Journal of Engg. Research Vol. 8 No. (4) December 2020 pp. 153-168 https://doi.org/10.36909/jer.v8i4.8349

Hybrid Radial Basis Function Neural Networks for Urban Traffic Signal Control

Burcu Caglar Gencosman

Department of Industrial Engineering, Bursa Uludag University, Bursa 16059, Turkey
Corresponding Author: burcucaglar@uludag.edu.tr

Submitted: 20/06/2019
Revised: 01/09/2019
Accepted: 08/09/2019

ABSTRACT

In this study, a real-world isolated signalized intersection with a fixed-time signal control system is considered. The signal timing plans are arranged regardless of the traffic density, and these plans cause delays in vehicle queues. To increase the efficiency of the intersection, an adaptive traffic signal control system is proposed to manage the intersection. To find the appropriate adaptive green times for each lane, simulations are performed by traffic simulation software using vehicle arrivals and other information about vehicle movements gathered from the real-world intersection. Then, a hybrid radial basis function neural network is developed to forecast the adaptive green times, which is trained and tested with historical arrivals and simulation results. The performance of the proposed network is compared with well-known data mining classification methods, such as support vector regression, k-nearest neighbors, decision tree, random forest, and multilayer perceptron methods, by different evaluation parameters. The comparison results provide that the developed radial basis function neural network outperforms other classification methods and can be successfully used for forecasting adaptive green times as an alternative to complex unsupervised classification methods.

Keywords: Adaptive traffic signal control; Data mining classification methods; Radial basis function neural networks; Traffic simulation.

INTRODUCTION

Increased vehicle traffic in urban areas adversely affects the efficiency of traffic flow, resulting in traffic congestion and longer travel times. The adverse effect of a traffic jam is felt even more at intersections. Although local governments have increased their capacities by expanding roads, intersection points are becoming a bottleneck for traffic flow. Therefore, controlling the intersection points with regard to traffic jams could positively affect the efficiency of the traffic flow.

One of the most challenging issues for local governments is the control of intersections, which have the primary importance in the regulation of urban transport. Traffic signal control plans have a noticeable impact on intersection control. Thus, optimal signal control plans are needed to eliminate problems in transportation systems. A signal timing policy that does not meet the needs of the real system increases queuing, delays, the release of harmful gases due to stopping and waiting, and, consequently, the cost of the intersection. However, with a successful signal control system, the efficiency of the traffic flow can be increased. As a result, comfort can also be increased by decreasing the delay. Besides, environmental conditions can be improved by reducing the emission of harmful gases, and the intersection capacity can be used more effectively by improving traffic safety (Grillo and Laperrouze, 2013).

Signal control strategies for intersections are generally classified as either *coordinated* or *isolated* systems (Yu et al., 2018). Coordinated intersections operate with one another to reduce delays. In contrast, signal timing plans at

isolated intersections operate independently, and these intersections do not have any connection. Isolated signal timing systems are summarized in two main categories: *nonadaptive* and *adaptive* signal control systems (Angulo et al., 2011). Nonadaptive traffic signal control corresponds to fixed-time signal control systems, and adaptive traffic signal control systems apply signal timing plans according to changes in the traffic flow. Since real-time monitoring of traffic conditions with developing technology allows reaching the instant real-world data, it would be easier to manage the traffic with adaptive signal control approaches (Angulo et al., 2011).

In this study, a real-world intersection with an isolated signal control system, which is managed by a nonadaptive signal control (fixed-time signalization), is considered. Fixed-time signal control systems operate signal timings with predetermined phase sequences and give priority to vehicle/pedestrian traffic approaching from different directions by predetermined time schedules. Since the vehicle arrivals from lanes at the considered intersection are unbalanced, the predetermined time schedules could cause problems mentioned before. However, in adaptive signal control systems, all of the approach lanes are continuously induced, and the cycle time and red/green times are automatically adjusted according to the traffic density information, which is instantly gathered from real-world by placed sensors to approach lanes. These systems are ideal, in that they minimize the total delays since they provide transition priority and duration by taking the traffic flow density into account. Therefore, the aim of this study is to improve the traffic flow at the intersection by proposing a new adaptive traffic signal control system, which adjusts real-time timing plans considering immediate changes of traffic flow, and dynamically update the green times according to vehicle arrivals and reduce delays in the queue. For this reason, the studies related to this topic in the literature are examined in the next section.

LITERATURE REVIEW

A successful traffic signal control policy should manage the fluctuations in traffic demands, providing real-time control with adaptive characteristics (Liu, 2007). However, the nonlinear nature of traffic control systems has led to traditional methods failing to produce successful solutions. On the other hand, successful results have been obtained in solving nonlinear problems using developing computer technologies and data mining techniques. To solve various problems in transportation systems, data mining clustering and classification methods have become the preferred methods concerning the storage, management, and analysis of real-world data (An et al., 2011; Zhang et al., 2011; Zhu et al., 2018).

Recently, artificial intelligence techniques, one of the most popular data mining tools, have been used successfully in isolated and coordinated systems; the majority of these are neural networks (ANN) (Spall and Chin, 1997; Srinivasan et al., 2006), fuzzy logic, reinforcement learning, and Q-learning applications (Araghi et al., 2015; Aslani et al., 2017; Bazzan, 2009; LA and Bhatnagar, 2011; Li et al., 2009; Liu, 2007). On the other hand, one of the ANN models, the *Radial Basis Function (RBF) neural network*, has been used for various traffic problems, such as traffic volume forecasting (Park et al., 1998; Xie and Zhang, 2006; Zhu et al., 2014), traffic flow prediction (Amin et al., 1998; Chen, 2017; Jun and Ying, 2008; Yang et al., 2010), truck detection (Haj Mosa et al., 2016), and public transport flow prediction (Çelikoğlu and Ciğizoğlu, 2007). Research about the applications in transportation systems reveals that the RBF neural network is mostly used for traffic flow forecasting. However, in this study, the RBF neural network is used to ensure traffic signal control. Although the RBF neural network has been used successfully for various intelligent transportation problems, to the best of our knowledge, it has not been used for predicting green times at an isolated signalized intersection.

To construct an adaptive traffic signal control system by estimating adaptive green times, a hybrid *RBF* neural network is developed; the hidden neuron centers are determined with *K-means,* and the output weights are updated with the *Recursive Least Squares (RLS)* method –it is referred to as the *RBF-RLS* network- (Haykin, 2009; Park et al., 1998; Zhu et al., 2014). Then, the performance of the developed network is compared with well-known classifiers in the literature. Well-known classification methods (henceforth referred to as *classifiers*) include decision tree classifiers, rule-based classifiers, neural networks, and support vector machines (Tan et al., 2005). Table 1 presents some examples of intelligent transportation studies that include classification methods in the literature.

Table 1. Examples of intelligent transportation studies with classification methods.

Study	Problem Type	Classification method
(Sun et al., 2003)	traffic flow prediction	linear regression
(Zhenyu et al., 2013)	traffic speed estimation	linear regression
(Bin et al., 2006; Vanajakshi and Rilett, 2007)	travel time and arrival time prediction	support vector machine
(Xiao and Liu, 2012)	traffic accident detection	support vector machine
(Shi and Abdel-Aty, 2015)	real-time crash prediction	random forest method
(Hu et al., 2016)	short-term traffic flow	particle swarm optimization with support vector regression (PSO-SVR)
(Barbour et al., 2018)	prediction of arrival times of freight traffic on railroads	support vector regression

Although all the considered methods are used in some branches of intelligent transportation systems (traffic flow prediction, traffic speed estimation, traffic accident detection, travel time prediction, etc.), some of the well-known data mining classifiers have not been evaluated in terms of forecasting the green times. Since the proposed RBF-RLS network is compared with some well-known classifiers considering different evaluation performance indices, to the best of our knowledge, this study will be the first to compare the classifiers for forecasting the green times at an isolated signalized intersection. The performance analysis shows that the RBF-RLS network could be a successful method for forecasting adaptive green times at an isolated signalized intersection. In this sense, the performance analysis section will shed light on future studies regarding predicting adaptive green times with classifiers.

The rest of the paper is organized as follows. Section 3 gives the details of the data collection process from the real-world intersection and describes the constructed database. Section 4 summarizes well-known classifiers in the literature and gives the details of the performance parameter indices. Section 5 compares the prediction results of various classifiers with the RBF-RLS network. The study is concluded and future directions are discussed in Section 6.

PROBLEM STATEMENT AND DATA COLLECTION

Problem definition

In this study, a real-world isolated signalized intersection is considered in the province of Kayseri in Turkey, which is managed by a fixed-time signalization. The streets in the intersection are shown in Figure 1. Traffic on Yavuz Sultan Selim and Mehmet Akif Ersoy streets are observed at different times of the day, and it is determined that the traffic flow of Yavuz Sultan Selim Street is more intense than that of Mehmet Akif Ersoy Street. Due to the application of fixed-time signal control plans, the green times are lavish in Mehmet Akif Ersoy Street at certain times of the day, but, on the contrary, the green times in Yavuz Sultan Selim Street are insufficient, causing queues and delays. To eliminate this imbalance, the green times must be adaptively determined according to traffic intensity. For this purpose, an adaptive signal control system needs to be developed as an alternative to the currently used nonadaptive static/fixed-time signal control system. Data collection from the real-world and the development of the database, which includes adaptive green times, are detailed in next subsections.

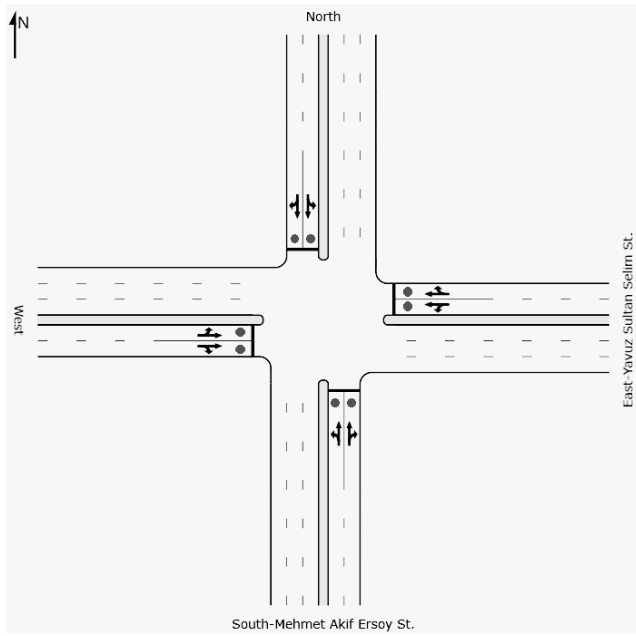

Figure 1. Configuration of the intersection.

Data collection with magnetic sensors

To construct a database, vehicle information is gathered from the lanes, and general parameters related to the intersection are determined by observations. Two major systems collect the arrival information; *image-based* and *sensor-based* systems. Image-based systems provide queue lengths; however, their infrastructure and installation costs are high, and the quality of the data depends on weather conditions. On the other hand, sensor-based systems provide the number of vehicles waiting in the queue regardless of the environmental conditions. Thus, more accurate and constant data can be collected. These sensors have a very low energy requirement (10 years of battery life), are easy to install due to their small size, and can perform wireless communication. Therefore, magnetic sensors are preferred in this study.

The arrivals at the intersection are detected by the magnetic sensors embedded underground in front of the traffic lights at each lane, as shown with dots in Figure 1. The existence of a vehicle is recorded to a database as 0-1 via the wireless communication of sensors and a server located close to the lanes. The information of whether a vehicle is waiting or passing by at the light is analyzed automatically with the help of an algorithm in C#.Net by examining from zero to one and one to zero changing times, which is then converted into the number of vehicle arrivals. Additionally, a program has been developed to detect the number of arrivals to the intersection at the specified time intervals. This program ensures the flexibility of real-time traffic signal control. Since the number of arrivals can be recorded by desired time periods, it is possible to construct a database with *any*-minute arrivals such as 1-minute arrivals or 15-minute arrivals. If the proposed network is trained with 1-minute arrivals, it could be possible to forecast real-time green times by only using the past 1-minute vehicle arrivals so that adaptive green times could be produced with instant real-time vehicle arrival information. Similarly, if the time interval is set as 15 minutes, it could be possible to forecast the green times by using the past 15-minute vehicle arrivals to lanes. The length of the time period depends on the user and/or the management policy of the intersection.

The total number of vehicles arriving at the intersection on different days and hours is calculated using the developed software and recorded to the database. This database is used for the simulation of the adaptive signal control system by Sidra Intersection 5.1 software (SIDRA Solutions, 2019) and detailed in the next subsection.

Traffic simulation for an adaptive system

To determine the adaptive green times, the intersection is considered as a system with full traffic actuated. With the help of actuated signal control (also called adaptive traffic signal control), timing parameters can be dynamically adjusted to respond to real-time traffic arrival changes (Guo et al., 2019). The time interval for arrivals is determined as 60 minutes, and the peak time (peak hour) is calculated as 15 minutes. The lane information and turns are introduced to the simulation, and the intersection geometry is determined. Additionally, the actuated (adaptive) signal analysis method is chosen for the signalization. The green light duration is set to 8-40 seconds, which is determined after conversations with the municipal authorities. The hourly arrivals are then added to the system. The proportion of heavy vehicles is identified as 20%. The peak time factor is the proportion found by the ratio between the number of arrivals during maximum traffic and the maximum number of vehicles arriving at the peak time. The total number of arrivals to the intersection between 18.00-19.00 hours is 1012. The arrivals for the peak hour are examined at 15 minutes intervals, and the maximum number of arrivals is determined to be 275 during 18.00-18.15. The peak hour factor is calculated with these values as 1012/(4*275)=0.92 (Akçelik, 2012). To cover the unit time for the volume (60 minutes), the maximum number of arrivals is gathered by multiplying the 15 minutes interval by 4 to calculate the peak flow factor. The traffic light phases are defined as in Figure 2. As an example, when the green light is on for Street 1, the red light is on for other streets.

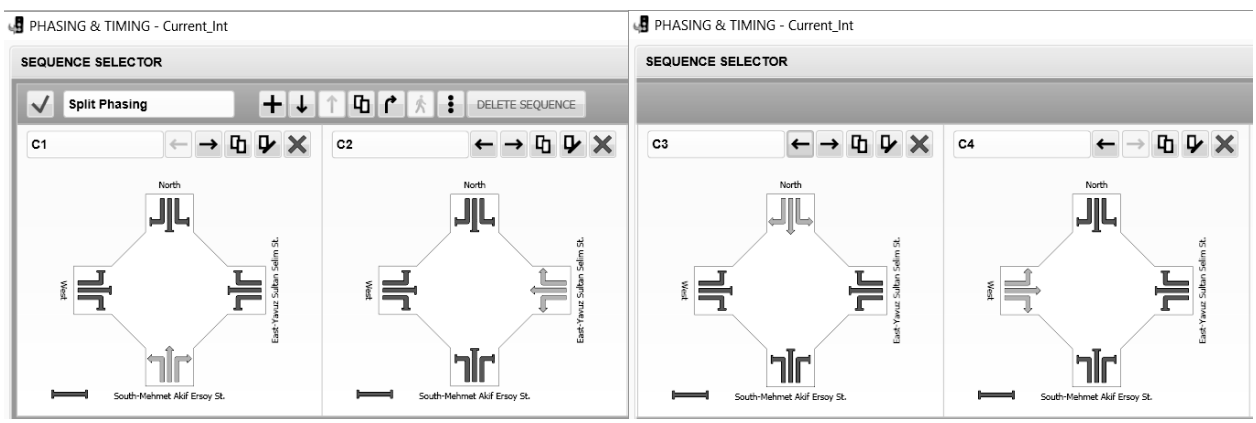

Figure 2. The phases of traffic lights.

Finally, an objective function must be defined for the simulation model. The green times can be determined according to different objective functions (C.K. Wong and Wong, 2003). In this paper, the aim is to minimize the saturation at the intersection. The saturation is calculated by the demand volume/capacity ratio, and it is desirable that this ratio is smaller than one.

Adaptive system simulations are repeated for arrivals recorded between January 2017 and March 2017, and adaptive green times collected from the simulation model are registered to a database. This database includes inputs (vehicle arrivals for each lane) and outputs (adaptive green times for each lane) for a specific time interval. The constructed dataset is provided in (Çağlar Gençosman, 2019). The developed classifiers and the evaluation parameter indices of the methods are detailed in the next section.

METHODOLOGY

Performance evaluation parameters

To test the success of the methods, different performance indices are used. The first performance index, the root mean square error (RMSE), has been commonly used as a standard statistical metric to measure model performance. RMSE is a scale-dependent metric, and it punishes variance by giving errors with larger absolute values more weight than errors with smaller absolute values (Chai and Draxler, 2014) as shown in equation (1). The second performance index is the mean absolute error (MAE), which is also a scale-dependent metric that shows the prediction error by considering the balance between positive and negative errors, as shown in equation (2). The last index is the mean absolute percentage error (MAPE), which is a scale-independent metric that ensures a simple way to define the accuracy, as shown in equation (3) (Chai and Draxler, 2014).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y|\ \ |i-\hat{Y}_i)^2}{n}} \quad (1)$$

$$MAE = \frac{\sum_{i=1}^{n}|Y_i-\hat{Y}_i|}{n} \quad (2)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i-\hat{Y}_i|}{Y_i} \quad (3)$$

where Y_i is the actual measurement; $\hat{Y}_i$ is the predicted value, and n is the number of measurements. Since these parameters represent the prediction error in their own way, the aim of this study is to determine a classifier that ensures the smallest values of evaluation parameters. The proposed classifiers are detailed in the next subsections.

The radial basis function neural networks

Developed by inspiration from the learning process of the human brain and suitable for solving nonlinear problems, artificial neural networks (ANNs) are quite successful for the discovery of the relationship between input and output parameters (Öztemel, 2012). One of the most popular models of ANN is the RBF neural networks (Hornik et al., 1990).

RBF neural networks are three-layer networks that provide feed-forward learning. RBF neural network with local generalization capability and high convergence speed can equally approach any continuous function with uninterrupted accuracy (Zhu et al., 2014). RBF neural network consists of an input layer, a nonlinear hidden layer (radial-basis layer), and a linear output layer. Figure 3 shows an *n-h-m* RBF neural network consisting of n nodes in the input layer, m nodes in the output layer, and h nodes in the hidden layer. The network input vector is $x = (x_1, x_2, \dots, x_n) \in R^n$ and the output vector is $y = (y_1, y_2, \dots, y_m)^T$. The radial-basis activation function of the k hidden node in the network is defined as $\phi_k(.)$ and w_{ik} represents the weight of the output layer (Okkan and Dalkılıç, 2012). In the RBF neural network, there is no transaction between the input layer and the hidden layer, so the inputs are given to the hidden layer without any changes. The information from the input layer is transmitted to the output layer after being processed by the RBF in the hidden layer and then multiplied by the weight values (w_{ik}) of the hidden node k and the output node i. The output values are calculated as in equation (4).

$$y_i = \sum_{k=1}^{N} w_{ik}\phi_k(x, c_k) = \sum_{k=1}^{N} w_{ik}\phi_k(\|x - c_k\|)\ i = 1 \dots m \quad (4)$$

The radial basis centers selected from a subset of the input vector space are presented with $c = (c_1, c_2, \dots, c_n) \in R^n$, and also $\|.\|$ is the distance from the center of the input vector. The neurons in the hidden layer usually calculate the vector distance between the x_k information and c_j usually with the Euclidean criterion $d = \|c_j - x_k\|$, which is evaluated by RBF (Aggarwal, 2018).

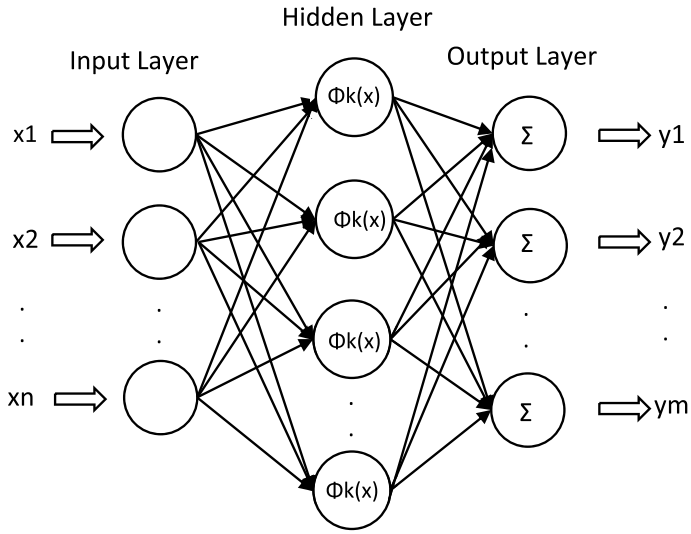

Figure 3. An RBF network.

Many different basis functions are used in RBF neural networks, including *Linear, Cubic, Gaussian, Multiple Quadratic*, and *Inverse Multiple Quadratic* functions. Previous studies reported that the performance of the network does not depend on the choice of basis function (Chen et al., 2007; Gomm and Yu, 2000). In this study, the Gaussian function in equation (5) is used.

$$\phi_k(x) = exp\left(\frac{-\|x-c_k\|^2}{2\sigma^2}\right) \quad (5)$$

The exponential effect of the distance between the center points c_k and the data points x_n, which are supposed to represent the data by RBF, is calculated by equation (5).

Furthermore, the σ standard deviation value represents the dispersion parameter, which significantly influences the performance of RBF neural network. Several methods have been used in the literature to determine the hidden neuron centers (c_k) and update the output weights (w_{ik}). In this study, hidden neuron centers are identified with *K-means,* and output weights are updated with *Recursive Least Squares (RLS)* method (Chen, 1995; Haykin, 2009).

Once the centers are identified, the output of the RBF neural network for a given Q-training set is determined by equation (6). Then, the outputs produced by the network are compared with the expected output, and the weights are updated with the RLS algorithm. The RBF neural network in this study has a hybrid learning procedure due to the use of the *k-means* method for hidden layer training and the *RLS* method for output layer training (henceforth referred to as the RBF-RLS network). The RBF-RLS network is coded in MATLAB 2013 software.

$$y_q = \sum_{k=1}^{N} w_{ik}\phi_k(x(q), c_k) q = 1,2,\dots,Q \quad (6)$$

The proposed RBF-RLS network can be summarized in six steps:

1. Define the input layer. The size of the input layer is determined as 4, which represents the arrivals to each lane at a specific time interval.
2. Define the hidden layer. The size of the hidden layer is determined as 16 after various performance comparisons. It also represents the number of clusters, the parameter *K*. The *K* parameter of the *k-means* algorithm controls the performance and the computational complexity of the proposed method.

3. K-means algorithm computes the mean of clusters, which are also the radial basis centers of the input vectors, presented with $c = (c_1, c_2, \ldots, c_n) \in R^n$.
4. The Gaussian function calculates the vector distance between the x_k information and c_j for the hidden neurons and provides the spread of the centers discovered by the *k-means* algorithm.
5. Define the output layer. The size of the output layer is determined as 4, which represents the adaptive green times for each lane. The training of the output layer can begin after the training of the hidden layer is completed. The function $\phi_k(x)$ represents the outputs of K units in the hidden layer, and the training sample is defined by $\{\phi(x_i), d_i\}_{i=1}^N$ where d_i is the desired response at the overall output of the RBF-RLS algorithm for input x_i. This training is performed by the RLS algorithm, and the output weights (w_{ik}) are updated.
6. Once the network training is completed, test the network with unseen data.

The *epoch* indicates one iteration through the training set. It directly affects the learning quality of the network, which is determined as 200. The experiments are performed for three different values of the learning rate parameter η; η=0.01, η=0.05, and η=0.1. 10-fold cross-validation is used for training and testing the RBF-RLS network. Table 2 shows the experimental results of the RBF-RLS network with different learning rate parameters. The results provide that η=0.05 will be used for comparisons with other classifiers.

It must be mentioned that the prediction quality for four lanes are evaluated individually and calculated RMSE, MAE, and MAPE values for each lane. Then, the average of these performance indices is presented in the following tables. In other words, the performance indices of all classifiers in this study are determined by calculating the average performance results for four outputs.

Table 2. The RBF-RLS network with different learning rate parameters and a 10-fold cross-validation

Classifier	RMSE	MAE	MAPE
RBF-RLS (η=0.01)	0.622	0.318	0.033
RBF-RLS (η=0.05)	0.609	0.305	0.031
RBF-RLS (η=0.1)	0.627	0.328	0.034

Additionally, another RBF neural network method in WEKA, *RBFRegressor*, is used to predict the adaptive green times (Witten et al., 2016). The initial centers are found using *simple k-means*. The usage of attribute weights is activated, and the conjugate gradient descent is used for updates. The *ridge* parameter, which is used to penalize the size of the weights in the output layer, is set to λ=0.01 and λ=0.05. The experimental results with these assumptions, which are summarized in Table 3, show that the *RBFRegressor* method performs better with λ=0.01.

Table 3. The *RBFRegressor* method with different ridge parameters.

Classifier	RMSE	MAE	MAPE
RBFRegressor (λ=0.01)	1.204	0.502	0.046
RBFRegressor (λ=0.05)	1.206	0.506	0.047

It must be mentioned that the database structure differs for the RBF-RLS network and other classifiers. The complete database includes four inputs representing the vehicle arrivals for each lane and four outputs representing the estimated adaptive green times for each lane. The RBF-RLS network has four input neurons in the input layer, one hidden layer, and four output neurons in the output layer. However, this structure cannot be used because of the architecture of the WEKA software, which only allows for one output. To predict the green times for the four

lanes, each classifier is run four times for each lane. Then, the prediction results of the green times for four lanes are evaluated together, and the average values of RMSE, MAE, and MAPE are calculated. In other words, the performance indices of all classifiers in this study are determined by calculating the average performance results for four outputs. Additionally, all the prediction values are rounded to the nearest integer, and the error and other evaluation parameter indices are calculated between the desired value (integer) and the predicted value (integer) for all of the classifiers, including the RBF-RLS network.

Support vector regression

Support vector machines (SVM) can be used for classifying both linear and nonlinear data (Han et al., 2012). SVM transform the original training data into a higher dimension by using nonlinear mapping. Thus, a linear model constructed in the new space can represent a nonlinear decision boundary in the original space (Witten et al., 2016). This method aims to find a special kind of model called the *maximum margin hyperplane*, which separates data into two classes using *support vectors* (Han et al., 2012).

When the data are linearly inseparable, a nonlinear SVM, which is a *quadratic optimization problem*, can be developed by extending the approach for a linear SVM. To solve the quadratic optimization problem, a nonlinear mapping function called the *kernel function* is required. Two frequently used kernel functions for nonlinear mapping are the *radial basis function (RBF) kernel* and the *sigmoid kernel* (Witten et al., 2016). The SVM with the *RBF kernel* is simply a type of an RBF network, and one with the *sigmoid kernel* represents another type of a neural network, specifically a multilayer perceptron (MLP) with one hidden layer. The kernel functions presented in equations (7-9) are used in this study. The formulation of the *polynomial kernel* function of degree *h* is presented in equation (7), and the *RBF kernel* function is presented in equation (8), assuming that X_i and X_j represent *n*-dimensional vectors, and σ is a Gaussian parameter. The *Pearson VII function-based universal kernel* function (*Puk*) is presented in equation (9), where the parameter ω controls the shape of the curve. The details can be found in Üstün et al. (2006).

$$K(X_i, X_j) = (X_i . X_j + 1)^h \tag{7}$$

$$K(X_i, X_j) = exp\left(\frac{-\|X_i - X_j\|^2}{2\sigma^2}\right) \tag{8}$$

$$K(X_i, X_j) = \frac{1}{\left[1+\left(\frac{2\sqrt{\|X_i - X_j\|^2}\sqrt{2^{(1/\omega)}-1}}{\sigma}\right)^2\right]^{\omega}} \tag{9}$$

Although the structure of a maximum margin hyperplane is only suitable for classification, an SVM can also be used for numeric predictions, such as a regression method (called *Support Vector Regression-SVR*). The SVR uses the same iterations with the SVM with some small differences (I. H. Witten, 2016; Üstün et al., 2006).

To develop the SVR method for green time predictions, the *SMOreg* (*Sequential Minimal Optimization for regression*) method in WEKA is used with the following parameter values [41]. The penalty parameter *C* is assumed as *C*=1, *C*=5, and *C*=10 with experiments performed for each value. Additionally, three different kernel functions are selected for kernel function shown in (7-9): the *polynomial kernel* function (*PolyKernel),* the *Gaussian RBF kernel* function *(RBFKernel),* and the *Pearson VII function-based universal kernel* function (*Puk*). Although various algorithms can be used at the learning stage, the *RegOptimizer* is used to select the learning algorithm, and the default algorithm *RegSMOImproved* is used to estimate the green times (Shevade et al., 2000). Since the *SMOreg* method works with one output, the model is developed by four inputs for each arrival to each lane and one output for the green times. Therefore, the model is run four times to predict the green times for each lane with the same inputs (four arrivals for each lane), and the average of the performance parameters is calculated and used for comparisons. The same assumptions will be made for other classifiers in next subsections. Table 4 demonstrates the prediction results of the *SMOreg* method with different penalty parameters and different kernel functions. According to the comparison results,

the minimum RMSE, MAE, and MAPE values are obtained by the *SMOReg* method with the *Puk* kernel function and the *C*=5 penalty parameter. Thus, this structure will be used for comparison with the other classifiers.

Table 4. Comparison results of the *SMOreg* method with different penalty parameters and kernel functions.

Classifier	RMSE	MAE	MAPE
SMOReg (*PolyKernel*, *C*=1)	2.224	1.144	0.096
SMOReg (*PolyKernel*, *C*=5)	2.218	1.157	0.098
SMOReg (*PolyKernel*, *C*=10)	2.220	1.157	0.098
SMOReg (*RBFKernel*, *C*=1)	2.454	1.188	0.094
SMOReg (*RBFKernel*, *C*=5)	2.012	1.002	0.082
SMOReg (*RBFKernel*, *C*=10)	1.716	0.882	0.074
SMOReg (*Puk*, *C*=1)	1.176	0.449	0.041
SMOReg (*Puk*, *C*=5)	1.171	0.442	0.04
SMOReg (*Puk*, *C*=10)	1.175	0.445	0.04

K-nearest neighbors

The *k-nearest neighbors' classifier*, which is an instance-based learning algorithm (Aha et al., 1991), is used for estimating green times. An object is classified by the majority votes of its neighbors, and the object is then assigned to the most common class (*k* is usually a small positive integer) between its nearest neighbors. Nearest-neighbor classifiers learn by comparing the training set and the test set. Each instance in the dataset has *n* attributes that represent a point in an *n*-dimensional space, which also means that all the instances in the training set are stored in the *n*-dimensional pattern space. The *k*-nearest neighbor classifier investigates the space for the *k* training instances (*k*-nearest neighbors) that are closest to the unknown instance (Han et al., 2012). A distance metric can be used to define the "closeness", such as *Euclidean distance* as in equation (10). The distance between $X_1=(x_{11},x_{12},\ldots,x_{1n})$ and $X_2=(x_{21},x_{22},\ldots,x_{2n})$ is

$$dist(X_1,X_2)=\sqrt{\sum_{i=1}^{n}(x_{1i}-x_{2i})^2} \quad (10)$$

If $k = 1$, then the object is only assigned to the class of the nearest neighbor. If the classifier is used for a numeric prediction, it will return a real-valued prediction. In this case, the average value of the real-valued labels associated with the *k*-nearest neighbors is calculated. The *IBk* method is chosen for the k-nearest neighbor algorithm, and the default parameters of WEKA are used with $k = 1$, $k=2$ and $k=3$ (Aha et al., 1991).

Table 5. The *IBk* method with different *k* training instances.

Classifier	RMSE	MAE	MAPE
IBk (*k*=1)	1.209	0.491	0.044
IBk (*k*=2)	1.203	0.490	0.044
IBk (*k*=3)	1.195	0.468	0.041

Table 5 ensures that the best prediction results are obtained with *k*=3 neighbors. Therefore, the *IBk* method will be used within *k*=3 for comparisons with other classifiers.

Model tree

As one of the supervised classification algorithms, the *decision trees* (*model trees*) can be used for both classification and regression problems (Holmes et al., 1999). A decision tree has internal nodes that represent a test on an attribute; branches denote an outcome of the test, the leaf nodes include a class label, and the root node is the topmost node (Han et al., 2012). Decision trees can be transformed into classification rules without requiring any domain knowledge or parameter settings. Different classification models can be used to construct decision trees, such as linear regression, logistic regression, or neural networks (Han et al., 2012). However, model trees allow for building decision trees out of any model, which results in a generic version of decision trees.

Model trees have a similar structure to decision trees, but they use linear functions at the leaves instead of discrete class labels (Quinlan, 1992). The purpose of a model tree is to construct a decision tree hierarchy that complies with several smaller portions of the training set, such that the overall model tree becomes proper to the full training set. The *M5Rules* method in WEKA is chosen to model the tree algorithm, which generates a decision list for regression problems using separate-and-conquer (Holmes et al., 1999; Quinlan, 1992).

Random forest

The *Random forest (RF) algorithm* is a supervised learning algorithm that can be used for both classification and regression problems. It constructs various decision trees and combines these decision trees to obtain a more accurate prediction (Breiman, 2001). The results are obtained from an ensemble of decision trees, and the final result is determined according to a majority vote. Since *RF* searches for the best feature among a randomly selected subset of features, it ensures diversity when growing the trees, which results in a better model. The *RandomForest* method is developed with the default WEKA parameter values (Breiman, 2001). The maximum depth of the tree is determined to be unlimited, and the number of iterations to be performed is 100. The number of randomly chosen attributes is set to 0.

Multilayer perceptron

Neural networks (NN) can be used for both classification and prediction, which are detailed in previous sections. One of the popular learning mechanisms in NN is called *backpropagation*. Backpropagation learns from the comparison between the *desired value* (target) and the *prediction* of the network by processing the training data set. For each training instance, the weights of the network are updated to minimize the mean squared error between the desired value and the predicted value. Since these updates are made in the backward direction from the output layer through each hidden layer down to the first layer, the method is called *backpropagation*. The details can be found in Haykin (2009) and J. Han (2012).

The *MultilayerPerceptron* method with backpropagation in WEKA is used with automatically hidden layers and different parameter values. The activation function between the hidden layers and the output layer is determined as a *sigmoid* function. The *learning rate* is a constant having a value between 0 and 1. The learning rate prevents to be stuck at a local minimum in decision space and supports in finding the global minimum. If the learning rate is too small, the learning will get slow.

On the contrary, if the learning rate is too large, the global optimum will be skipped due to a large oscillation between solutions. As a general assumption, the learning rate can be set as $1/t$, where t indicates the number of iterations through the training set. The learning rate η is assumed as η=0.01, η=0.05, η=0.1, and η=0.2. The *momentum* parameter α is necessary when updating the weights. A small proportion of the updated value from the previous iteration is added to the new weight change. This addition smooths the search process by making changes in directionless abrupt. The momentum parameter is assumed as α=0.1 and α=0.2. The number of *epochs* is determined as 200, which represents one iteration through the training set. The *MultilayerPerceptron* method is constructed with different learning rate and momentum values, and prediction results regarding evaluation parameter indices are presented in Table 6.

Table 6. The *MultilayerPerceptron* method with different learning rate and momentum values.

Classifier	RMSE	MAE	MAPE
MultilayerPerceptron (α=0.1, η=0.01)	1.231	0.559	0.052
MultilayerPerceptron (α=0.1, η=0.05)	1.199	0.512	0.048
MultilayerPerceptron (α=0.1, η=0.1)	1.197	0.506	0.047
MultilayerPerceptron (α=0.1, η=0.2)	1.210	0.529	0.050
MultilayerPerceptron (α=0.2, η=0.01)	1.222	0.549	0.051
MultilayerPerceptron (α=0.2, η=0.05)	1.199	0.511	0.048
MultilayerPerceptron (α=0.2, η=0.1)	1.198	0.507	0.047
MultilayerPerceptron (α=0.2, η=0.2)	1.210	0.528	0.050

Experiments with different combinations of learning rate and momentum parameters ensure that the *MultilayerPerceptron* method provides better green time estimations within α=0.1 and η=0.1 values as in Table 6. This structure will be used for comparing with other methods in the next section.

RESULTS AND DISCUSSION

The RBF-RLS network is compared with other classifiers. To validate the prediction results, 10-fold cross-validation is used for all of the methods. Table 7 presents the comparison results of different classifiers regarding the evaluation parameter indices. The first column shows the methodology, the second column shows the name of the method in WEKA, and the last three columns show the evaluation parameters. According to the comparison results, the RBF-RLS method outperforms other classifiers by minimizing the performance parameters. The radial basis function network in WEKA called *RBFRegressor* performs worse than the RBF-RLS method. In fact, according to the RMSE values, *RBFRegressor* makes the worst predictions out of all the models. The second best-developed method is the SVR algorithm (*SMOreg*). Although the MAE and the MAPE values of *SMOreg* are close to the values of the RBF-RLS, the RMSE value of *SMOreg* is almost twice as poor as the RMSE value of the RBF-RLS. Considering all the evaluation parameters, the *MultilayerPerceptron* method performs worse than the random forest (*RandomForest*), model tree (*M5Rules*), and k-nearest neighbors (*IBk*), which is a surprising result. The rule-based methods, such as model trees and random forest, perform better than a neural network model.

As a result, the RBF-RLS method outperforms the other classifiers. Additionally, it is determined that the support vector regression, *k*-nearest neighbors, random forest, and model tree methods can also be used successfully in adaptive signal control systems. To the best of our knowledge, these methods have not been used before and have not been compared in the field of traffic signal control. The comparison results obtained in this study can be evaluated as a recommendation for future studies on adaptive signal control systems.

Table 7. Comparison results of the different classifiers.

Classifier	WEKA Method	RMSE	MAE	MAPE
RBF-RLS (η=0.05)	---	0.609	0.305	0.031
Support Vector Regression	*SMOreg* (*Puk*, *C*=5)	1.171	0.442	0.040
K-nearest Neighbors	*IBk* (*k*=3)	1.195	0.468	0.041
Model Tree	*M5Rules*	1.199	0.487	0.044
Random Forest	*RandomForest*	1.192	0.471	0.042
Multilayer Perceptron	*MultilayerPerceptron* (α=0.1, η=0.1)	1.197	0.506	0.047
RBF Regressor	*RBFRegressor* (λ=0.01)	1.204	0.502	0.046

CONCLUSION

In this study, a real-world isolated signalized intersection is considered. When the vehicle arrivals at the intersection are examined, the irregularity between vehicle arrivals in terms of the streets and the unstable distribution of arrivals during the day is observed. To correct this imbalance, it is aimed to develop a signal control system that reduces the waiting time in the queue and is updated dynamically according to the number of vehicle arrivals. For this purpose, as an alternative to the current situation, which has been managed by a fixed-time signal control, an adaptive signal control system is developed using traffic simulation software and aimed to provide a real-time/dynamic adaptation of the green times according to the number of vehicle arrivals. The RBF-RLS network was proposed, and its performance was compared with well-known classifiers in the literature. The comparison results showed that the RBF-RLS network outperforms the other classifiers. However, the support vector regression, *k*-nearest neighbors, random forest, and model tree methods can also be successfully used for signal time control. This exploration will contribute to the literature about traffic systems and signal timing control studies.

As an alternative to supervised classifiers, it is possible to develop unsupervised learning methods and compare the performances of different supervised and unsupervised classifiers. In this way, the performance analysis of the data mining techniques for traffic signal control can be strengthened, which may be a guide for further research.

Considering real-world applications, the RBF-RLS network can be easily adapted to the traffic signal control unit and can also be applied as a decision mechanism for green times. With the implementation at the considered intersection in this paper, traffic flow can be regulated, and vehicle delays can be reduced. This method can be extended by collecting accurate data from the relevant intersections and applied in other problematic intersections. Thus, the real-world applications of the method can be expanded, and the positive effects achieved through intersections with adaptive signal control can be increased.

ACKNOWLEDGMENT

The research is supported by the TUBITAK, The Scientific and Technological Research Council of Turkey, under Project No. 7150713. The author is extremely thankful to Kayseri Transportation Department for providing the necessary equipment for the collection of data from the real system, and BITEG Ltd. for transferring the collected data to the database and providing the real-life adaptation of the developed method.

REFERENCES

Aha, D.W., Kibler, D. & Albert, M.K. 1991. Instance-based learning algorithms. Machine Learning, **6**(1): 37–66. doi:10.1007/BF00153759.

Aggarwal, C.C. 2018. Radial Basis Function Networks, in: Neural Networks and Deep Learning. Springer, Yorktown Heights, NY, USA.

Akçelik, R. 2012. SIDRA Intersection user Guide. Australia.

Amin, S.M., Rodin, E.Y., Liu, A.P., Rink, K. & Garcia-Ortiz, A. 1998. Traffic Prediction and Management via RBF Neural Nets and Semantic Control. Computer-Aided Civil and Infrastructure Engineering, **13**(5):315-327. doi:10.1111/0885-9507.00110.

An, S., Lee, B.H. & Shin, D.R. 2011. A Survey of Intelligent Transportation Systems, in: 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks, 32-37. doi:10.1109/CICSyN.2011.76.

Angulo, E., Romero, F.P., García, R., Serrano-Guerrero, J. & Olivas, J.A. 2011. An adaptive approach to enhanced traffic signal optimization by using soft-computing techniques. Expert Systems with Applications, **38**(3):2235-2247. doi:10.1016/J.ESWA.2010.08.011.

Araghi, S., Khosravi, A. & Creighton, D. 2015. A review on computational intelligence methods for controlling traffic signal timing. Expert Systems with Applications, **42**(3):1538-1550. doi:10.1016/J.ESWA.2014.09.003.

Aslani, M., Mesgari, M.S. & Wiering, M. 2017. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. Transportation Research Part C: Emerging Technologies, **85**: 732-752. doi:10.1016/J.TRC.2017.09.020.

Barbour, W., Martinez Mori, J.C., Kuppa, S. & Work, D.B. 2018. Prediction of arrival times of freight traffic on US railroads using support vector regression. Transportation Research Part C: Emerging Technologies, **93**: 211-227. doi:10.1016/j.trc.2018.05.019.

Bazzan, A.L.C. 2009. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. Autonomous Agents and Multi-Agent Systems, 18: 342–375. doi:10.1007/s10458-008-9062-9.

Bin, Y., Zhongzhen, Y. & Baozhen, Y. 2006. Bus Arrival Time Prediction Using Support Vector Machines. Journal of Intelligent Transportation Systems, **10**(4):151-158. doi:10.1080/15472450600981009.

Breiman, L. 2001. Random Forests. Machine Learning, **45**(1):5-32. doi:10.1023/A:1010933404324.

Çağlar Gençosman, B. 2019. Vehicle Arrivals. Available Online:

http://dx.doi.org/10.17632/4w4ws2tsrp.2#file-e5eef725-7c1d-40b9-9399c1cfaec476bc.

Çelikoğlu, H.B. & Ciğizoğlu, H.K. 2007. Modelling public transport trips by radial basis function neural networks. Mathematical and Computer Modelling, **45**(3-4): 480–489. doi:10.1016/J.MCM.2006.07.002.

Chai, T. & Draxler, R.R. 2014. Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. Geoscientific Model Development, **7**: 1247-1250. doi:10.5194/gmd-7-1247-2014.

Chen, D. 2017. Research on Traffic Flow Prediction in the Big Data Environment Based on the Improved RBF Neural Network. IEEE Transactions on Industrial Informatics, **13**(4): 2000-2008. doi:10.1109/TII.2017.2682855.

Chen, S. 1995. Nonlinear time series modelling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning. Electronics Letters, **31**(2): 117-118. https://doi.org/10.1049/el:19950085.

Chen, S., Billings, S.A. & Chent, S. 2007. Neural networks for nonlinear dynamic system modeling and identification. International Journal of Control, **56**(2): 319-346. https://doi.org/10.1080/00207179208934317.

Gomm, J.B. & Yu, D.L. 2000. Selecting radial basis function network centers with recursive orthogonal least squares training. IEEE Transactions on Neural Networks, **11**(3), 306–314. doi:10.1109/72.839002.

Grillo, F. & Laperrouze, J. 2013. Measuring the Cost of Congestion on Urban Area and the Flexible Congestion Rights. Journal of Management and Sustainability, **3**(2). doi:10.5539/jms.v3n2p40.

Guo, Q., Li, L. & Ban, X. 2019. Urban traffic signal control with connected and automated vehicles: A survey. Transportation

Research Part C: Emerging Technologies, 101: 313-334. doi:10.1016/J.TRC.2019.01.026.

Haj Mosa, A., Kyamakya, K., Junghans, R., Ali, M., Al Machot, F. & Gutmann, M. 2016. Soft Radial Basis Cellular Neural Network (SRB-CNN) based robust low-cost truck detection using a single presence detection sensor. Transportation Research Part C, **73**: 105-127. doi:10.1016/j.trc.2016.10.016.

Han, J., Pei, J. & Kamber, M. 2012. Data Mining: Concepts and Techniques, 3rd. ed. Morgan Kaufmann Publishers, Elsevier, USA.

Haykin, S.S. 2009. Neural Networks and Learning Machines, 3rd. ed. Pearson Publishing, Upper Saddle River, NJ, USA.

Holmes, G., Hall, M. & Prank, E. 1999. Generating Rule Sets from Model Trees. Springer, Berlin, Heidelberg, 1–12. doi:10.1007/3-540-46695-9_1.

Hornik, K., Stinchcombe, M. & White, H. 1990. Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks. Neural Networks, **3**: 551-560.

Hu, W., Yan, L., Liu, K. & Wang, H. 2016. A Short-term Traffic Flow Forecasting Method Based on the Hybrid PSO-SVR. Neural Processing Letters, **43**(1): 155–172. doi:10.1007/s11063-015-9409-6.

Jun, M. & Ying, M. 2008. Research of Traffic Flow Forecasting Based on Neural Network, in: 2008 Second International Symposium on Intelligent Information Technology Application, 104-108. doi:10.1109/IITA.2008.207.

LA, P. & Bhatnagar, S. 2011. Reinforcement Learning With Function Approximation for Traffic Signal Control. IEEE Transactions on Intelligent Transportation Systems, **12**(2): 412-421. doi: 10.1109/TITS.2010.2091408.

Li, C., Wang, M., Yang, S.-H. & Zhang, Z. 2009. Urban Traffic Signal Learning Control Using SARSA Algorithm Based on Adaptive RBF Network, in: 2009 International Conference on Measuring Technology and Mechatronics Automation, 658-661. doi:10.1109/ICMTMA.2009.445.

Liu, Z. 2007. A Survey of Intelligence Methods in Urban Traffic Signal Control, IJCSNS International Journal of Computer Science and Network Security, **7**(7):105-112.

Okkan, U. & Dalkılıç, H.Y. 2012. Monthly Runoff Model for Kemer Dam with Radial Based Artificial Neural Networks. Teknik Dergi, **23**(2): 5957-5966.

Öztemel, E. 2012. Yapay Sinir Ağları, third ed. Papatya Publishing, İstanbul, Turkey.

Park, B., Messer, C.J. & Urbanik, T. 1998. Short-Term Freeway Traffic Volume Forecasting Using Radial Basis Function Neural Network. Transportation Research Record: Journal of the Transportation Research Board, 1651, 39-47. doi:10.3141/1651-06.

Quinlan, J.R. 1992. Learning with Continuous Classes, in: Adams, A.; Sterling, L. (Ed.), In 5th Australian Joint Conference on Artificial Intelligence. World Scientific, Australia, 343-348. doi:10.1142/9789814536271.

Shevade, S.K., Keerthi, S.S., Bhattacharyya, C. & Murthy, K.R.K. 2000. Improvements to the SMO algorithm for SVM regression. IEEE Transactions on Neural Networks, **11**(5): 1188-1193. doi:10.1109/72.870050.

Shi, Q. & Abdel-Aty, M. 2015. Big Data applications in real-time traffic operation and safety monitoring and improvement on urban expressways. Transportation Research Part C: Emerging Technologies, **58**: 380-394. doi:10.1016/J.TRC.2015.02.022.

SIDRA Solutions, 2019. Available Online: http://www.sidrasolutions.com/.

Spall, J.C. & Chin, D.C. 1997. Traffic-responsive signal timing for system-wide traffic control. Transportation Research Part C: Emerging Technologies, **5**(3-4): 153-163. doi:10.1016/S0968-090X(97)00012-0.

Srinivasan, D., Choy, M.C. & Cheu, R.L. 2006. Neural Networks for Real-Time Traffic Signal Control. IEEE Transactions on Intelligent Transportation Systems, **7**(3): 261-272. doi:10.1109/TITS.2006.874716.

Sun, H., Liu, H.X., Xiao, H., He, R.R. & Ran, B. 2003. Use of Local Linear Regression Model for Short-Term Traffic Forecasting. Transportation Research Record: Journal of the Transportation Research Board 1836, 143-150. doi:10.3141/1836-18.

Tan, P.N., Steinbach, M. & Kumar, V. 2005. Introduction to Data Mining, first ed. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA.

Üstün, B., Melssen, W.J. & Buydens, L.M.C. 2006. Facilitating the application of Support Vector Regression by using a

universal Pearson VII function based kernel. Chemometrics and Intelligent Laboratory Systems, 81(1): 29-40. doi:10.1016/J.CHEMOLAB.2005.09.003.

Vanajakshi, L. & Rilett, L.R. 2007. Support Vector Machine Technique for the Short Term Prediction of Travel Time, in: 2007 IEEE Intelligent Vehicles Symposium, 600-605. doi:10.1109/IVS.2007.4290181.

Witten, I.H., Frank, E., Hall, M.A. & Pal, C.J. 2016. Data Mining: Practical Machine Learning Tools and Techniques, 2nd. ed. Morgan Kaufmann Publishers, Elsevier, USA.

Xiao, J. & Liu, Y. 2012. Traffic Incident Detection Using Multiple-Kernel Support Vector Machine. Transportation Research Record: Journal of the Transportation Research Board, **2324**(1): 44-52. doi:10.3141/2324-06.

Xie, Y. & Zhang, Y. 2006. A Wavelet Network Model for Short-Term Traffic Volume Forecasting. Journal of Intelligent Transportation Systems, **10**(3): 141-150. doi:10.1080/15472450600798551.

Yang, W., Yang, D., Zhao, Y. & Gong, J. 2010. Traffic flow prediction based on wavelet transform and Radial Basis Function network, in: 2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM), 969-972. doi:10.1109/ICLSIM.2010.5461098.

Yu, C., Feng, Y., Liu, H.X., Ma, W. & Yang, X. 2018. Integrated optimization of traffic signals and vehicle trajectories at isolated urban intersections. Transportation Research Part B: Methodological, **112**: 89-112. doi:10.1016/J.TRB.2018.04.007.

Zhang, J., Wang, F.Y., Wang, K., Lin, W.H., Xu, X. & Chen, C. 2011. Data-Driven Intelligent Transportation Systems: A Survey. IEEE Transactions on Intelligent Transportation Systems, **12**(4): 1624-1639. doi:10.1109/TITS.2011.2158001.

Zhenyu S., Danna Z. & Xia, Y. 2013. Urban road traffic speed estimation for missing probe vehicle data based on multiple linear regression model, in: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), 118-123. doi:10.1109/ITSC.2013.6728220.

Zhu, J.Z., Cao, J.X. & Zhu, Y. 2014. Traffic volume forecasting based on radial basis function neural network with the consideration of traffic flows at the adjacent intersections. Transportation Research Part C: Emerging Technologies, **47**: 139-154. doi:10.1016/J.TRC.2014.06.011.

Zhu, L., Yu, F.R., Wang, Y., Ning, B. & Tang, T. 2018. Big Data Analytics in Intelligent Transportation Systems: A Survey. IEEE Transactions on Intelligent Transportation Systems, **20**(1): 383-398. doi:10.1109/TITS.2018.2815678.