Code will be available on: https://github.com/SunnyBingoMe/sun2017flowaware-github

Flow-Aware WPT k-Nearest Neighbours Regression for Short-Term Traffic Prediction

Bin Sun*, Wei Cheng†, Prashant Goswami* and Guohua Bai*
*Blekinge Institute of Technology, Karlskrona 37179, Sweden
Email: {bin.sun, prashant.goswami, guohua.bai}@bth.se
†Kunming University of Science and Technology, Kunming 650093, China
Email: wei.cheng@bth.se

***Abstract*—Robust and accurate traffic prediction is critical in modern intelligent transportation systems (ITS). One widely used method for short-term traffic prediction is k-nearest neighbours (kNN). However, choosing the right parameter values for kNN is problematic. Although many studies have investigated this problem, they did not consider all parameters of kNN at the same time. This paper aims to improve kNN prediction accuracy by tuning all parameters simultaneously concerning dynamic traffic characteristics. We propose weighted parameter tuples (WPT) to calculate weighted average dynamically according to flow rate. Comprehensive experiments are conducted on one-year real-world data. The results show that flow-aware WPT kNN performs better than manually tuned kNN as well as benchmark methods such as extreme gradient boosting (XGB) and seasonal autoregressive integrated moving average (SARIMA). Thus, it is recommended to use dynamic parameters regarding traffic flow and to consider all parameters at the same time.**

***Keywords*—Flow-Aware, Weighted Parameter Tuples, k-Nearest Neighbours Regression, Short-Term Traffic Prediction**

I. Introduction

Increasing road traffic is nowadays causing more congestion and accidents which gain more attention from public and authorities due to severe loss of life and property [1], [2]. Efficient traffic management and automatic accident detection are key requirements in modern intelligent transportation systems (ITS). Reliable and accurate short-term traffic forecasting is necessary for achieving efficient traffic management and accident detection [3]. Predicting short-term traffic is a complex task, which has been a research subject of many studies in the past few decades [4].

The existing short-term traffic forecasting methods can be divided into two categories which are parametric and non-parametric methods [4]–[6]. A typical parametric method is seasonal autoregressive integrated moving average (SARIMA) [7]. Moving average considers that the near-future data is similar to the latest data and uses a weighted average of latest data as predictions. An autoregressive model is based on interdependent observations of stationary time series. Values of one stationary time series are in a range with a constant variance considering existing data and future data with a constant average. Within a stationary series, interdependency between history and future can be used to make a prediction. If the original time series is non-stationary, an integrated part is needed in the model to make it stationary by conduct differencing. Traffic data usually meet the peak during noon time and bottom after midnight. Besides, weekends have a different traffic scenario when being compared with workdays. Those patterns are modelled as seasonal part in SARIMA.

Within the non-parametric category, decision trees have been widely used. Gradient boosting is one way to improve decision trees. It improves tree models by focusing on badly predicted instances iteratively. Extreme gradient boosting (XGB) is a fast and regularised gradient boosting implementation with higher calculation speed and more robustness against overfitting. As a state-of-the-art method, XGB has shown outstanding performance and efficiency which is comparable with or better than random forest [8], [9]. Thus, XGB is used as a benchmark method in this study.

Another important non-parametric algorithm is k-Nearest Neighbours (kNN) [10]. Both classification and regression tasks can be handled using kNN. For time series regression, the key idea in kNN is to find history data with similar patterns of most recent data. This paper uses kNN because of the substantial increase in data availability [4], the flexibility of kNN for solving non-linear problems and easiness of understanding and implementation [11]. Some studies show that kNN is better than traditional methods (such as Kalman filter and SARIMA) [12], [13]. However, some others reported that kNN has similar performance with traditional methods [14], [15]. Thus, SARIMA is also used as a benchmark method in this study.

Three parameters of kNN are the number of nearest neighbours (k), search step length (d) (also known as lag) and window size (v) (also known as constraint) [14]. Though distance measurement of neighbours can also be considered as a dynamic parameter, it is beyond this paper's scope and will be addressed separately. Besides, when m (number of predict steps ahead) changes, the parameters should also be tuned. Many studies tried to tune the parameter k and some work also tried to tune d while few researchers considered v.

Previous work has focused on some of the parameters, but the value assignment for those three parameters at the same time is still a problem. On one aspect, we propose to use weighted parameter tuples (WPT) to improve kNN by considering all parameters together. On another aspect, we focus on flow-aware parameter tuning, while previous studies focused on improving kNN from temporal aspect (time-aware)

 http://ieeexplore.ieee.org/document/8024503/

[8], [15], such as the hour of day or day of week etc.

II. Background and Problem Settings

This section contains some background definitions and formulate the problem in this paper mathematically.

A. kNN Regression for Prediction

Suppose there is a time series with data on time points: $\cdots, t-2, t-1, t, \cdots$ where t is the near-future to predict. The time interval can differ from system to system and is usually between 1 minute and 15 minutes.

The following steps are employed to predict the traffic at time t as shown in Figure 1.

Step one is to construct state vector as the query to represent current traffic state by selecting the latest data. If we choose search step length as d, the state vector of current flow rate is:

$$\mathbf{r}_{[t]} = [r_{t-1}, r_{t-2}, \cdots, r_{t-d}] \tag{1}$$

while the state vector of current speed is:

$$\mathbf{s}_{[t]} = [s_{t-1}, s_{t-2}, \cdots, s_{t-d}] \tag{2}$$

For the distance measurement with two variables, the state vector is:

$$\mathbf{S}_{[t]} = \begin{bmatrix} r_{t-1} & r_{t-2} & \cdots & r_{t-d} \\ s_{t-1} & s_{t-2} & \cdots & s_{t-d} \end{bmatrix} \tag{3}$$

The search step length is sometimes referred as "window" in some areas, though we have another kNN parameter named window. To distinguish them, the phrase "search step length" is used in this work.

Step two is to find the most similar vectors (nearest neighbours) of history days when being compared with current query state vector. For instance, data from time point $t-d$ to $t-1$ of yesterday (Day_{-1}), from $t-d$ to $t-1$ of Day_{-2} etc. are the neighbours to compare.

In kNN, window size (constraint) (v) is used to describe the maximum time point shift when searching for neighbours. If $v = 2$, the maximum shift of time point is two, therefore, all possible shifts are -2,-1,0,+1,+2. For instance, Figure 1 is showing situation with $v = 1$ and the number of neighbours within yesterday is three instead of one. The time points of neighbours in yesterday are: $[t_0 - d - 1, \cdots, t_0 - 2]$ (with shift -1), $[t_0 - d, \cdots, t_0 - 1]$ (with shift 0) and $[t_0 - d + 1, \cdots, t_0]$ (with shift +1). Thus, if there are n days to search, then there are $n \times (2v + 1)$ neighbours to search to find k nearest neighbours.

Finally, the average value of all k nearest neighbours' predictions is calculated as the final prediction.

B. Mathematical Problem Settings

The three critical parameters of kNN include the number of neighbours k, search step length d and window size v. Suitable values should be assigned to them. If k has M_K options, d has M_H and v has M_V options, the possible number of tuples of those parameters is huge: $M_K \cdot M_H \cdot M_V$. When three parameters are considered together, a legitimate combination is called a *tuple* which is a 3-tuple in this work. For instance, (k = 8, d = 4, v = 0) is a tuple that can be used by the three kNN parameters.

III. Methodology

The kNN regression algorithm is introduced in the previous section. This section describes how the usage of WPT can enhance kNN and how the parameter-tuples' predictions can be employed to generate a weighted average. Later on, we explain how the weights are generated and present several weighting methods to assign weights.

The general idea is to conduct predictions using different values of (k, d, v) values for all training time points, and the performance of tuples can be measured. According to the performance, weighting functions generate weights for each configuration of (k, d, v). Later, selected (k, d, v) tuples are used to make prediction and a weighted average is calculated as the final result.

An overview of the proposed WPT algorithm is shown in Figure 2.

A. Traffic Metrics Selection

Traffic engineering contains three fundamental metrics which are flow rate, speed and density [16]. Given two metrics, it is just enough to calculate the remaining metric. Most previous studies use only one of them, which is usually flow rate, sometimes speed. This work uses both flow rate and speed.

B. Training Weights

The following content explains how to train weights and to predict traffic using all parameter tuples.

We conduct multi steps ahead prediction using kNN with all given parameter values for each training time point. A tuple set ($\mathbb{P}$) is a Cartesian product of three sets which contains values of k, d and v: $\mathbb{K}$ contains M_K values, $\mathbb{D}$ contains M_D values and $\mathbb{V}$ contains M_V values. The performance of each tuple (k, d, v) is measured. The set of the tuples to be measured is:

$$\begin{gathered} \mathbb{P} = \mathbb{K} \times \mathbb{D} \times \mathbb{V} = \\ \{(k_{i_k}, h_{i_d}, v_{i_v}) | k_{i_k} \in \mathbb{K}; h_{i_d} \in \mathbb{D}; v_{i_v} \in \mathbb{V}\} \end{gathered} \tag{4}$$

where $i_k = 1, 2, \cdots, M_K; i_d = 1, 2, \cdots, M_D; i_v = 1, 2, \cdots, M_V$ and $\mathbb{P}$ contains $M_P = M_K \cdot M_D \cdot M_V$ tuples of k, d and v.

For each $p_{i_p} \in \mathbb{P}, (i_p = 1, 2, \cdots, M_P)$, three values are used to set up kNN and make prediction. Later, prediction error (ε) is measured. Error of flow rate prediction for each p_{i_p} is measured as $\varepsilon(r_{[t]})_{i_p} = f_e(\hat{r}_{[t]}(p_{i_p}), r_{[t]})$ where $\hat{r}_{[t]}(p_{i_p})$ is the predicted flow rate at t using p_{i_p} and f_e is the function to measure prediction error. For error of speed prediction, the same measurement is used as $\varepsilon(s_{[t]})_{i_p} = f_e(\hat{s}_{[t]}(p_{i_p}), s_{[t]})$ where $\hat{s}_{[t]}(p_{i_p})$ is the predicted speed at t using p_{i_p}.

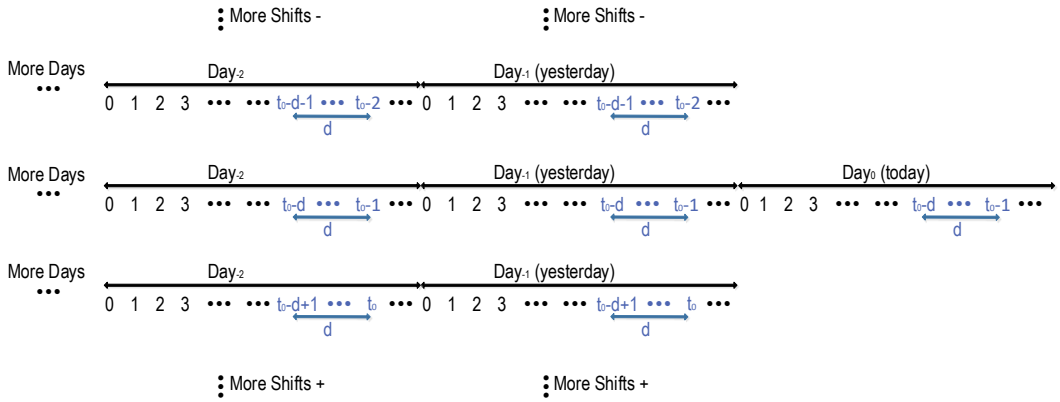

Figure 1. Prediction for time t using kNN with window size $v = 1$ (shifts: -1,0,+1). History data last until time point $t - 1$. Search step length (lag) is d.

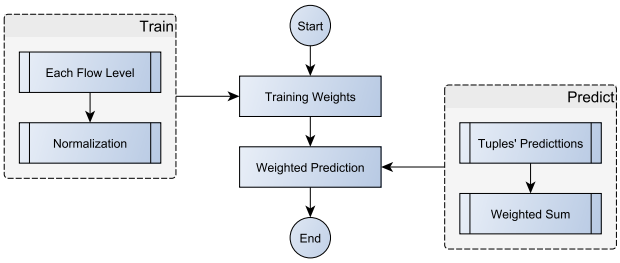

Figure 2. Overview of the proposed WPT algorithm.

1) Flow-Aware Weights: Previous research has shown that the traffic has time-variant dynamic characteristics in the different hour of day, day of the week, holiday versus non-holiday etc. [8], [15]. However, if records are separated by time of day, it is not guaranteed that the flow situation is the same, neither the day of week, etc. Instead, records are separated by flow rate levels. A practical method is to separate the flow into ten levels equally from the lowest flow rate to the highest flow rate. For instance, for a road with peak flow of one hundred cars per time unit and zero during night, from 0 to 10 will be the lowest level while from 90 to 100 is the highest level. WPT adapts the weights of tuples to the flow according to separated flow levels. The present flow rate is determined by averaging flow rate of the last 15 minutes, as it is the minimum time to have stable traffic flow [20].

2) Weighting Function: It is worth mentioning that the weights are for tuples, not for neighbours or search steps (lags). *Rank-based weighting* generates scores according to the ranks of candidates when sorted according to increasing order of distances from the subject profile [21]. Previous work shows that rank-based weighting is better than inverse weighting [22]. When using weight dispersion measure of 1, the score generated by rank-based weighting (c_{rnk}) is as below (Equation 5).

$$c_{rnk} = M_P - \gamma + 1 \tag{5}$$

where γ is the rank of the candidate ($\gamma = 1, 2, \cdots, M_P$). The candidate with $\gamma = 1$ has the lowest ε. The scores for all p's form an arithmetic sequence: $[M_P, M_P - 1, \cdots, 1]$.

Each p_i is tested and corresponding scores are calculated and added to the p_i's total score. The tuples with highest scores (e.g., top 25%) are used on each flow level. Finally, a normalisation procedure is conducted for weights to make sure the sum of weights is 1 for each flow level. An easy-to-understand logic will be presented in experimental design later. The final weights are as below:

$$\mathbf{W} = [w_1, w_2, \cdots, w_{M_P}] \tag{6}$$

C. Predicting using Weights

Instead of assigning weights to nearest neighbours, WPT assigns them to the parameter tuples (p_i).

To predict the flow rate, the following steps are taken.

Firstly, all tuples of (k, d, v) in $\mathbb{P}$ are used to calculate predictions. The predicted flow rate values are noted as below (Equation 7):

$$\hat{\mathbf{R}} = [\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_{i_p}, \cdots \hat{r}_{M_P}]; i_p = 1, 2, \cdots, M_P \tag{7}$$

As the weights are $\mathbf{W}$, the weighted flow rate prediction is:

$$\hat{r} = \mathbf{W} \cdot \hat{\mathbf{R}} \tag{8}$$

To predict the speed, a similar procedure is used.

1) Distance of Neighbours: Euclidean distance (ED) is a widely used traditional and ordinary distance in transportation data for finding nearest neighbours [24]. It is easy to understand, implement and fast to calculate [26].

Here is a brief description of ED. Suppose $i(1 \leq i \leq n_1)$ is instance index and $j(1 \leq j \leq n_2)$ is variable index in dataset $\mathbf{X} = [[x_{ij}]]$ that contains n_1 observations of n_2 variables.

ED_i is the distance between query instance $\mathbf{x_i}$ and a neighbour instance $\mathbf{x'_i}$. Both instances have d intervals. As:

$$\mathbf{x_i} = \begin{bmatrix} \mathbf{r}_{[t]} \\ \mathbf{s}_{[t]} \end{bmatrix}; \mathbf{x'_i} = \begin{bmatrix} \mathbf{r'}_{[t]} \\ \mathbf{s'}_{[t]} \end{bmatrix} \tag{9}$$

$$ED_i = |\mathbf{x_i} - \mathbf{x'_i}| = \frac{\sum_{i_d=1}^{d} \sqrt{(r_{t-i_d} - r'_{t-i_d})^2 + (s_{t-i_d} - s'_{t-i_d})^2}}{d} \tag{10}$$

Big values of d may cause the curse of dimension problem for distance measurement, which is caused by the sparse space [27]. To avoid this problem, we consider all search steps in the time domain as one dimension instead of d dimensions. This is done by averaging the distances of all search steps. Some other types of distances are available, for example, Mahalanobis distance [28], [29], which is beyond this paper's scope.

IV. Experiments

A. Data Specification

The real world data is collected by a traffic management centre who deployed dozens of devices along a highway named Kunshi. Each device sends one statistical record at five-minute intervals. Each record contains a timestamp and some statistical values such as flow rate and average speed. Part of the data from one monitoring device on the road is used. The time range of data is from April 2013 to May 2014. This road is under-saturated except during holidays.

B. Experimental Design

The beginning 80% data is used as training data to train weights, the remaining 20% is left as test data. To get rid of the influence of incidents, kNN ignores one neighbour if more than 10% steps in its searching steps or prediction steps contain incident data.

For the values of k, d and v, exponential incremental values are used as follows: $\mathbb{K} = \{2, 4, 8, \cdots, 256\}$, $\mathbb{D} = \{2, 4, 8, \cdots, 256\}$, $\mathbb{V} = \{0, 4, 8, 16, 32\}$. v starts from 0 so that the results can be compared with previous studies which did not use window. As 256 days is more than half, which is supposed to be big enough for k when v is zero. For d, 256 is the maximum value because one day contains $288 \cdot 5$ minutes. For v, the values start from 0 to compare with general no window situation. Thus, $\mathbb{P}$ contains $M_P = 320$ pairs of k, d and v.

To solve the huge calculation load, graphics processing unit (GPU) is used to reduce calculation time especially for training stage. GPU makes it practical to conduct a huge amount of calculations by accelerating them significantly. The detailed algorithm logic diagram with GPU kernels is shown in Figure 3. The analysis of one year data using non-optimized R implementation would consume more than a year to conduct experiments on a central processing unit (CPU). GPU reduces this computational time to hours. As the original traffic data is only around one megabyte, it is divided into smaller chunks which are copied into shared memory from global memory and shared by threads. This further accelerates calculations. Four kernels are developed for the experiments and the first two kernels are used in both training and predicting stage.

The experiment is conducted using CUDA [30] driver v8.0 and runtime v7.5 on GeForce 690 card. The analysis environment is R programming language [31] v3.3. XGB implementation is from R library *xgboost* v0.4.4. One parameter of XGB should be set, which is the number of iterations. In this paper, 16 can produce the best results. SARIMA implementation is from R library *forecast* v7.2 in which a fully automatic and objective SARIMA parameter value assignment is implemented.

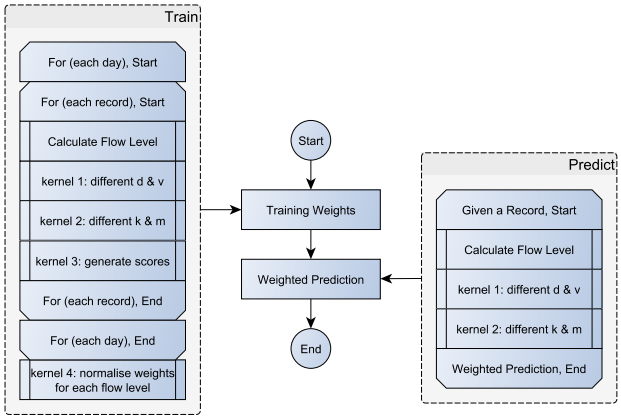

Figure 3. WPT logic diagram with GPU kernels.

C. Performance Measurement (f_e)

To measure the performance of prediction, mean absolute error (MAE) is used as f_e. For the flow rate prediction measurement:

$$\mathrm{MAE} = \frac{\sum_{\delta=1}^{q} |\hat{r}_\delta - r_\delta|}{q} \tag{11}$$

where $\hat{r}$ is the predicted flow rate, r is the grant truth flow rate, and q is the number of records. Mean absolute percentage error (MAPE) is not used because the flow rate is sometimes zero after midnight before morning.

As mentioned before, the records are separated to ten dataset $\mathbf{X'}$ ($\mathbf{x_i}$) according to ten evenly separated flow levels. Each flow level contains about ninety thousand records in average for training, which are used to generate scores and weights.

V. Results

This section firstly provides results regarding the impact of flow rate. Later comes the comparison of different benchmark methods (XGB and SARIMA) in terms of the accuracy.

A. Impact of Flow Rate

The weighted averages of parameter values are shown in Figure 4. Though WPT uses weights to calculate prediction, the weights can also be applied to parameter options to analyse how flow impact parameters. The results show that the parameter values are not linearly correlated with flow rate. There is no obvious similarity among patterns of three parameters when flow changes.

Besides, those patterns may differ when comparing decreasing flow with increasing flow. The traffic flow rate increases from 3 am to 15 pm and decreases after 15 pm until the next early morning. The patterns of weighted k and d are similar, but the pattern of v values is different when flow level is lower than 4. Thus, only v's values are plotted as shown in Figure 5.

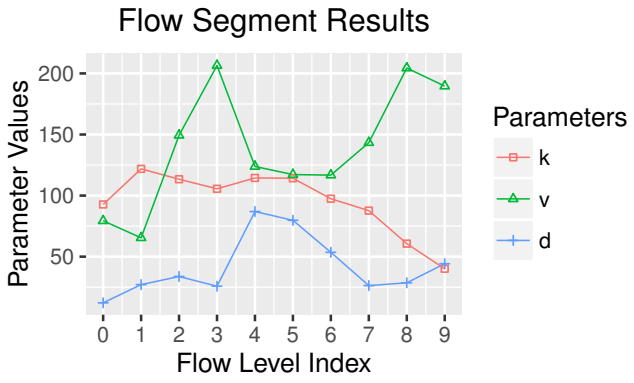

Figure 4. Parameters impacted by flow rate when the flow rate is increasing during morning time. Values of v are squared to ease display on the same scale.

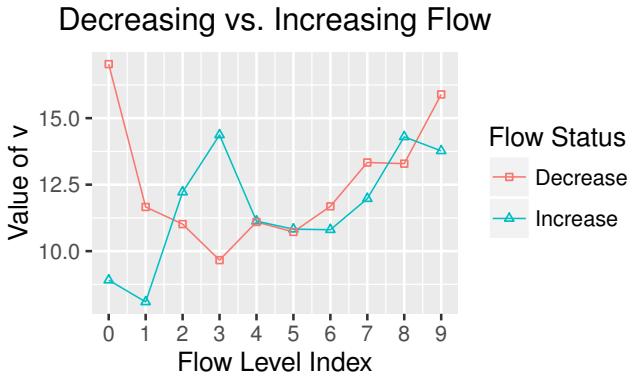

Figure 5. Values of v for increasing vs. decreasing flows.

B. Benchmarking

The benchmark results are shown in Figure 6. When m increases, the error of SARIMA grows from 5.959 to 7.505 which is 25.95%. The error of XGB grows from 5.853 to 6.064 (3.61%). The error of WPT only grows from 5.702 to 5.836 (2.34%). Thus, WPT is more robust. Besides, WPT is giving 0.5% higher accuracy when being compared with the best possible result (5.772) of manually tuned parameter tuple p for any m.

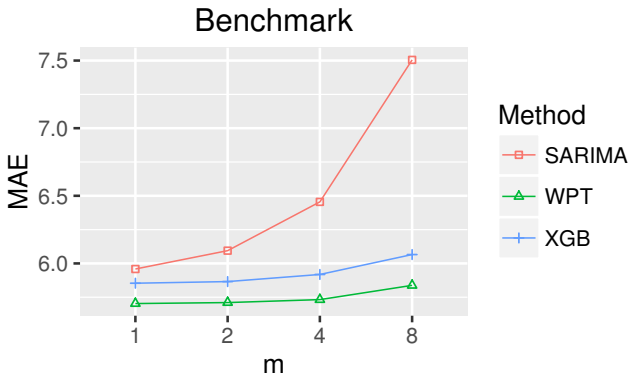

Figure 6. Results of benchmarking WPT with SARIMA and XGB.

VI. Analysis of Results

Previous work used relatively small values for parameters by default (e.g. $k = 8$ or $10, d = 3$ or 4 and $v = 0$), which is hard to produce optimal results.

Considering flow rate (Figure 4), the higher the flow rate the less the neighbours needed. The reason is high flow occurs in holidays and number of holidays are much less than workdays. For d and v, the trend is not monotonous. None of the relations between any of the three parameters and flow rate is linear. Thus, considering different flow levels separately is a good idea, especially when there is enough data to train weights. For incremental and decremental flow, as v has different relations with flow rate, it is necessary to treat incremental and decremental flows separately.

The benchmark results show that WPT gives not only the most accurate but also the most robust result when m changes. Considering all m choices, in average, WPT gives flow rate prediction accuracy 5.744, which is 3.05% improvement when compared with 5.925 given by XGB, and 11.7% improvement in comparison to 6.503 given by SARIMA.

To reduce the calculation time further, we used top 25% of the tuples on each flow level. If the GPU kernels are modified accordingly, it can accelerate prediction stage by four times. The experiment shows it also increases accuracy by discarding bad tuples.

VII. Conclusion

This paper proposes to use WPT to make kNN dynamically tuned regarding dynamic flow rate levels. WPT gives the performance that cannot be achieved using the manual tuning. Besides, WPT is 3.05% better than XGB and 11.7% better than SARIMA. WPT is not only accurate but also space efficient. Only weights are saved which is one hundred kilobyte in the experiment, and one-year history traffic data uses less than one-megabyte space. Additionally, WPT is more robust when predicting multi-step ahead.

One problem we are facing is that the real-world data is usually dirty which has to be handled when designing and implementing WPT algorithm. Cleaning data in the pre-processing stage is not easy since manual registration work and testimony of incident eyewitnesses often lack accuracy. Besides, the distance measurement of neighbours is also important. We plan to investigate these topics further in the future work.

Acknowledgment

We would like to thank reviewers for their valuable suggestions. This work was supported by National Natural Science Foundation of China (NSFC), under grant agreement 61364019.

REFERENCES

[1] A. M. de Souza, R. S. Yokoyama, G. Maia, A. Loureiro, and L. Villas, "Real-time path planning to prevent traffic jam through an intelligent transportation system," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, Jun. 2016, pp. 726–731.

[2] S. An, T. Zhang, X. Zhang, and J. Wang, "Unrecorded Accidents Detection on Highways Based on Temporal Data Mining," *Mathematical Problems in Engineering*, vol. 2014, no. 852495, 2014.

[3] J. Guo, W. Huang, and B. M. Williams, "Real time traffic flow outlier detection using short-term traffic conditional variance prediction," *Transportation Research Part C: Emerging Technologies*, vol. 50, pp. 160–172, Jan. 2015.

[4] S. Oh, Y.-J. Byon, K. Jang, and H. Yeo, "Short-term Travel-time Prediction on Highway: A Review of the Data-driven Approach," *Transport Reviews*, vol. 35, no. 1, pp. 4–32, Jan. 2015.

[5] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, Part 1, pp. 3–19, Jun. 2014.

[6] L. Lin, Q. Wang, and A. W. Sadek, "Short-Term Forecasting of Traffic Volume - Evaluating Models Based on Multiple Data Sets and Data Diagnosis Measures," *Transportation Research Record*, vol. 2392, pp. 40–47, 2013.

[7] R. J. Hyndman and G. Athanasopoulos, "Seasonal ARIMA models," in *Forecasting: Principles and Practice*. oTexts, 2015.

[8] D. Wang, Q. Zhang, S. Wu, X. Li, and R. Wang, "Traffic flow forecast with urban transport network," in *2016 IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, Aug. 2016, pp. 139–143.

[9] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *22nd SIGKDD Conference on Knowledge Discovery and Data Mining*, San Francisco, Aug. 2016.

[10] B. Yu, X. Song, F. Guan, Z. Yang, and B. Yao, "K-Nearest Neighbor Model for Multiple-Time-Step Prediction of Short-Term Traffic Condition," *Journal of Transportation Engineering*, vol. 142, no. 6, p. 04016018, 2016.

[11] J. Moysen, L. Giupponi, and J. Mangues-Bafalluy, "On the potential of ensemble regression techniques for future mobile network planning," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, Jun. 2016, pp. 477–483.

[12] B. Yoon and H. Chang, "Potentialities of Data-Driven Nonparametric Regression in Urban Signalized Traffic Flow Forecasting," *Journal of Transportation Engineering*, vol. 140, no. 7, p. 04014027, Jul. 2014.

[13] H. Hong, W. Huang, X. Xing, X. Zhou, H. Lu, K. Bian, and K. Xie, "Hybrid Multi-metric K-Nearest Neighbor Regression for Traffic Flow Prediction," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 2262–2267.

[14] Z. Zheng and D. Su, "Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm," *Transportation Research Part C-Emerging Technologies*, vol. 43, pp. 143–157, Jun. 2014.

[15] P. Dell'Acqua, F. Bellotti, R. Berta, and A. De Gloria, "Time-Aware Multivariate Nearest Neighbor Regression Methods for Traffic Flow Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3393–3402, 2015.

[16] R. P. Roess, E. S. Prassas, and W. R. McShane, *Traffic Engineering*, 4th ed. Upper Saddle River, NJ: Prentice Hall, Jul. 2010.

[17] S. Innamaa, "Self-adapting traffic flow status forecasts using clustering," *IET Intelligent Transport Systems*, vol. 3, no. 1, pp. 67–76, Mar. 2009.

[18] E. Chung, "Classification of traffic pattern," in *10th World Congress on Intelligent Transport Systems*, vol. 10, Brussels, Nov. 2003, p. 11.

[19] Y. Kamarianakis, H. O. Gao, and P. Prastacos, "Characterizing regimes in daily cycles of urban traffic using smooth-transition regressions," *Transportation Research Part C-Emerging Technologies*, vol. 18, no. 5, pp. 821–840, Oct. 2010.

[20] TRB, *Highway Capacity Manual (HCM)*. Washington, DC: Transportation Research Board, 2010.

[21] F. G. Habtemichael and M. Cetin, "Short-term traffic flow rate forecasting based on identifying similar traffic patterns," *Transportation Research Part C-Emerging Technologies*, vol. 66, pp. 61–78, May 2016.

[22] F. Habtemichael, M. Cetin, and K. Anuar, "Methodology for quantifying incident-induced delays on freeways by grouping similar traffic patterns," in *Transportation Research Board 94th Annual Meeting*, Washington DC, 2015.

[23] B. Everitt, "Euclidean distance," in *The Cambridge Dictionary of Statistics*. Cambridge: Cambridge University Press, 2002, p. 134.

[24] H. Chang, Y. Lee, B. Yoon, and S. Baek, "Dynamic near-term traffic flow prediction: System-oriented approach based on past experiences," *IET Intelligent Transport Systems*, vol. 6, no. 3, pp. 292–305, Sep. 2012.

[25] Y. Qi and B. L. Smith, "Identifying nearest neighbors in a large-scale incident data archive," *Transportation Research Record*, vol. 1879, no. 1, pp. 89–98, 2004.

[26] P. Cai, Y. Wang, G. Lu, P. Chen, C. Ding, and J. Sun, "A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 62, pp. 21–34, Jan. 2016.

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

[28] H. Hong, X. Zhou, W. Huang, X. Xing, F. Chen, Y. Lei, K. Bian, and K. Xie, "Learning Common Metrics for Homogenous Tasks in Traffic Flow Prediction," in *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Dec. 2015, pp. 1007–1012.

[29] S. Moghaddam, R. Noroozi, and B. Hellinga, "Real Time Prediction of Roadway Travel Times for Traveller Information and Proactive Control," in *OTC & CITE Joint Conference, Kitchener, Ontario, Canada*, 2014.

[30] Nvidia, *Cuda Programming Guide 8.0*. Nvidia, Sep. 2016.

[31] R Core Team, *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2016.