Short-term Traffic Flow Prediction Based on Kernel Learning Methods (Postprint)

Authors: Qiuli Wang, Li Jun

Date: 2018-05-18T00:00:00+00:00

Abstract

Leveraging the powerful nonlinear mapping capabilities of kernel learning, this paper proposes a class of prediction models based on kernel learning methods for short-term traffic flow prediction. Kernel Recursive Least Squares (KRLS), based on Approximate Linear Dependence (ALD) technique, can reduce computational complexity and storage requirements, representing an online kernel learning method suitable for large-scale datasets. Kernel Partial Least Squares (KPLS) projects input variables onto latent variables, extracting latent features by utilizing covariance information between input and output variables. Kernel Extreme Learning Machine (KELM) employs kernel functions to represent the unknown nonlinear feature mapping of the hidden layer, calculates the network' s output weights through a regularized least squares algorithm, and can achieve good generalization performance with extremely fast learning speed. To validate the effectiveness of the proposed methods, KELM, KPLS, and ALD-KRLS are applied to various real-world traffic flow datasets and compared with existing methods under identical conditions. Experimental results demonstrate that both prediction accuracy and training speed are improved across different kernel learning methods, reflecting the application potential of kernel learning approaches in short-term traffic flow prediction.

Full Text

Preamble

Short-term Traffic Flow Forecasting Based on Kernel Learning Methods

Wang Qiuli, Li Jun
(School of Automation & Electrical Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract: Leveraging the powerful nonlinear mapping capabilities of kernel learning, this paper proposes a class of prediction models based on kernel learning methods for short-term traffic flow forecasting. Kernel Recursive Least Squares (KRLS) employs Approximate Linear Dependence (ALD) techniques to reduce computational complexity and memory requirements, making it an online kernel learning method suitable for large-scale datasets. Kernel Partial Least Squares (KPLS) projects input variables onto latent variables, extracting latent features by utilizing covariance information between inputs and outputs. Kernel Extreme Learning Machine (KELM) uses kernel functions to represent the unknown nonlinear feature mapping of hidden layers, with output weights computed through a regularized least squares algorithm, enabling excellent generalization at extremely fast learning speeds. To validate the proposed methods, KELM, KPLS, and ALD-KRLS were applied to various real-world traffic flow datasets and compared against existing methods under identical conditions. Experimental results demonstrate that these kernel learning methods achieve improved prediction accuracy and training speed, highlighting their potential for short-term traffic flow forecasting applications.

Keywords: kernel learning methods; short-term traffic flow; forecasting

0 Introduction

Machine learning methods have achieved numerous successful applications in short-term traffic flow forecasting in recent years. References [3–5] applied neural networks to traffic flow prediction models, while references [6–8] utilized Support Vector Machines (SVM) and Least Squares Support Vector Machines (LSSVM) for short-term traffic flow forecasting. As a critical component of Intelligent Transportation Systems (ITS), traffic flow prediction plays a vital role in urban traffic signal control, congestion management, and route planning [1–2]. Accurate traffic forecasting provides travelers with real-time information to save time and costs while enabling traffic authorities to develop proactive management strategies that enhance urban network efficiency.

Extreme Learning Machine (ELM) [9] randomly selects hidden layer nodes and their parameters in Single-hidden Layer Feedforward Networks (SLFNs), offering extremely fast learning speeds. When the hidden layer mapping is unknown, kernel matrix methods can substitute for it, forming the KELM approach [11]. References [10–13] successfully applied ELM and KELM to time series prediction. Considering feature extraction techniques, references [14, 15] employed PCA-SVM and KPCA-SVM methods that combine SVM with Principal Component Analysis and Kernel Principal Component Analysis, respectively, effectively improving prediction accuracy to some extent.

Reference [16] introduced the KPLS method as a nonlinear extension of linear PLS in high-dimensional feature spaces. For online kernel learning, reference [17] presented the KRLS method, which uses sparsification techniques to limit

kernel matrix size for large-scale dataset training. Reference [18] applied KRLS to time series prediction with promising results. Given the success of SVM, ELM, and related methods in short-term traffic flow forecasting, and considering that research on kernel learning methods like KELM in this domain remains relatively limited, this paper proposes a class of kernel learning prediction methods based on KELM, KPLS, and KRLS to further enhance both prediction accuracy and computational efficiency. To validate the effectiveness of these KRLS, KELM, and KPLS-based models, they were applied to real-world traffic data from various regions and compared against existing methods such as SVM under identical conditions, thereby broadening the application potential of kernel learning methods in intelligent transportation systems.

1 Kernel Learning Methods

Given training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ with $\mathbf{x}_i \in \mathbb{R}^m$ and $\mathbf{y}_i \in \mathbb{R}^n$, we construct matrices $\mathbf{X} \in \mathbb{R}^{N\times m}$ and $\mathbf{Y} \in \mathbb{R}^{N\times n}$. The data are mapped to a high-dimensional feature space via $\phi : \mathbb{R}^m \to \mathcal{F} \subseteq \mathbb{R}^M$. To avoid computational difficulties in high-dimensional spaces, we define a kernel function $k(\cdot,\cdot)$ satisfying Mercer' s condition, yielding the kernel matrix $\mathbf{K}$ with elements $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$.

1.1 KELM Method

KELM extends the ELM method to nonlinear feature spaces. For an SLFN with L hidden nodes, the network output is:

$$\mathbf{y}_i = \sum_{j=1}^{L} \beta_j h_j(\mathbf{x}_i) = \mathbf{h}(\mathbf{x}_i)\beta, \quad i = 1, \dots, N$$

where β_j denotes the weight vector connecting the jth hidden node to the output layer, and $h_j(\mathbf{x}_i) = G(\mathbf{b}_j, \mathbf{w}_j, \mathbf{x}_i)$ represents the output function of the jth hidden node. For RBF hidden nodes, the activation function takes the form $G(\mathbf{b}_j, \mathbf{w}_j, \mathbf{x}_i) = \exp(-b_j\|\mathbf{x}_i - \mathbf{w}_j\|^2)$.

If the SLFN can approximate N training samples with zero error, i.e., $\sum_{i=1}^N \|\mathbf{o}_i - \mathbf{y}_i\| = 0$, there exists β_j, $\mathbf{w}_j$, and $\mathbf{b}_j$ such that:

$$\mathbf{H}\beta = \mathbf{Y}$$

where $\mathbf{H}$ is the hidden layer output matrix. The ith column of $\mathbf{H}$ represents the output vector of the ith hidden node across all inputs, while the ith row corresponds to the hidden layer feature mapping for input $\mathbf{x}_i$. When the number of hidden nodes L is less than the number of samples N, SLFNs can still approximate training samples with minimal error [9]. In this case, $\mathbf{H}$ is not square, so there exists $\widehat{\beta}$ such that:

$$\|\mathbf{H}\hat{\beta} - \mathbf{Y}\| = \min_{\beta} \|\mathbf{H}\beta - \mathbf{Y}\|$$

Thus, unlike conventional feedforward neural networks that adjust all layer weights, ELM only needs to train the output weight matrix β, making its learning process equivalent to solving the least squares solution of equation (4).

To further improve generalization capability and matrix solution stability, we introduce a regularization parameter λ based on ridge regression to constrain the solution for β. The solution to equation (7) can be expressed as:

$$\hat{\beta} = \mathbf{H}^T \left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{Y}$$

Correspondingly, the ELM network output becomes:

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\hat{\beta} = \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{Y}$$

In solving equations (8) and (9), we further consider representing the unknown hidden layer nonlinear feature mapping using kernel functions. By introducing a kernel function satisfying Mercer' s condition and defining $\mathbf{K} = \mathbf{H}\mathbf{H}^T$ with elements $K_{ij} = \mathbf{h}(\mathbf{x}_i)\mathbf{h}(\mathbf{x}_j)^T = k(\mathbf{x}_i, \mathbf{x}_j)$, the KELM network output becomes:

$$f(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ k(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{\lambda} + \mathbf{K} \right)^{-1} \mathbf{Y}$$

1.2 KPLS Method

KPLS treats the high-dimensional feature space as the dual of the original space, establishing a linear PLS regression model in the dual space through kernel transformation. PLS selects latent feature variables based on the covariance between input $\mathbf{X}$ and output $\mathbf{Y}$ matrices. The regression model can be written in matrix form as:

$$\mathbf{Y} = \mathbf{T}\mathbf{B} + \mathbf{F}$$

where $\mathbf{T}$ is the score matrix of latent variables, $\mathbf{B}$ is the regression coefficient matrix, and $\mathbf{F}$ is the residual matrix.

Reference [19] extends linear PLS to high-dimensional nonlinear feature spaces, forming the KPLS method. The algorithm implementation is as follows:

a) Randomly initialize the latent vector $\mathbf{u}$ from the output matrix $\mathbf{Y}$ and define the kernel matrix $\mathbf{K} = \Phi\Phi^T$.

b) Compute the latent vector $\mathbf{t}$ for the input matrix and normalize it:

$$\mathbf{t} = \mathbf{K}\mathbf{u}, \quad \mathbf{t} \leftarrow \frac{\mathbf{t}}{\|\mathbf{t}\|}$$

c) Compute the loading vector $\mathbf{c}$:

$$\mathbf{c} = \mathbf{Y}^T\mathbf{t}$$

d) Compute $\mathbf{u} = \mathbf{Y}\mathbf{c}$ and normalize it:

$$\mathbf{u} \leftarrow \frac{\mathbf{u}}{\|\mathbf{u}\|}$$

e) Repeat steps b)-d) until convergence.

f) Deflate the kernel matrix $\mathbf{K}$ and output matrix $\mathbf{Y}$:

$$\mathbf{K} \leftarrow \left(\mathbf{I} - \mathbf{t}\mathbf{t}^T\right)\mathbf{K}\left(\mathbf{I} - \mathbf{t}\mathbf{t}^T\right), \quad \mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t}\mathbf{c}^T$$

After extracting p latent variables, the regression coefficient matrix $\mathbf{B}$ is obtained from equation (22), yielding the KPLS regression model prediction output on the training dataset as:

$$\hat{\mathbf{Y}} = \mathbf{K}\mathbf{U}\left(\mathbf{T}^T\mathbf{K}\mathbf{U}\right)^{-1}\mathbf{T}^T\mathbf{Y}$$

where the columns of matrix $\mathbf{U}$ consist of vectors $\mathbf{u}_j$, and the columns of matrix $\mathbf{T}$ consist of vectors $\mathbf{t}_j$.

To eliminate bias effects in the regression model, data mapped to the feature space must be centered. The kernel matrices formed by training and test data require the following transformation:

$$\mathbf{K} \leftarrow \mathbf{K} - \mathbf{1}_N\mathbf{1}_N^T\mathbf{K} - \mathbf{K}\mathbf{1}_N\mathbf{1}_N^T + \mathbf{1}_N\mathbf{1}_N^T\mathbf{K}\mathbf{1}_N\mathbf{1}_N^T$$

where $\mathbf{K}_t$ is the uncentered test data kernel matrix of dimension $N_t \times N$, N_t is the test dataset size, $\mathbf{1}_N$ is an N-dimensional column vector of ones, $\mathbf{1}_{N_t}$ is an N_t-dimensional column vector of ones, and $\mathbf{I}$ is the identity matrix.

For time series prediction modeling with single output, the regression coefficient vector α has elements α_i. The KPLS prediction model from equation (17) can be expressed as:

$$\hat{y}_t = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_t, \mathbf{x}_i)$$

If only the first p latent variables are extracted, the KPLS prediction model can be interpreted as a linear PLS regression model in the nonlinear feature space:

$$\hat{\mathbf{y}} = \mathbf{T}_p \mathbf{w}$$

where $\mathbf{T}_p$ contains the first p latent variables (nonlinear principal components), and $\mathbf{w}$ is the weight coefficient vector.

1.3 KRLS Method

During the training phase, assume training data up to time t are available: $\{(\mathbf{x}_i, y_i)\}_{i=1}^{t}$. A dictionary $\mathcal{D}_t$ can be constructed from a subset of training data $\{\mathbf{x}_{j_i}\}_{i=1}^{s_t}$, where s_t is the number of selected training data points. In feature space, these are linearly independent data vectors. When new data $\mathbf{x}_t$ arrives, we determine whether it satisfies the ALD condition. If $\delta_t \leq \nu$, where ν is a threshold parameter affecting sparsity, we find the coefficient vector $\mathbf{a}_t$ that satisfies:

$$\min_{\mathbf{a}_t} \left\| \phi(\mathbf{x}_t) - \sum_{j=1}^{s_{t-1}} a_{t,j} \phi(\mathbf{x}_{j_i}) \right\|^2 \leq \nu$$

Solving equation (23) yields the optimal coefficient vector $\mathbf{a}_t$ and the ALD condition:

$$\delta_t = \min_{\mathbf{a}_t} \left\| \phi(\mathbf{x}_t) - \sum_{j=1}^{s_{t-1}} a_{t,j} \phi(\mathbf{x}_{j_i}) \right\|^2 = k_{tt} - \mathbf{k}_t^T \mathbf{a}_t$$

where $\mathbf{k}_t = [k(\mathbf{x}_t, \mathbf{x}_{j_1}), \ldots, k(\mathbf{x}_t, \mathbf{x}_{j_{s_{t-1}}})]^T$ and $k_{tt} = k(\mathbf{x}_t, \mathbf{x}_t)$. If δ_t is sufficiently small, the approximation becomes:

$$\phi(\mathbf{x}_t) \approx \sum_{j=1}^{s_{t-1}} a_{t,j} \phi(\mathbf{x}_{j_i})$$

Extending to feature space, define matrix $\Phi_t = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_t)]$. The residual component vector is $\phi_t^{\text{res}} = \phi(\mathbf{x}_t) - \sum_{j=1}^{s_{t-1}} a_{t,j} \phi(\mathbf{x}_{j_i})$, where $a_{t,j}$ is the jth element of coefficient vector $\mathbf{a}_t$. The matrix expression becomes:

$$\Phi_t \approx \Phi_{t-1} \mathbf{A}_t$$

where $\mathbf{A}_t$ is a matrix with elements a_{ij}. The kernel matrix $\mathbf{K}_t$ can be approximated as:

$$\mathbf{K}_t \approx \mathbf{A}_t^T \mathbf{K}_{t-1} \mathbf{A}_t$$

where $\mathbf{K}_{t-1}$ is the kernel matrix formed by all s_{t-1} elements in the dictionary.

For single-output time series prediction, the RLS algorithm minimizes the sum of squared errors:

$$L(\mathbf{w}_t) = \sum_{i=1}^{t} (y_i - \mathbf{w}_t^T \phi(\mathbf{x}_i))^2 = \|\mathbf{Y}_t - \Phi_t^T \mathbf{w}_t\|^2$$

where $\mathbf{Y}_t = (y_1, \ldots, y_t)^T$. The optimal solution for $\mathbf{w}_t$ is:

$$\mathbf{w}_t = \left(\Phi_t \Phi_t^T\right)^{-1} \Phi_t \mathbf{Y}_t$$

Substituting equation (26) into (28), we obtain $\mathbf{w}_t = \Phi_t \alpha_t$, where α_t is a vector of s_t "reduced" coefficients. Equation (29) then becomes:

$$L(\alpha_t) = \|\mathbf{Y}_t - \mathbf{K}_t \alpha_t\|^2$$

The minimization solution for equation (30) is:

$$\alpha_t = \mathbf{K}_t^{\dagger} \mathbf{Y}_t$$

The multidimensional input-output ALD-KRLS implementation proceeds as follows:

a) **Training Phase:** Given threshold parameter ν. At time $t = 1$, obtain data $(\mathbf{x}_1, y_1)$, set $\mathcal{D}_1 = \{\mathbf{x}_1\}$, $s_1 = 1$, and compute $P_1 = [1/k_{11}]$.

b) Let $t = t + 1$ and obtain new data $(\mathbf{x}_t, y_t)$.

c) Compute equation (24). If $\delta_t > \nu$, add $\mathbf{x}_t$ to the dictionary, and the dictionary matrix grows accordingly: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t\}$, $s_t = s_{t-1} + 1$. Update $\mathbf{A}_t$:

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Since $\mathbf{x}_t$ is a data vector in the dictionary, we have:

$$\Phi_t = \begin{bmatrix} \Phi_{t-1} & \phi(\mathbf{x}_t) \end{bmatrix}, \quad \mathbf{K}_t = \begin{bmatrix} \mathbf{K}_{t-1} & \mathbf{k}_t \\ \mathbf{k}_t^T & k_{tt} \end{bmatrix}$$

From equation (24), we obtain $\mathbf{a}_t = \mathbf{K}_{t-1}^{-1} \mathbf{k}_t$, and $\mathbf{A}_t$ updates to:

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{a}_t^T & 0 \end{bmatrix}$$

d) If $\delta_t \leq \nu$, update the dictionary as $\mathcal{D}_t = \mathcal{D}_{t-1}$, $s_t = s_{t-1}$. Update $\mathbf{A}_t$:

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{a}_t^T & 0 \end{bmatrix}$$

Update $\mathbf{P}_t$ using the matrix inversion lemma:

$$\mathbf{P}_t = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1}\mathbf{a}_t\mathbf{a}_t^T\mathbf{P}_{t-1}}{1 + \mathbf{a}_t^T\mathbf{P}_{t-1}\mathbf{a}_t}$$

Recompute β_t as:

$$\beta_t = \mathbf{P}_t\mathbf{A}_t^T\mathbf{Y}_t$$

e) Iterate steps 2-4 until all training data are processed.

f) **Testing Phase:** For a trained KRLS model, the prediction output is:

$$\hat{y}_t = \mathbf{k}_t^T\alpha_{t-1}$$

If the number of training data is N, the computational complexity of the KRLS algorithm is $O(Nm^2)$.

Following time series modeling approaches, the different kernel learning methods are applied to short-term traffic flow prediction instances. The established prediction model is:

$$\hat{y}_{t+h} = f(\mathbf{x}_t)$$

where h denotes the prediction horizon and $\mathbf{x}_t = [y_t, y_{t-1}, \ldots, y_{t-m+1}]^T$ is a multi-dimensional input vector constructed from historical time series values, with m as the embedding dimension. For univariate traffic flow time series, training data pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ are constructed, where the model output dimension $n = 1$. The methods KRLS, KPLS, and KELM are then compared against KPCA-SVM and SVM under identical conditions.

2 Short-term Traffic Flow Prediction Examples

The experiments employ Normalized Root Mean Square Error (NRMSE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) as performance metrics. The NRMSE expression is:

$$\text{NRMSE} = \sqrt{\frac{\sum_{t=1}^{N}(\hat{y}_t - y_t)^2}{\text{var}(y_t) \cdot N}}$$

where $\hat{y}_t$ denotes the model' s predicted output, y_t is the actual output, $\text{var}(y_t)$ is the variance, and N is the number of test data points. The Gaussian kernel function used in experiments is:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

where σ is the width parameter of the Gaussian kernel.

2.1 Example 1

This example uses real traffic flow data from a highway observation station in Beijing, with a total observation period of [duration]. The first 285 data groups serve as training data, and the remaining 92 groups as test data. The embedding dimension is set to $m = 4$. Model parameters for different kernel learning methods are determined via cross-validation. KELM, KPLS, KRLS, and KPCA all use Gaussian kernels with width $\sigma = 50$. The KELM regularization parameter is $\lambda = 15$; KPLS uses $p = 10$ latent variables; ALD-KRLS has maximum dictionary capacity $s_{\max} = 200$ and threshold $\nu = 0.1$. For comparison, SVM uses penalty factor $C = 0.5$ with $\epsilon = 0.01$; KPCA-SVM uses 15 nonlinear principal components with a linear SVM kernel. Performance evaluation results are listed in Tables 1 and 2.

shows the 15-minute ahead prediction results for different kernel learning methods, while presents the 30-minute ahead prediction results. The results indicate that KELM, KPLS, and KRLS all achieve good prediction performance, with KRLS showing the best results. [Figure 1: see original paper] and [Figure 2: see original paper] display the prediction curves for 15-minute and 30-minute ahead forecasting on the test set, respectively. The figures demonstrate that all kernel learning methods accurately predict actual traffic flow values with relatively small error fluctuations. [Figure 3: see original paper] shows the MSE convergence curve of the KRLS method on the training dataset, confirming its fast convergence speed.

2.2 Example 2

This example uses traffic flow data provided by a UK transportation agency, sampled at 15-minute intervals during March 2011, with 336 time points selected. The embedding dimension is $m = 3$. The first 235 data points serve as training input, and the last 100 as test samples.

Parameter settings are as follows: KELM, KPLS, KRLS, KPCA, and SVM all use Gaussian kernels with width $\sigma = 100$. The KELM regularization parameter

is $\lambda = 20$; KPLS uses $p = 10$ latent variables; ALD-KRLS has maximum dictionary capacity $s_{\max} = 90$ and threshold $\nu = 0.1$. For SVM comparison, the penalty factor is $C = 0.5$ with $\epsilon = 0.01$; KPCA-SVM uses 10 nonlinear principal components with a linear SVM kernel.

[Figure 4: see original paper] and [Figure 5: see original paper] compare the prediction results on the test set for 15-minute and 30-minute ahead single-step and multi-step predictions, respectively. The results show that the proposed kernel learning methods exhibit good prediction performance. Performance metrics for 15-minute and 30-minute ahead predictions are given in and . Compared with SVM, KELM shows slightly improved prediction accuracy, KPLS performs moderately, while KRLS demonstrates significantly higher prediction precision. Convergence time comparisons reveal that KELM and KRLS converge quickly, KPLS trains slower, and ALD-KRLS, with its adaptive online training characteristics, achieves the fastest convergence and highest prediction accuracy.

3 Conclusion

This paper proposes kernel learning prediction models based on KELM, KPLS, and KRLS for short-term traffic flow forecasting. The KELM method replaces the random mapping of traditional ELM with kernel mapping, improving network generalization and learning speed. The KPLS method offers strong feature extraction capability and easy algorithmic implementation. The KRLS method features time-varying adaptive learning characteristics and controls kernel matrix size through sparsification algorithms, thereby improving computational efficiency. Experimental results demonstrate that applying different kernel learning methods to short-term traffic flow prediction is a feasible and effective approach. Additionally, while KELM and KPLS are suitable for offline training, their limitation lies in computational difficulties with kernel matrices when training datasets exceed certain scales. In contrast, KRLS is appropriate for online training, making it applicable to large-scale datasets.

References

[1] Lippi M, Bertini M, Frasconi P. Short-term traffic flow forecasting: an experimental comparison of time-series analysis and supervised learning [J]. IEEE Trans on Intelligent Transportation Systems, 2013, 14(2): 124-132.

[2] Sun S, Zhang C, Yu G. A bayesian network approach to traffic flow forecasting [J]. IEEE Trans on Intelligent Transportation Systems, 2006, 7(1): 124-132.

[3] Messer C, Thomas U I. Short-term freeway traffic volume forecasting using radial basis function neural network [J]. Transportation Research Record: Journal of the Transportation Research Board, 1998, 1651(1): 45-52.

[4] Li J, Huang J. Application of a local autoregressive method based on self-organizing mapping neural network in network traffic prediction [J]. Information and Control, 2016, 45(1): 120-128.

[5] Xu R, Zhou D, Jiang S, et al. Adaptive particle swarm neural network traffic flow prediction model [J]. Journal of Xi' an Jiaotong University, 2015, 49(10): 103-108.

[6] Zhang Y L, Xie Y C. Forecasting of short term freeway volume with support vector machines [J]. Transportation Research Record, 2007, 12(2): 12-20.

[7] Chen X, Liu X, et al. Research on short-term traffic flow prediction of road network based on GA-LSSVR model [J]. Journal of Transportation Systems Engineering and Information Technology, 2017, 17(1): 60-73.

[8] Shang Q, Yang Z, et al. Short-term traffic flow prediction based on singular spectrum analysis and CKF-LSSVM [J]. Journal of Jilin University: Engineering and Technology Edition, 2016, 46(6): 1792-1798.

[9] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: a new learning scheme of feedforward neural networks [C]// Proc of IEEE International Joint Conference on Neural Networks. 2004, 2: 985-990.

[10] Shang Q, Yang Z, Li Z, et al. Short-term traffic flow prediction based on phase space reconstruction and RELM [J]. Journal of South China University of Technology: Natural Science Edition, 2016, 44(4): 109-114.

[11] Huang G B, Wang D H, Lan Y. Extreme learning machines: a survey [J]. International Journal of Machine Learning and Cybernetics, 2011, 2(2): 107-122.

[12] Li J, Li D. Wind power time series prediction based on optimized kernel extreme learning machine [J]. Acta Physica Sinica, 2016, 65(13): 33-42.

[13] Ma C, Zhang Y. Online kernel extreme learning machine and its application in time series prediction [J]. Information and Control, 2014, 43(5): 624-629.

[14] Chen Q, Chen X, Wu Y. Optimization algorithm with kernel PCA to support vector machines for time series prediction [J]. Journal of Computers, 2010, 5(3): 301-308.

[15] Sun Z, Fox G. Traffic flow forecasting based on combination of multidimensional scaling and SVM [J]. International Journal of Intelligent Transportation Systems Research, 2014, 12(1): 20-25.

[16] Rosipal R, Trejo L. Kernel partial least squares regression in reproducing kernel Hilbert space [J]. Journal of Machine Learning Research, 2001, 2(2): 97-123.

[17] Engel Y, Mannor S, Meir R. The kernel recursive least-squares algorithm [J]. IEEE Trans on Signal Processing, 2004, 52(8): 2275-2285.

[18] Fan H, Song Q. A sparse kernel algorithm for online time series data prediction [J]. Expert Systems with Applications, 2013, 40(6): 2174-2181.

[19] Li J, Dong H. Chaotic system modeling based on wavelet kernel partial least squares regression [J]. Acta Physica Sinica, 2008, 57(8): 4756-4765.

Note: All figure and table markers ([FIGURE:N], [TABLE:N]) have been preserved exactly as in the original text. Mathematical equations and notation remain unchanged to maintain technical accuracy. The translation prioritizes readability while preserving the original academic structure and content.

Note: Figure translations are in progress. See original paper for figures.