in Fig. 11—18. Firstly, with n=4, the trained GPR model can only capture the traffic flow tendency, because the n is too small and more information should be needed, as shown in the Fig. 11. As a result, for the corresponding distribution of error in the Fig. 12, only 45 percent of errors locates between the zone [-50, 50] (veh/h). The biggest error happens in 300 veh/h and the RMSE is 92 veh/h. However, when n is increased to 24, estimated traffic flow is more similar to the real one than that with n=4, as the Fig. 17 shows, which means that the increase of n can help to model the profile more exactly. Therefore, 86 percent of errors locates in the zone [-50, 50] (veh/h), which is almost twice bigger than that with n=4. The RMSE is 20.18 veh/h. Third times lower than the one with n=4. The biggest error is 170 veh/h. However, in Fig. 17, most of the obvious error happens in weekends, because the profile's shape of traffic flow is very different between workdays and the weekend, which motivates us to build a GPR model only for weekdays to improve the performance. This subject is discussed in detail in the following subsection.

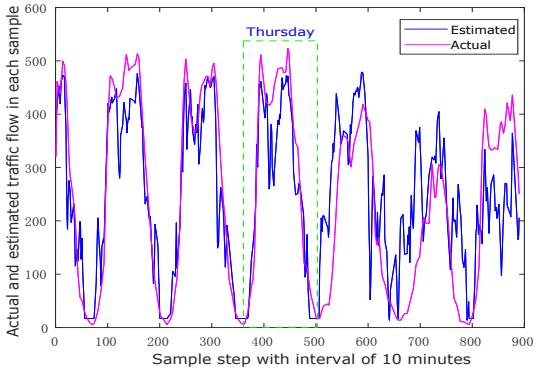

Figure 11: Actual VS estimated traffic flow with n=4 for a week

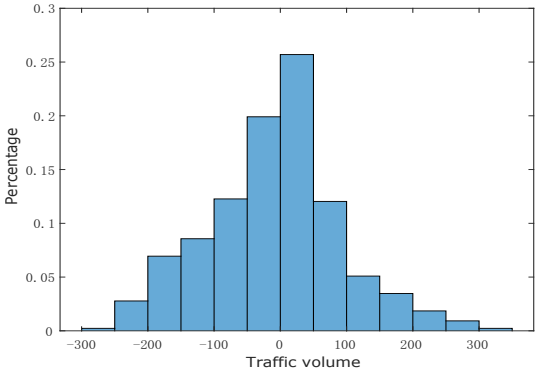

Figure 12: Normalized estimation error with n=4 for a week

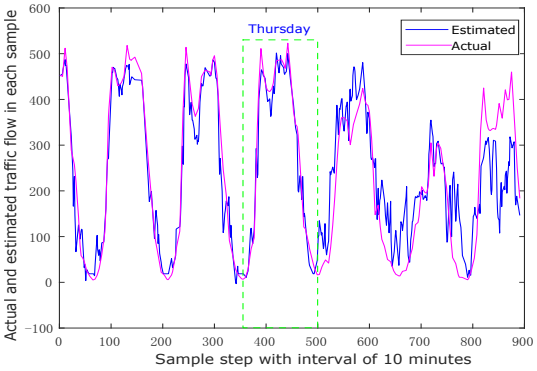

Figure 13: Actual VS estimated traffic flow with n=8 for a week

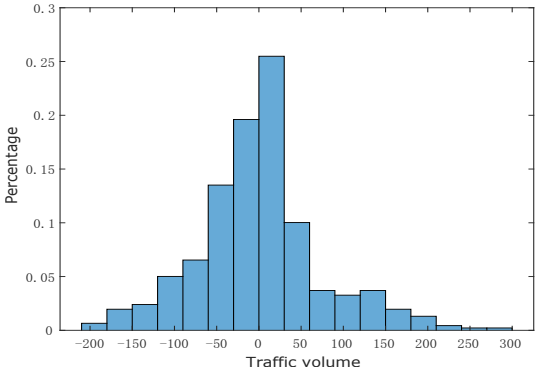

Figure 14: Normalized estimation error with n=8 for a week

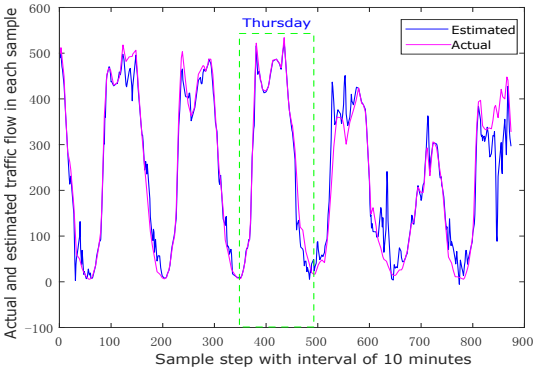

Figure 15: Actual VS estimated traffic flow with n=16 for a week

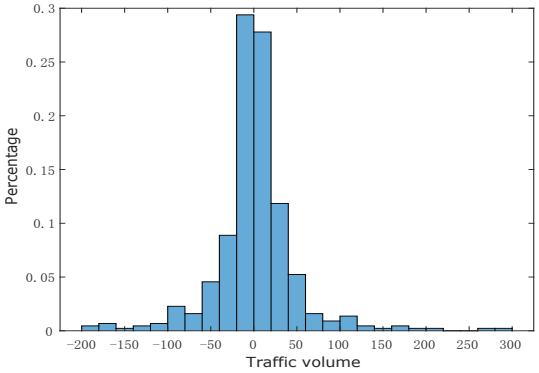

Figure 16: Normalized estimation error with n=16 for a week

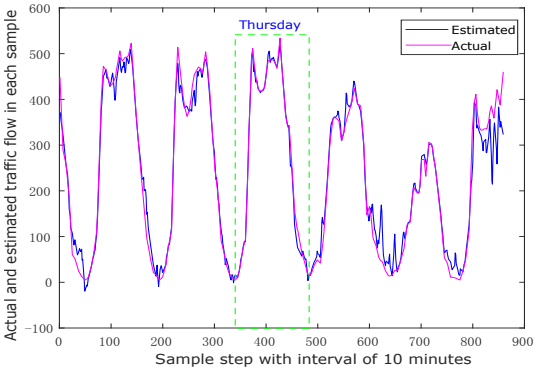

Figure 17: Actual VS estimated traffic flow with n=24 for a week

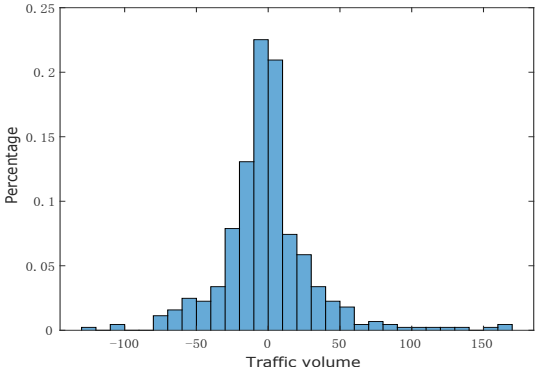

Figure 18: Normalized estimation error with n=24 for a week

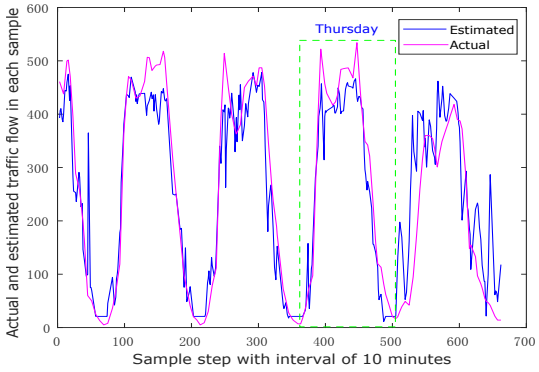

Figure 19: Actual VS estimated traffic flow with n=4 for weekdays only

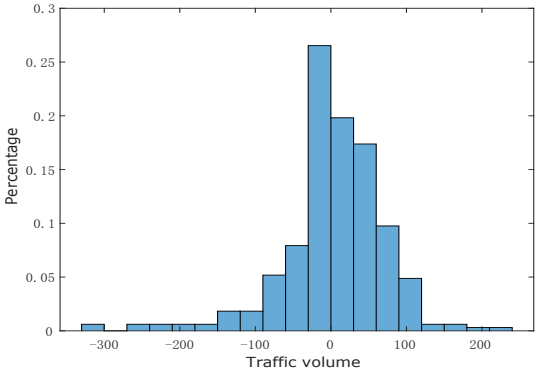

Figure 20: Normalized estimation error with n=4 for weekdays only

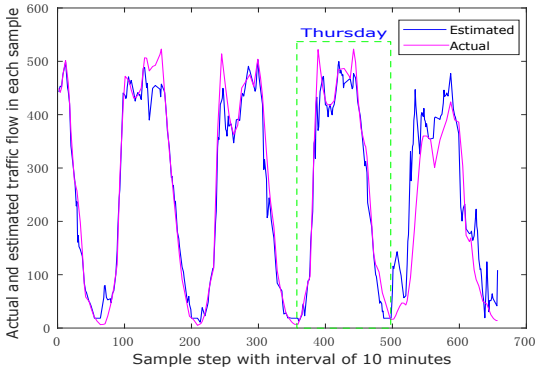

Figure 21: Actual VS estimated traffic flow with n=8 for weekdays only

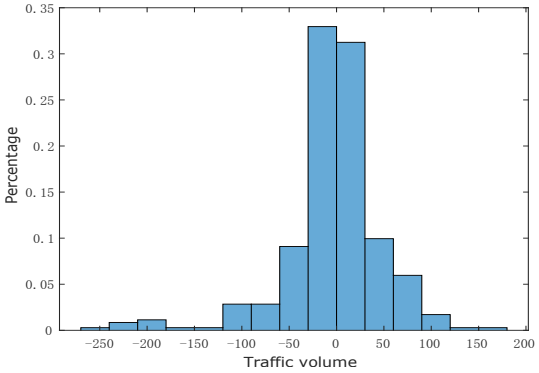

Figure 22: Normalized estimation error with n=8 for weekdays only

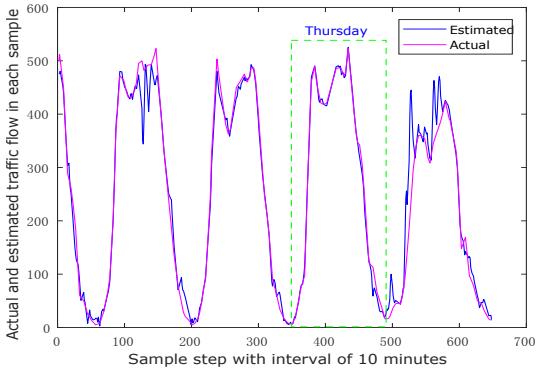

Figure 23: Actual VS estimated traffic flow with n=16 for weekdays only

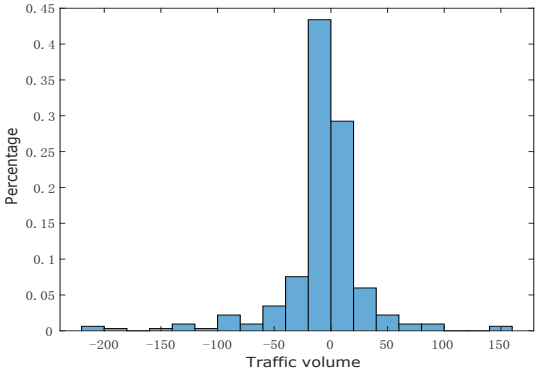

Figure 24: Normalized estimation error with n=16 for weekdays only

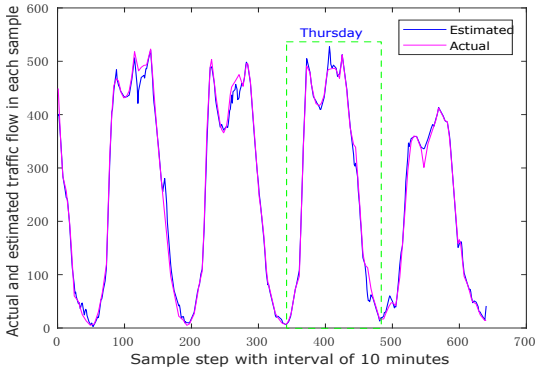

Figure 25: Actual VS estimated traffic flow with n=24 for weekdays only

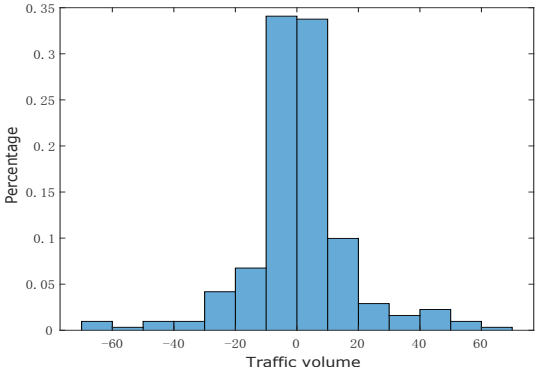

Figure 26: Normalized estimation error with n=24 for weekdays only

5.1.6. Experimental performance for only weekdays

The performance is compared between GPR models for whole weeks and only weekdays. For n=4, compared with the GPR model for the whole week (refers to Fig. 11), the GPR model for only weekdays (refers to Fig. 19) can capture the real data more precisely, and can get a higher percentage of errors (62.8%) locating between the zone [-50,50]. Furthermore, when the n is increased to 24, as shown in the Fig.26, 97 percent of errors in the zone [-50,50] , which is 11% higher than that in the GPR model for the whole week, as Fig. 18 shows. Therefore the performance can be improved by building a GPR model with smaller time zone, for example, only the weekdays instead of a whole week.

5.2. Multi-models simulation case

Firstly, the experiments' parameters are presented. Secondly, FCD profiles on training data sets are clustered into k (here $k = 4$) clusters by k-means method. At last, the obtained results (estimated traffic flows) are compared with the real observed data (real traffic flows measured by sensors).

Prior to the above clustering process, the dimension of FCD profiles is reduced by PCA (Principal Component Analysis) algorithm. Having a look at the eigen vectors' coefficients after the PCA has been applied to the initial representation shows us that the initial basis is not preserved. Indeed, most of the eigenvectors result from a combination of a large number of the initial base's vectors. As an illustration, figure 27) represents the eigenvector basis as a 33x33 gray level image. The darker is the pixel, the closer to 0 is the magnitude of the corresponding eigenvector's coefficient. Rather than observing columns mainly made of dark pixels with a few bright pixels, we see a wide distribution of gray shaded pixels affecting a large part of the image.

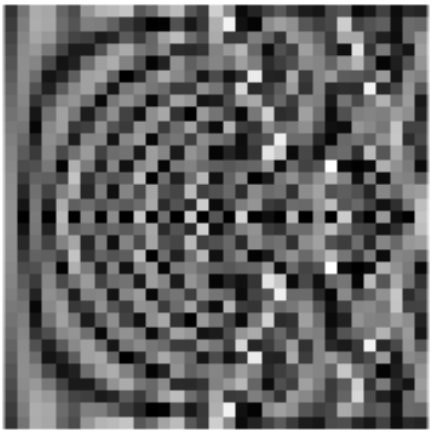

Figure 27: Representation of the eigenvector basis after PCA

5.2.1. Experiments' parameter

The experiments are executed on the road segment named "Boulevard Lahure, Douai, 59500, France" with GPS of origin (50.380163,3.079186) and destination (50.390721,3.081124) for the duration of 26 weeks. The FCD are extracted directly from the Google Map API[43]. The total number of sample data is 20208 (6*24*7*16), owing to the 10 minutes sampling period. The actual traffic flows data are provided by Douai's town council. All the sample days are divided into two sets: one containing 80 percent of the source data for training the model, and another with 20 percents of the source data for testing the model. The overfitting problem is avoided by applying the 5-folds cross validation method. The kernel function of the GPR is Rational Quadratic, since this one performed the best along the experiments. The quadratic kernel function is applied for SVM. The dimension of the feature vector is 33, because the half width of the samples window (n) is defined as 16. All training days are classified into 4 clusters with k-means algorithm. The units of parameters are as follows: traffic flow, RMSE and RMSD are expressed in veh/h, while travel durations are in s.

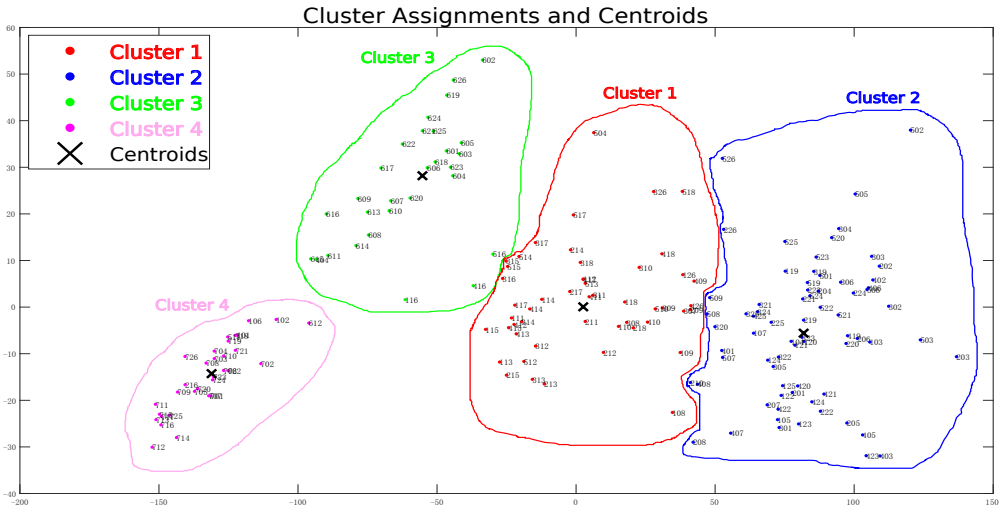

Figure 28: Training data sets are clustered into four clusters by k-means algorithm, after their dimensions are reduced by PCA method. Each day is expressed as a point, beside which the numbering schemes are defined as follows: 1) the first number represents the day within each week; 2) the last two numbers express the order of week. For example, the number "203" means the Tuesday in the third week.

5.2.2. Labeling results from PCA and k-means for training data

The clustering results are presented in Fig.28 for days with different travel durations profiles. We can observe that PCA leads to a drastic reduction of the dimension of the features space from its original value to 2, significantly decreasing the complexity of the k-means based labeling process. At the end of the clustering, training days are divided into 4 clusters: cluster 1 and 2 include most of weekdays and a small part of weekends; cluster 3 consists of most of Saturday and some weekdays, which proves that the traffic flows during some weekdays are similar to Saturday. This is the main reason explaining the different performances between multi-model from k-means selection and multi-model from manual selection: in multi-model with manual selection, all weekdays belong to the same group (refers to Figs.3 and 4); cluster 4 is mainly made of Sunday. Therefore, PCA associated to k-means can cluster the travel durations profiles into different clusters more reasonably and precisely than the manual way.

5.2.3. Comparison of statistical results

Table 4: Comparison of statistical results (RMSE and RMSD are expressed in veh/h)

Criteria	Single model	Manual multi-models	K-means multi-models
RMSE	37.3434	37.1286	23.827
RMSD	37.0281	36.9523	23.8084

The RMSE and RMSD for single model, manual multi-models and K-means multi-model are compared in the Tab.4. All types of models can estimate traffic flows with low RMSE and RMSD. Furthermore the K-means multi-models can reduce RMSE and RMSD by up to 36.2% and 35.9%, respectively, compared with single model. Because each day in testing data set is firstly input to the trained SVM model to choose the suitable trained GPR model. However, the multi-model with manual selection does not significantly improve the performance compared with single model, because such model cannot group the FCD flows profiles precisely. As we noticed before, all weekdays labeled using manual selection naturally belong to the same group. However, many FCD flows profiles during weekdays are similar to that in Saturday, as shown in Fig. 28.

The differences between the three models tend to narrow if we focus on peak hours. In the figures 29 to 40 we have considered the morning peak hours (6:00 - 9:30 am) and the evening peak hours (4:00 - 7:30 pm). Figures 29, 31, 33, 35, 35, 39 and 39 represent actual and estimated flows over a succession of peak hours. There is no time scale in these graphs, the peak hour periods succeed one another in chronological order. The corresponding error histograms

are shown in Figures 30, 32, 36, 36, 36, 40 and 40. In a general way, we can see that by considering only the peak hours, the interest of the multi-models (manual or learned) is much more reduced. We can also see that the afternoon peak hours are better estimated than the morning peak hours (16% maximum error versus 24%). This is probably due to the fact that these peaks are much more variable over the week compared to the rest of the day. This leads to some noise in the estimation.

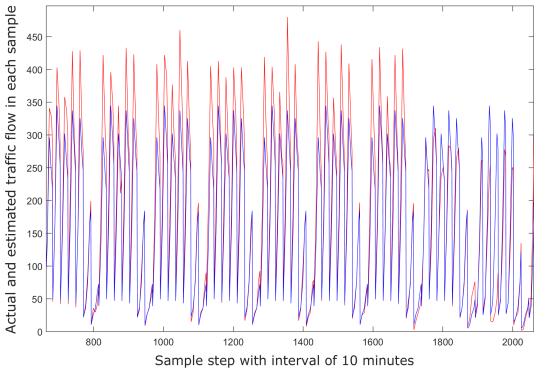

Figure 29: Actual (RED) VS estimated traffic flow (BLUE) with single model for morning peak hours

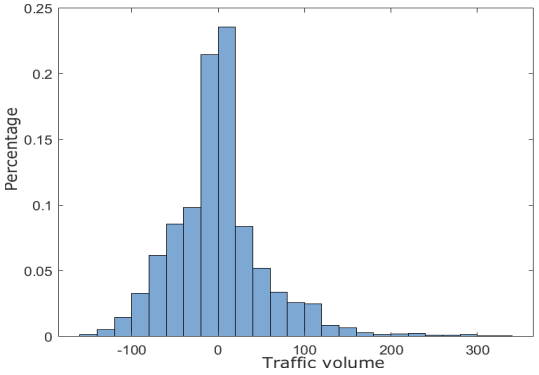

Figure 30: Normalized estimation error with single model for morning peak hours

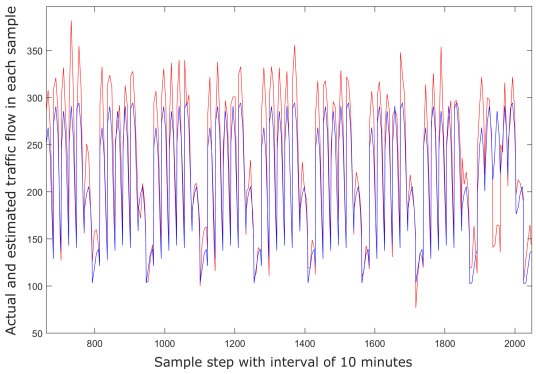

Figure 31: Actual (RED) VS estimated traffic flow (BLUE) with single model for evening peak hours

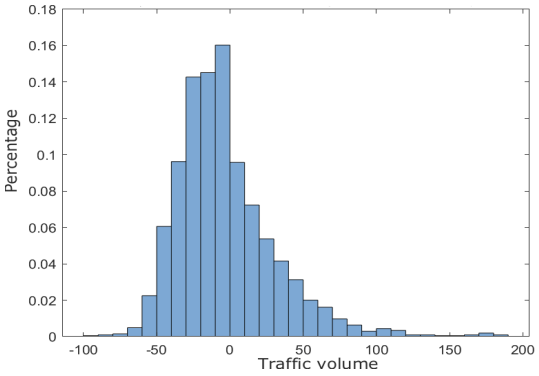

Figure 32: Normalized estimation error with single model for evening peak hours

6. Conclusion and future works

This work illustrates that Machine learning techniques based on Agregated Data permit to estimate the Traffic flows according to Floating Car Data (FCD) only on the basis of trained regressors. Principal Component Analysis (PCA) coupled with k-means technique allows to differentiate clusters of daily FCD profiles. Models are built for each cluster tuned by selecting the appropriate multi-model Gaussian Processes Regressors (GPR) using the Support Vector Machine (SVM) classifier generates coherent traffic flow.

The experimental results show that the multi-model with k-means selection can significantly reduce the Root Mean Square Error (RMSE) and Root-Mean-Square Deviation (RMSD), compared with single model or multi-model with manual selection. The produced estimation by single or multi-models is promising enough to be used instead of static sensors based measurements. Practically speaking, this could have very interesting consequences: once the GPR model has been learned for a given road (on the basis of a relatively brief in site data acquisition process), the FCD are sufficient for continuously providing a good enough estimation of the vehicles flow along that road, using our GPR regressor. Thus, as an alternative approach to investing in a costly static sensor systems for each of the roads requiring a flow estimation, we could rather use movable measurement units to acquire the data for training the GPR, and then move this system to another place and repeat the process until each model of these ”strategic” roads is

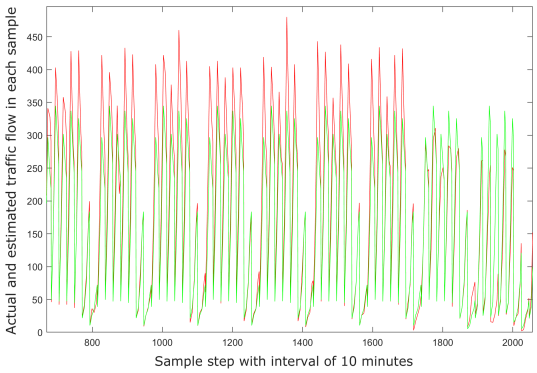

Figure 33: Actual (RED) VS estimated traffic flow (GREEN) with manual multi-models for morning peak hours

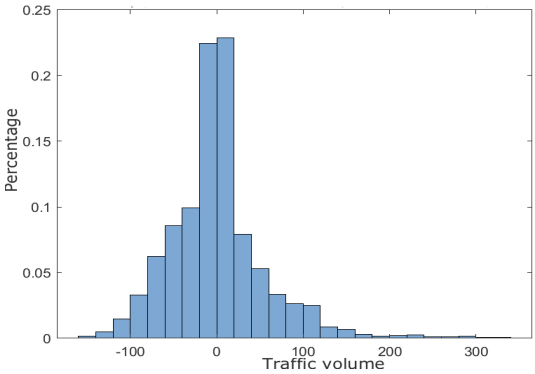

Figure 34: Normalized estimation error with manual multi-models for morning peak hours

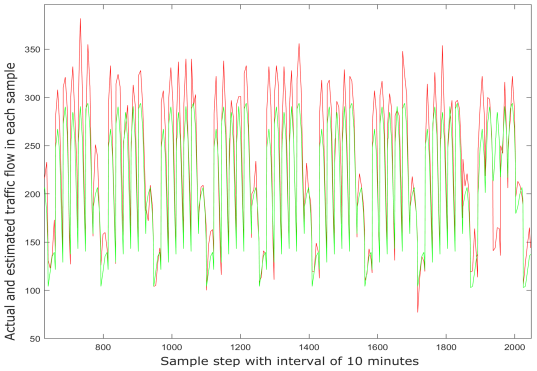

Figure 35: Actual (RED) VS estimated traffic flow (GREEN) with manual multi-models for evening peak hours

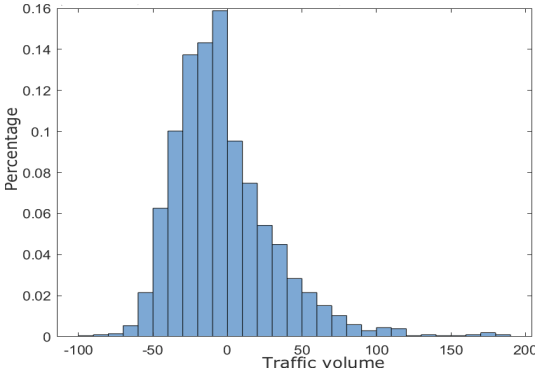

Figure 36: Normalized estimation error with manual multi-models for evening peak hours

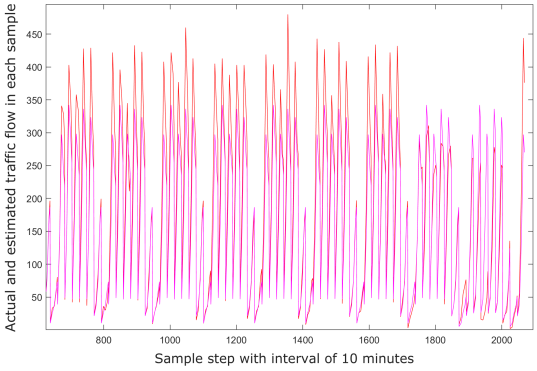

Figure 37: Actual (RED) VS estimated traffic flow (MAGENTA) with K-means multi-models for morning peak hours

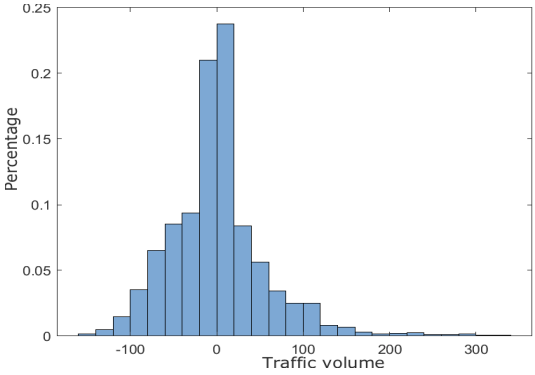

Figure 38: Normalized estimation error with K-means multi-models for morning peak hours

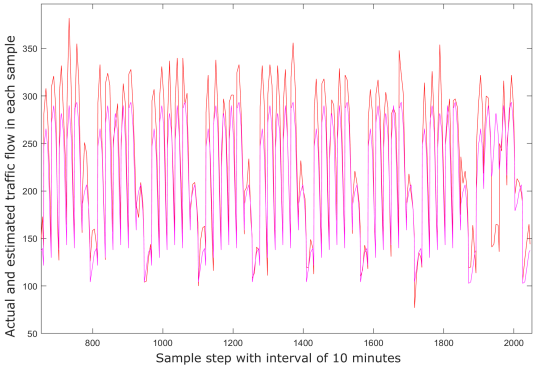

Figure 39: Actual (RED) VS estimated traffic flow (MAGENTA) with K-means multi-models for evening peak hours

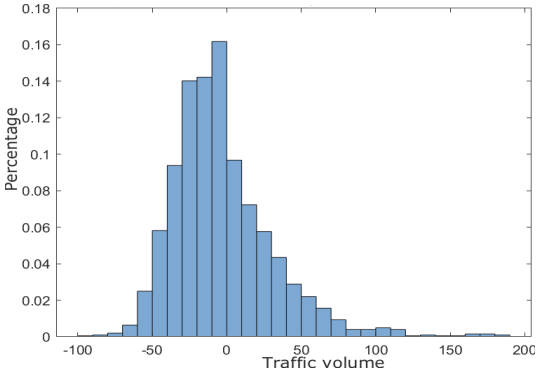

Figure 40: Normalized estimation error with K-means multi-models for evening peak hours

learned. Finaly, a GPR regressor combined with its corresponding FCD stream would act as a kind of virtual sensor, cost effective vehicles flow sensor.

These measurements, in turn, are used to build and update urban transportation networks' models, which are necessary components required for future mobility to better manage circulation flows within our cities and increase the safety of the users. In a near future, these users could be "human drivers" as well as autonomous cars. The latter may directly use the provided models while directly feeding them with their FCD for an even better quality of traffic estimation. This is exactly what we are currently developping in the scope of the ORIO project (refers to the Acknowledgment section). In fact, the proposed algorithm can be used for all traffic situation and not only urban traffic. The urban context is chosen in the work because the implementation of physical sensors (radar, induction loops) are more difficult than highway.

As a critical reflection, two main drawbacks can be highlighted. A first one concerns the use of non-sparse methods on input data which could be an issue with longer sequences to analyze, leading to obtaining a more complex learned model. A second one, classical in every machine learning solutions, relates to our dependence on the quality of input data. Let's remember that we need two types of input data: the actual flow measured on studied road section and travel time supplied by the FCD provider (e.g. Google in this article).

In a simulation perspective, our results are encouraging regarding the capacity of generating traffic flows from punctual real measurements and more global aggregated data. The virtual sensors reproduce flows according to the curve of the day. The aim is for these sensors to be distributed all over the network. The multi-model permits to gain in accuracy and to provide a collection of 'typical days' to use in simulations. Our perspective is first to generate virtual sensors in key road segments that capture the main entrance and way out of a studied area. Beyond the simulation perspective, questions remain on the possibility to take advantage of a model learned in a specific road segment for other 'similar' road segments in the network. The test of the proposed algorithms on other areas of the city and other transportation context will be the subject of our future work. To do so, the model can integrate topological information (dimension of the segment, relative position to the city center...) as well as temporal information in a multi-model learned from several road segments in order to help in producing generalizable models.

The input vector of the GPR regressor is a sequence of consecutive FCD values centered at the **sample step** for which the flow estimation has to be produced. To make it clear, the proposed GPR does not directly take into account the **"time"** (says hours, minutes and seconds), neither does it use the name of the day itself while it produces a rather good estimation of the flows. It is no secret saying that traffic flows evolve with the time of the day and the day of the week (the traffic during the weekend sometimes has nothing to do with what it is on business days!). Then, instead of feeding our GPR learning process with data including a complete week, we plan to build sets of "period of the day" related GPR regressors (for instance: night, morning, midday, afternoon and evening) in replacement of "weekly" GPR regressors computed from several road segment data.

Acknowledgment

This work was supported by the project ORIO (Observing the peRformances of urban Infrastructures and mobility / preventing collisions with vulnerable people using Opportunistic radar). The project ORIO is done within the framework of ELSAT2020, which is co-financed by the European Union with the European Regional Development Fund, the French state and the Hauts de France Region Council. The real traffic flow data from induction loops on the road "86 Boulevard de la République, Douai, France" was provided by the town hall of Douai (mairie de douai).

References

[1] J. Li, J. Boonaert, A. Doniec, G. Lozenguez, Toward reliable estimations of urban traffic flows from machine learning and floating car data, in: 15th World Conference on Transport Research, Vol. In presse, 2019.

[2] J. Li, J. Boonaert, A. Doniec, G. Lozenguez, Traffic flow multi-model with machine learning method based on floating car data, in: 6th International Conference on Control Decision and Information Technologies, 2019, pp. 512–517.

[3] P. van den Haak, T. Bakri, R. Van Katwijk, M. Emde, M. Snelder, et al., Validation of google floating car data for applications in traffic management, in: Transportation Research Board 97th Annual Meeting, no. 18-00609, 2018.

[4] T. Jeske, Floating car data from smartphones: What google and waze know about you and how hackers can control traffic, Proceeding of the BlackHat Europe (2013) 1–12.

[5] S. Cheung, S. Coleri, B. Dundar, S. Ganesh, C.-W. Tan, P. Varaiya, Traffic measurement and vehicle classification with single magnetic sensor, Transportation research record: journal of the transportation research board (1917) (2005) 173–181.

[6] B. Coifman, D. Beymer, P. McLauchlan, J. Malik, A real-time computer vision system for vehicle tracking and traffic surveillance, Transportation Research Part C: Emerging Technologies 6 (4) (1998) 271 – 288.

[7] T. Miwa, Y. Tawada, T. Yamamoto, T. Morikawa, En-route updating methodology oftravel time predictionusing accumulated probe-car data.

[8] S. Turksma, The various uses of floating car data, in: 10th International Conference on Road Transport Information and Control, 2000, pp. 51–55.

[9] J. Yoon, B. Noble, M. Liu, Surface street traffic estimation, in: Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, 2007, pp. 220–232.

[10] C. De Fabritiis, R. Ragona, G. Valenti, Traffic estimation and prediction based on real time floating car data, in: 11th International IEEE Conference on Intelligent Transportation Systems, 2008.

[11] C. Nanthawichit, T. Nakatsuji, H. Suzuki, Application of probe-vehicle data for real-time traffic-state estimation and short-term travel-time prediction on a freeway, Transportation Research Record: Journal of the Transportation Research Board (1855) (2003) 49–59.

[12] T. Erdelić, T. Carić, M. Erdelić, L. Tišljarić, A. Turković, N. Jelušić, Estimating congestion zones and travel time indexes based on the floating car data, Computers, Environment and Urban Systems 87 (2021) 101604.

[13] Y. Asakura, T. Kusakabe, N. X. Long, T. Ushiki, Incident detection methods using probe vehicles with on-board gps equipment, Transportation Research Procedia 6 (2015) 17–27.

[14] Y. Chen, C. Chen, Q. Wu, J. Ma, G. Zhang, J. Milton, Spatial-temporal traffic congestion identification and correlation extraction using floating car data, Journal of Intelligent Transportation Systems 25 (3) (2021) 263–280.

[15] L. Chen, J. Shi, M. Cheng, H. Zhu, L. Sun, Characteristics of urban road non-recurrent traffic congestion based on floating car data, in: Proceedings of 4th International Conference on Electronic Information Technology and Computer Engineering, 2020, p. 120–126.

[16] X. Dai, M. A. Ferman, R. P. Roesser, A simulation evaluation of a real-time traffic information system using probe vehicles, in: Proceedings of IEEE Intelligent Transportation Systems, 2003.

[17] J. Hong, X. Zhang, Z. Wei, L. Li, Y. Ren, Spatial and temporal analysis of probe vehicle-based sampling for real-time traffic information system, in: IEEE Intelligent Vehicles Symposium, 2007.

[18] J. J. Vázquez, J. Arjona, M. Linares, J. Casanovas-Garcia, A comparison of deep learning methods for urban traffic forecasting using floating car data, Transportation Research Procedia 47 (2020) 195–202.

[19] C. de Fabritiis, R. Ragona, G. Valenti, Traffic estimation and prediction based on real time floating car data, in: 11th IEEE International Conference on Intelligent Transportation Systems, 2008, pp. 197–203.

[20] B. S. Kerner, C. Demir, R. G. Herrtwich, S. L. Klenov, H. Rehborn, M. Aleksic, A. Haug, Traffic state detection with floating car data in road networks, in: Proceedings of IEEE Intelligent Transportation Systems, 2005, pp. 44–49.

[21] A. Sunderrajan, V. Viswanathan, W. Cai, A. Knoll, Traffic state estimation using floating car data, Procedia Computer Science 80 (2016) 2008–2018.

[22] J. Hong, X. Zhang, Z. Wei, L. Li, Y. Ren, Spatial and temporal analysis of probe vehicle-based sampling for real-time traffic information system, in: IEEE Intelligent Vehicles Symposium, 2007, pp. 1234–1239.

[23] H. Wu, Comparing google maps and uber movement travel time data, Transport Findings (2019) 1–5.

[24] J. H. Wu, M. Florian, S. He, An algorithm for multi-class network equilibrium problem in pce of trucks: application to the scag travel demand model, Transportmetrica 2 (1) (2006) 1–9.

[25] W. H. Lam, H.-J. Huang, Calibration of the combined trip distribution and assignment model for multiple user classes, Transportation Research Part B: Methodological 26 (4) (1992) 289–305.

[26] W. Brilon, J. Lohoff, Speed-flow models for freeways, Procedia-Social and Behavioral Sciences 16 (2011) 26–36.

[27] D. Groth, S. Hartmann, S. Klie, J. Selbig, Principal components analysis, in: Computational Toxicology, Springer, 2013, pp. 527–547.

[28] S. T. Lim, D. F. W. Yap, N. A. Manap, Medical image compression using block-based pca algorithm, in: International Conference on Computer, Communications, and Control Technology, 2014, pp. 171–175.

[29] K. P. Sinaga, M.-S. Yang, Unsupervised k-means clustering algorithm, IEEE Access 8 (2020) 80716–80727.

[30] A. Alsayat, H. El-Sayed, Social media analysis using optimized k-means clustering, in: IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA), 2016, pp. 61–66.

[31] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, D. Haussler, Support vector machine classification and validation of cancer tissue samples using microarray expression data, Bioinformatics 16 (10) (2000) 906–914.

[32] V. Vapnik, Statistical learning theory. 1998, Vol. 3, Wiley, 1998.

[33] V. Vapnik, The nature of statistical learning theory, Springer science & business media, 2013.

[34] C. E. Rasmussen, Gaussian processes in machine learning, in: Advanced lectures on machine learning, Springer, 2004, pp. 63–71.

[35] Y. Xie, K. Zhao, Y. Sun, D. Chen, Gaussian processes for short-term traffic volume forecasting, Transportation Research Record: Journal of the Transportation Research Board (2165) (2010) 69–78.

[36] J. Hu, J. Wang, Short-term wind speed prediction using empirical wavelet transform and gaussian process regression, Energy 93 (2015) 1456–1466.

[37] T. Idé, S. Kato, Travel-time prediction using gaussian process regression: A trajectory-based approach, in: Proceedings of the SIAM International Conference on Data Mining, 2009.

[38] J. Hu, X. Li, Y. Ou, Online gaussian process regression for time-varying manufacturing systems, in: 13th IEEE International Conference on Control Automation Robotics & Vision (ICARCV), 2014, pp. 1118–1123.

[39] F. V. Jensen, An introduction to Bayesian networks, Vol. 210, UCL press London, 1996.

[40] D. J. MacKay, Gaussian processes-a replacement for supervised neural networks?, Citeseer (1997) 1–31.

[41] C. K. Williams, C. E. Rasmussen, Gaussian processes for machine learning, the MIT Press 2 (3) (2006) 4.

[42] T. Chai, R. R. Draxler, Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature, Geoscientific model development 7 (3) (2014) 1247–1250.

[43] Google, Google map api, `https://cloud.google.com/maps-platform`, accessed: 2018-11-20.