



A novel double incremental learning algorithm for time series prediction

Jinhua Li¹ · Qun Dai¹ · Rui Ye¹

Received: 5 September 2017 / Accepted: 13 March 2018 / Published online: 17 March 2018
© The Natural Computing Applications Forum 2018

Abstract

Based on support vector machine (SVM), incremental SVM was proposed, which has a strong ability to deal with various classification and regression problems. Incremental SVM and incremental learning paradigm are good at handling streaming data, and consequently, they are well suited for solving time series prediction (TSP) problems. In this paper, incremental learning paradigm is combined with incremental SVM, establishing a novel algorithm for TSP, which is the reason why the proposed algorithm is termed double incremental learning (DIL) algorithm. In DIL algorithm, incremental SVM is utilized as the base learner, while incremental learning is implemented by combining the existing base models with the ones generated on the new data. A novel weight update rule is proposed in DIL algorithm, being used to update the weights of the samples in each iteration. Furthermore, a classical method of integrating base models is employed in DIL. Benefited from the advantages of both incremental SVM and incremental learning, the DIL algorithm achieves desirable prediction effect for TSP. Experimental results on six benchmark TSP datasets verify that DIL possesses preferable predictive performance compared with other existing excellent algorithms.

Keywords Time series prediction (TSP) · Incremental SVM · Incremental learning · Double incremental learning (DIL) algorithm

1 Introduction

In the past few decades, time series prediction (TSP) has been a challenging problem in machine learning. Time series forecasting is an effective means for assessing the characteristics of dynamic systems and predicting trends in complex systems. Moreover, with the development of TSP theory, its application in real life becomes more and more extensive. In recent years, TSP has been increasingly applied in many fields, such as traffic flow forecasting [1], cargo sales forecasting [2], sunspot prediction [3] and stock market forecasting [4].

Time series is defined as a vector formed by data recorded at the same time interval. The general process of TSP can be divided into three steps: (1) collect historical

data; (2) design one model to study the characteristics of the data; and (3) use the model to predict future data. Among them, the second step is the most critical step.

In the early days, the common approaches for TSP were some traditional statistical models, such as exponential smoothing (ES), AutoRegressive integrated moving average (ARIMA) model and AutoRegressive conditional heteroskedasticity (ARCH), etc. [5]. However, these statistical models are only suitable to time series with linear features and cannot be applied to many complex systems with nonlinear features in the real world. Therefore, these models have great limitation in practical applications.

In the past few decades, the theories of machine learning have been gradually applied to the field of TSP, such as artificial neural networks (ANNs) [6–11], evolutionary computation [12], feed-forward neural networks (FNNs) [13–15] and support vector machine (SVM) [16–18]. Out of numerous methods, ANNs have strong learning and generalization ability, which have been in the leading position of TSP. The well-known ANNs-based models are fuzzy neural networks [9, 10], recurrent neural networks

✉ Qun Dai
daiqun@nuaa.edu.cn

¹ College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

[7, 11], wavelet neural networks [8] and radial basis function (RBF) neural networks [6].

Although ANNs have many advantages, they still have some drawbacks, such as longer training periods and being easy to fall into local optimal traps. Moreover, hidden layer sizes and learning rates are also difficult to determine. These are issues that affect the generalization capacity of ANNs and are difficult to avoid [19]. However, these problems can be solved by using SVM in conjunction with statistical theory and structural risk minimization criteria.

SVM is a powerful nonlinear algorithm, having important application in many fields of scientific research. It is capable of generating nonlinear discriminant boundaries through linear classifiers, while still has simple geometric explanations. The original SVM was only applied to classification problems. With the development of theories, support vector regression (SVR) was proposed [20], so that SVM is applicable to the field of TSP. SVR is able to effectively solve high-dimensional and complicated regression problems [21], making it promising for TSP.

Ma and Laskov et al. proposed incremental SVM on the basis of SVM [22, 23], which inherits the advantages of SVM, and these advantages will be described in detail below. In addition, incremental SVM learns new data by modifying the trained SVM model rather than retraining a model, so it is better at handling streaming data, which are constantly changing over time. While time series is a kind of typical streaming data, therefore, compared with SVM, incremental SVM is more suitable for TSP. Particularly, incremental SVM avoids the repetitive training of large numbers of samples when processing stream data, so its efficiency is much higher than SVM.

The most important factors affecting the generalization performance of incremental SVM are kernel functions and their parameters. There exist several widely used kernel functions, such as RBF kernel, polynomial kernel, linear kernel, and sigmoid kernel. The RBF kernel and polynomial kernel are always able to satisfy Mercer's theory, while other kernel functions are in a certain condition to meet the theory [24]. Since RBF kernel function can reduce the computational complexity and improve the generalization performance of models, it is adopted in the algorithm proposed in this paper.

Furthermore, we find that the combination of incremental learning paradigm and incremental SVM can further boost the performance for TSP, which motivates the proposal of the double incremental learning (DIL) algorithm in this work. Incremental learning was proposed firstly by Cauwenberghs et al. [25], which enables the algorithm to revise the previously generated model based on the new data points. The idea of incremental learning is to iteratively modify the effect of the new data point on the regression function to find its Kuhn–Tucker condition,

while simultaneously keep the previously trained data points satisfying the Kuhn–Tucker condition. The method iteratively and appropriately modifies the model when a new data point is input into the generated incremental SVM, rather than retraining the model from scratch. Although it is originally proposed for classification problems, incremental learning is also well suited to solve the problems of regression [23].

With regard to the definition of incremental learning, different literatures give different definitions [26–30]. In this paper, we adopt a universally accepted concept of incremental learning that satisfies the following conditions [29, 30]:

1. It is capable of learning new knowledge from new data.
2. Old data used for the existing models are not necessary when training a new model.
3. Knowledge obtained previously could be preserved.
4. It should be able to accommodate the changes in the characteristics of the new data.

Until now, a variety of incremental learning algorithms has been proposed to solve a variety of different problems. In some cases, incremental learning refers to the growing or pruning of model architectures [31–34]. In other cases, some forms of controlled modification of learner weights have been proposed, which are ordinarily implemented by retraining the samples with large prediction errors [35–38]. Though algorithms introduced above can absorb additional knowledge from new data, it is hard for them to simultaneously meet all the four above-mentioned conditions of incremental learning. They either need to access the previous original data, or are unable to retain the previously obtained knowledge, or cannot adapt to the changes of the attributes of new data.

Since ensemble learning usually has preferable performance in comparison with single classifiers, we incorporate it into our proposed algorithm to obtain the final predictive values. The classical ensemble learning paradigm is divided into two stages, i.e., the generation of base models and the combination of their decisions [39, 40]. What's more, many theoretical and experimental studies in the literature have confirmed that, when the dataset is properly divided into several subsets, compared to using the entire dataset, using each subset as the training set to generate a component model for the ensemble and integrating the decisions of the ensemble components often achieves better or, at least, similar generalization performance [41]. Later, we would analyze this from the perspective of TSP.

TSP is a research field with high practical value. For example, the forecasting of network access traffic allows the company to dispatch resources in a timely manner, so as to prevent a large number of concurrent visits, which

might cause the paralysis of the website. For another example, by forecasting the sales of goods, it is possible to determine the future purchase to prevent the goods from encountering poor sales or out of stock. For still another example, investors may be able to get the maximum profit by properly predicting the stock movements. These practical applications have greatly promoted the study of TSP. However, the existing algorithms exposed some problems in practice. Therefore, double incremental learning (DIL) algorithm is proposed in this work, with the motivation being to improve the prediction effect, and to further promote the application of TSP in practice. DIL integrates incremental learning paradigm together with incremental SVM, which is where the algorithm name, i.e., double incremental learning (DIL) algorithm, comes from.

The DIL algorithm proposed in this work satisfies all of the above-mentioned four conditions. Besides, DIL differs from the traditional incremental learning algorithms in that, it generates new base models for the unknown parts of the feature space, instead of generating new nodes for each previously unknown instance. This scheme is similar to the rationale of ensemble learning paradigm, which significantly improves the performance of DIL algorithm in TSP.

Specifically, the dataset is preprocessed firstly and then divided into several appropriate subsets. For each subset of the dataset, weights are assigned for each sample, and the training and testing subset are selected according to the weights. Based on the training subset, a base model is obtained by implementing the incremental SVM algorithm. The weight of the base model is set according to its prediction error, and then the weighted majority voting rule is used to combine the generated base models to get a composite model. Finally, weights of the samples are adjusted in terms of the prediction error of the composite model. The above steps are repeated until a sufficient number of base models are obtained, and then, the final composite model is achieved by integrating all the base models using the weighted majority voting rule.

Next, the advantages, innovations and contributions of the proposed DIL algorithm will be introduced from several ways.

First of all, in the DIL algorithm, a new sample weight update rule is proposed, which updates the weights based on the performance of the composite model produced so far, rather than updates the weights based on the performance of the base models. Therefore, the update to the weights is more reasonable, and this rule is conducive to improving the generalization performance and robustness of the model.

In addition, the weighted majority voting method, which allocates the corresponding weight of the base model according to the performance of each base model, is used in the base model integration. Therefore, the base model

with poor performance has lower discourse power, which allows the integrated model has better prediction performance.

Finally, since DIL has combined incremental learning paradigm with incremental SVM, it inherits several advantages from both of them. Incremental learning mainly brings two advantages to DIL. The first is that, it does not need to save historical data and can save the storage space. The second is that, it learns new data incrementally and makes full use of the learned knowledge without retraining the whole model, thus reducing learning time and improving learning efficiency.

DIL also inherits three major characteristics from incremental SVM. The first is its excellent generalization performance. The optimization goal of incremental SVM is to achieve the smallest structural risk, rather than the least empirical risk. Therefore, compared with many other excellent algorithms, it possesses better generalization performance. The second is the higher learning efficiency and better robustness, which are mainly benefited from the support vectors. The last point is, since incremental SVM itself has the characteristics of incremental learning, it has better generalization performance while processing time series data. Therefore, incremental SVM further enhances the overall performance of the DIL algorithm in handling TSP issues.

The numerical experiments are conducted based on six benchmark time series datasets, i.e., Mackey–Glass, Lorenz, Sunspot, Nikkei 225 Index (N225), Dow Jones Industrial Average Index (DJI), and Shanghai Stock Exchange Composite Index (SSE) datasets, to evaluate the effectiveness of the proposed DIL algorithm. The predictive performance of DIL is compared with some other excellent algorithms reported in the literature. From the comparison results, it can be concluded that DIL is superior to these comparative algorithms.

The rest of the paper is organized as follows. In Sect. 2, the required theoretical knowledge about SVM and incremental SVM will be described in detail. Section 3 will cover the details of the proposed DIL algorithm. The experimental results on the six benchmark time series datasets are reported in Sect. 4. Finally, in Sect. 5, the conclusions and outlook for future works are given.

2 Theoretical basis

SVM was originally a powerful algorithm for solving classification problems, which was proposed by Vapnik et al. [42]. With the development of relevant theories, Vapnik et al. proposed a kind of SVM for solving regression problems, i.e., SVR [43]. On the basis of SVM, incremental SVM is presented [22, 23], which inherits the

power of SVM. Furthermore, the performance of incremental SVM is more excellent for regression problems. Therefore, incremental SVM is implemented as the base learner of the proposed DIL algorithm. In the following of this section, the principle of incremental SVM is introduced.

Let's consider the regression problem. Assume that the training set is $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, where $(\mathbf{x}_i, y_i), i = 1, \dots, m$ are training samples, $\mathbf{x}_i, i = 1, \dots, m$ are feature vectors, and each element in the feature vectors is a real number; $y_i \in \mathbb{R}, i = 1, \dots, m$ represent the target values. m indicates the size of the dataset \mathbf{D} , that is, the number of training samples. The goal of learning is to get the model shown in Eq. (1):

$$f(\mathbf{x}) = \boldsymbol{\omega}^T \mathbf{x} + b, \tag{1}$$

where $\boldsymbol{\omega}$ and b are the model parameters. The output of the model $f(\mathbf{x})$ should be as close as possible to y . Similarly, \mathbf{x} is a feature vector and y represents the target value.

The above model is proposed in the original feature space, but in practice, usually the kernel function is used to map the original space to high-dimensional space to facilitate the solution. Let $\phi(\mathbf{x})$ denote the eigenvector after mapping \mathbf{x} to high-dimensional space, the model corresponding to Eq. (1) is as follows:

$$f(\mathbf{x}) = \boldsymbol{\omega}^T \phi(\mathbf{x}) + b. \tag{2}$$

According to literature [43], the solution of SVR is obtained as follows:

$$f(\mathbf{x}) = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \kappa(\mathbf{x}, \mathbf{x}_i) + b, \tag{3}$$

where $\kappa(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x})^T \phi(\mathbf{x}_i)$ is the kernel function, $\hat{\alpha}_i, \alpha_i$ are Lagrange multipliers.

According to Eq. (3), we can make the following marks:

$$v_i = \hat{\alpha}_i - \alpha_i \tag{4}$$

$$h(\mathbf{x}_i) = f(\mathbf{x}_i) - y_i = \sum_{j=1}^m \kappa_{ij} v_j - y_i + b \tag{5}$$

According to the value of v , the training dataset can be divided into the following three subsets:

$$\begin{aligned} \mathbf{S} &= \{\mathbf{x}_i | 0 < |v_i| < C\} \\ \mathbf{E} &= \{\mathbf{x}_i | |v_i| = C\} \\ \mathbf{R} &= \{\mathbf{x}_i | |v_i| = 0\} \end{aligned} \tag{6}$$

Assume that sample \mathbf{x}_c be a sample newly added to the training set. Since the elements in set \mathbf{S} satisfy $\Delta h(\mathbf{x}_i) = 0$, the following equations can be obtained:

$$\Delta h(\mathbf{x}_i) = \kappa_{ic} \Delta v_c + \sum_{\mathbf{x}_j \in \mathbf{S}} \kappa_{ij} \Delta v_j + \Delta b = 0, \quad \forall \mathbf{x}_i \in \mathbf{S} \tag{7}$$

$$\Delta v_c + \sum_{\mathbf{x}_j} \Delta v_j = 0 \tag{8}$$

Assume that set $\mathbf{S} = \{\mathbf{x}_{s_1}, \mathbf{x}_{s_2}, \dots, \mathbf{x}_{s_l}\}$, then the incremental value of v corresponding to the element in set \mathbf{S} should meet the following equation:

$$\mathbf{H} \begin{pmatrix} \Delta b \\ \Delta v_{s_1} \\ \vdots \\ \Delta v_{s_l} \end{pmatrix} = - \begin{pmatrix} 1 \\ \kappa_{s_1} \\ \vdots \\ \kappa_{s_l} \end{pmatrix} \Delta v_c \tag{9}$$

where

$$\mathbf{H} = \begin{pmatrix} 0 & 1 & \dots & 1 \\ 1 & \kappa_{s_1 s_2} & \dots & \kappa_{s_1 s_l} \\ \vdots & \vdots & & \vdots \\ 1 & \kappa_{s_l s_1} & \dots & \kappa_{s_l s_l} \end{pmatrix} \tag{10}$$

From Eq. (9), the following equations can be obtained:

$$\Delta b = \eta \Delta v_c \tag{11}$$

$$\Delta v_j = \eta_j \Delta v_c, \quad \forall \mathbf{x}_j \in \mathbf{S} \tag{12}$$

where η and η_j can be acquired by the following formula:

$$\begin{pmatrix} \eta \\ \eta_{s_1} \\ \vdots \\ \eta_{s_l} \end{pmatrix} = -\mathbf{H}^{-1} \begin{pmatrix} 1 \\ y_{s_1} y_c \kappa_{s_1 c} \\ \vdots \\ y_{s_l} y_c \kappa_{s_l c} \end{pmatrix} \tag{13}$$

For samples \mathbf{x}_j 's that are not in set \mathbf{S} , we can get $\eta_j = 0 (\forall \mathbf{x}_j \notin \mathbf{S})$. For samples in sets \mathbf{R} and \mathbf{E} , $\Delta h(\mathbf{x}_i)$ can be obtained from:

$$\Delta h(\mathbf{x}_i) = \left(\kappa_{ic} + \sum_{\mathbf{x}_j \in \mathbf{S}} \kappa_{ij} \Delta \eta_j + \eta \right) \Delta v_c = \gamma_i \Delta v_c \tag{14}$$

According to the principle of asymptotic movement, the value of Δv_c can be calculated in four cases, and the maximum one is taken as the final value of Δv_c . Since the detailed calculation method of Δv_c does not fall into the focus of this paper, we will not elaborate it here. When a new sample \mathbf{x}_c is added to the set \mathbf{S} , the matrix $\boldsymbol{\Psi} = \mathbf{H}^{-1}$ should be updated as follows:

$$\boldsymbol{\Psi} = \begin{pmatrix} & & & 0 \\ & \boldsymbol{\Psi} & & 0 \\ & & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} + \frac{1}{\gamma_c} \begin{pmatrix} \eta \\ \eta_{s_1} \\ \vdots \\ \eta_{s_l} \\ 1 \end{pmatrix} (\eta \quad \eta_{s_1} \quad \dots \quad \eta_{s_l} \quad 1) \tag{15}$$

3 Methodology

3.1 Base learner

The base learner is the cornerstone of DIL. Although the selection to the base learner is varied, it is necessary to select the appropriate base learner according to the specific problem. In this paper, incremental SVM is chosen as the base learner. We make such a choice mainly based on the following two considerations. First, incremental SVM inherits several advantages from SVM, such as good

generalization performance and high robustness. Second, incremental SVM can learn new data incrementally, which makes it more suitable for TSP, as previously mentioned.

In DIL, it is necessary to implement incremental SVM to generate a base model in each iteration. The final composite model is obtained by combining all the generated base models. Based on the theoretical analysis in the previous section, the pseudocode of incremental SVM can be gained, as shown in Algorithm 1.

Algorithm 1 Incremental SVM

Input: Dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$

Output: The model for prediction

- (1): Read the new instance x_c , calculate $h_c = h(x_c)$.
 - (2): If $|h_c| > \varepsilon$ & $|v_c| < C$, then continue to the next step, otherwise terminate the algorithm.
 - (3): Calculate η and γ according to Eq. (13) and Eq. (14).
 - (4): Calculate the maximum value of Δv_c : Δv_c^{\max} .
 - (5): Update the value of v_c : $v_c = v_c + \Delta v_c^{\max}$.
 - (6): Update the instances in set S : $v_s = v_s + \eta \Delta v_c^{\max}$.
 - (7): Update the examples in set R and E : $h_{c,r} = h_{c,r} + \gamma \Delta v_c^{\max}$.
 - (8): Suppose x_d is the sample that determines the maximum value of Δv_c .
 - If $x_d \in S$, then move x_d from set S to set R or E .
 - If $x_d \in R \cup E$, then move x_d from set R or E to set S .
 - If $x_d = x_c$, then terminate the algorithm.
 - (9): Update $\Psi = H^{-1}$ according to Eq. (15), then go to step (2).
-

3.2 The proposed DIL algorithm

In this section, a detailed description of the proposed DIL algorithm is presented. In the DIL algorithm, incremental learning is implemented by combining the existing base models with the base models generated on the new data. As mentioned previously, DIL inherits some merits from both incremental SVM and incremental learning; therefore, it is particularly suitable for time series forecasting. Moreover, the strategy of DIL is similar to the rationale of the adaptive boosting (AdaBoost) algorithm, thus, it naturally inherits performance improvement attribute of AdaBoost.

One major characteristic of DIL is that, each new base learner added to the ensemble is trained on a set of samples selected based on a distribution got by normalizing the weights of the samples, which ensures that samples with larger errors have a higher probability to be selected as training samples. In general, the samples with high

prediction errors are unknown samples, or samples that have not been used to train learners.

DIL generates a collection of weak learners and combines the predictive values obtained by individual learners using the method of weighted majority voting. This scheme is similar to the AdaBoost algorithm. During each iteration, DIL uses an update strategy to change the weight of the current sample, selecting different training data to obtain diverse weak learners. AdaBoost’s distribution update rule is designed to improve the accuracy of the classifier, while DIL’s distribution update strategy is optimized for learning new data incrementally and further decreasing predictive errors. For a detailed description about AdaBoost, please refer to [44].

In Algorithm 2, the specific pseudo code of the DIL algorithm is presented, and its block diagram is given in Fig. 1.

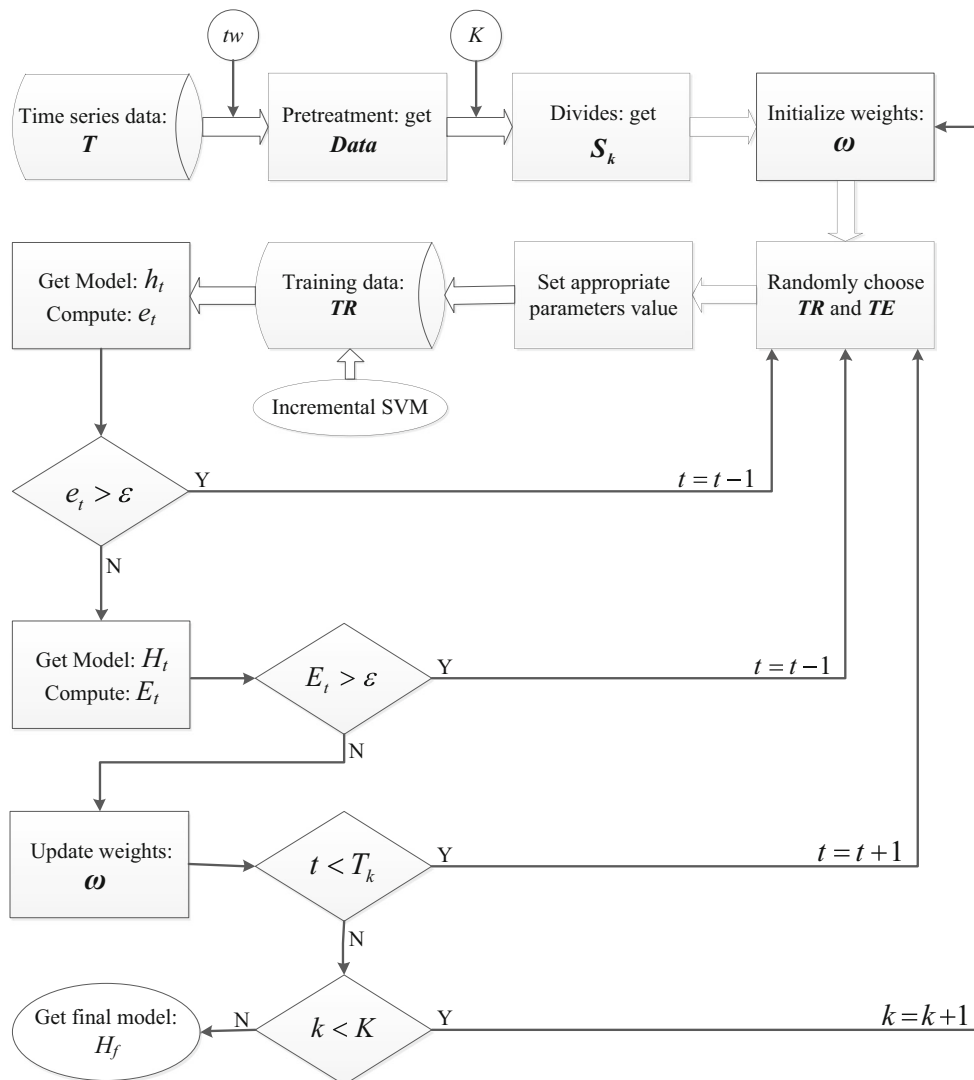


Fig. 1 Block diagram of the DIL algorithm

Algorithm 2 The Double Incremental Learning (DIL) Algorithm

Input: Time series dataset $T = \{x_1, x_2, \dots, x_n\}$.

Time window size tw .

Number of data subset K .

Number of iteration T_k .

Base learner *incremental SVM*.

Output: Final model H_f .

Pretreatment: Time series dataset T is preprocessed to obtain the dataset

$Data = \{(x_i, y_i), i = 1, 2, \dots, N\}$. Divide $Data$ into K parts: $S_k, k = 1, \dots, K$.

For k from 1 to K

Initialize $\omega(i) = 1/m, i = 1, \dots, m$, where m is the size of each subset.

For t from 1 to T_k

(1): Randomly choose training subset TR and testing subset TE according to ω .

(2): On the TR , set the appropriate parameters values and call the base learner.

(3): Get a base model h_t , and compute the error of h_t : e_t on the

$S_k, k = 1, \dots, K$. If $e_t > \varepsilon$, set $t = t - 1$, abandon h_t and go to step 1, where ε is set according to dataset. Otherwise calculate normalized error $\beta_t = e_t / (1 - e_t)$.

(4): Call the weighted majority vote method, get back a composite model:

$H_t = \arg \max_y \sum_{t: |h_t(x) - y| < \delta} \log(1/\beta_t)$, and calculate the composite error: E_t . If

$E_t > \varepsilon$, set $t = t - 1$, abandon H_t and go to step 1.

(5): Set $\Gamma_t = E_t / (1 - E_t)$, update weights: $\omega(i) = \begin{cases} \omega(i) \times \Gamma_t, & \text{if } |H_t(x_i) - y_i| < \delta \\ \omega(i), & \text{otherwise} \end{cases}$,

and set $\omega = \omega / \sum_{i=1}^m \omega(i)$.

End For

End For

Call weighted majority, obtain the final model: $H_f = \arg \max_y \sum_{k=1}^K \sum_{t: |h_t(x) - y| < \delta} \log(1/\Gamma_t)$.

We now give a detailed description of the DIL algorithm. The inputs of DIL are as follows:

1. The original time series dataset $\mathbf{T} = \{x_1, x_2, \dots, x_n\}$. $x_i, i = 1, \dots, n$, is the value at a certain time point t_i , which is a continuous value.
2. Time window size tw , which is required in preprocessing the original time series dataset.
3. The number of data subsets K , which is used to divide the dataset. The dataset obtained by the pretreatment is divided into K parts, to obtain K data subsets.
4. Number of iterations T_k , which means the number of iterations implemented on each data subset, indicating, meanwhile, the number of base learners generated on each data subset.
5. Base learner, i.e., incremental SVM, which is implemented in each iteration.

The final model H_f is the output of the proposed algorithm. Our purpose is to get a final model H_f that possesses powerful predictive capability. In this work, we focus on the research of one-step-ahead prediction, thus H_f is used to predict the data values at the next point in time.

Let $\mathbf{T} = \{x_1, x_2, \dots, x_n\}$ be a time series and $\mathbf{x}_i = (x_i, x_{i+1}, \dots, x_{i+tw-1})$ be the input of the i -th base learner, where tw denotes the time window size. $y_i = x_{i+t}$ is regarded as the target value. Then, (\mathbf{x}_i, y_i) represents a sample, and $\mathbf{Data} = \{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$ is our dataset.

DIL generates an ensemble consisting of weak learners, each base learner is trained on different subsets of the currently available data subset $\mathbf{S}_k, k = 1, \dots, K$. All of the K data subsets are gained by dividing \mathbf{Data} into K parts. In each iteration, $4/45.5$ of the dataset $\mathbf{S}_k, k = 1, \dots, K$ are utilized as the training data, and the remainder are used as the testing data. Each specific instance used to train the base learners is selected according to the weight of each instance in $\mathbf{S}_k, k = 1, \dots, K$. In each iteration, after updating, the weight vector ω is normalized, which makes the weights a distribution. The instances with higher prediction errors are more likely to be added into the training set for the next iteration. For each dataset $\mathbf{S}_k, k = 1, \dots, K$, the weight vector ω can be initialized as any value, while in this paper, each element value of the weight vector ω is initialized to $1/m$, so that, initially, each instance has the same chance of being selected into the training subset.

In the t th iteration, $t = 1, 2, \dots, T_k$, the DIL algorithm first selects the training subset \mathbf{TR} and the testing subset \mathbf{TE} from $\mathbf{S}_k, k = 1, \dots, K$ according to the weight vector ω (Step 1). Then, appropriate parameters are selected for incremental SVM to generate base model h_t (Step 2). The error e_t of model h_t on $\mathbf{S}_k = \mathbf{TR} + \mathbf{TE}, k = 1, \dots, K$ is defined as:

$$e_t = \sum_{i=1}^m \omega(i) \times |h_t(\mathbf{x}_i) - y_i| \quad (16)$$

where $|\cdot|$ indicates the absolute value (Step 3). That is simply the weighted sum of absolute deviations.

If $e_t > \varepsilon$, h_t will be discarded, and \mathbf{TR} and \mathbf{TE} will be rechosen, where ε is a threshold preset according to the distribution of the dataset. That is, whether a base model could be retained is mainly dependent on its performance over $\mathbf{S}_k, k = 1, \dots, K$. The threshold ε is used to measure whether h_t has reached the required level of performance. Since the value of ε is determined based upon the dataset, it usually owns different values for different datasets, but in general ε is less than $1/2$.

If $e_t \leq \varepsilon$ is satisfied, then calculate the normalized error $\beta_t (0 \leq \beta_t \leq 1)$ according to Eq. (17):

$$\beta_t = e_t / (1 - e_t). \quad (17)$$

The rule of weighted majority voting is then used to combine the base models generated in the previous t iterations (Step 4). When voting, the weight of each base model is the logarithm of the reciprocal of the normalized error β_t . Thus, base model with a smaller error rate is assigned a higher voting weight. The composite model H_t is obtained by combining every base model as follows:

$$H_t = \arg \max_y \sum_{t: |h_t(\mathbf{x}) - y| < \delta} \log(1/\beta_t). \quad (18)$$

In order to make it easier for the reader to understand the weighted majority voting rule in DIL, we give a schematic diagram in Fig. 2.

Note that the predicted value given by H_t is the value obtained by weighted majority voting within a certain error range. That is, if the total number of votes received in the interval $(y - \delta, y + \delta)$ is the highest, then the combined forecasting is y .

The composite error E_t of model H_t is computed on $\mathbf{S}_k, k = 1, \dots, K$ as:

$$E_t = \sum_{i=1}^m \omega(i) \times |H_t(\mathbf{x}_i) - y_i|, \quad (19)$$

The composite error E_t and the error e_t have the same mathematical significance. If $E_t > \varepsilon$, then discard the current composite model H_t , select a new training subset and generate a new H_t . It is found that, in most cases, the condition $E_t \leq \varepsilon$ could be satisfied, because the performance of each base model h_t has been verified in step 3. If $E_t \leq \varepsilon$ is satisfied, the composite normalized error Γ_t will be calculated as

$$\Gamma_t = E_t / (1 - E_t). \quad (20)$$

The weight vector ω is updated and normalized so that the weights become a distribution. And then, they are used

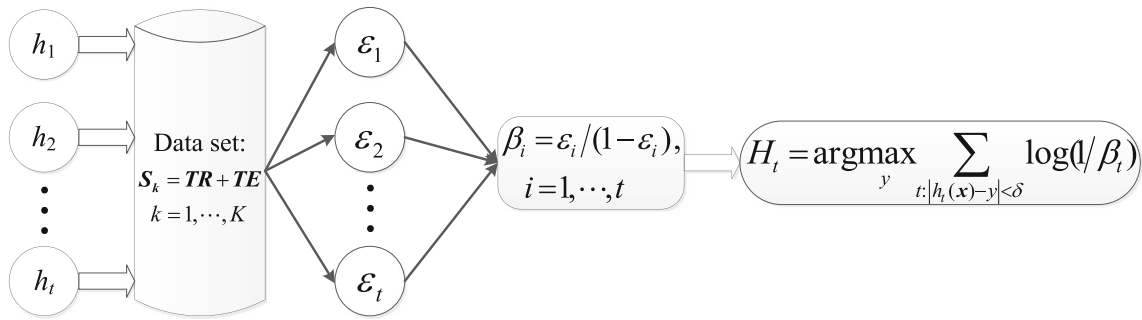


Fig. 2 The weighted majority voting rule in DIL

to select the training and testing subsets, i.e., TR and TE , respectively, for the next iteration. The specific weights update method is as follows:

$$\omega(i) = \begin{cases} \omega(i) \times \Gamma_r, & \text{if } |H_t(\mathbf{x}_i) - y_i| < \delta \\ \omega(i), & \text{otherwise} \end{cases} \quad (21)$$

Furthermore, the weight vector ω is normalized as:

$$\omega = \omega / \sum_{i=1}^m \omega(i). \quad (22)$$

The weights update rule is one of the most important parts of the DIL algorithm. In order to make it easier to understand, its schematic diagram is given in Fig. 3.

Following this rule, if the prediction error of the composite model H_t for y_i is within a certain range, i.e., $|H_t(\mathbf{x}_i) - y_i| < \delta$, the corresponding weight $\omega(i)$ is multiplied by a factor Γ_r . According to the definition of Γ_r , its value is less than 1. If $|H_t(\mathbf{x}_i) - y_i| < \delta$ is not satisfied, the corresponding weight $\omega(i)$ will remain unchanged. According to this rule, instances with higher prediction errors are more likely to be selected into TR in the next iteration. If we regard those instances, whose prediction errors are large, as hard instances, while the instances with small prediction errors as simple instances, then the algorithm would be more and more concerned about hard instances, and the hard instances will be further intensively studied. Therefore, the DIL algorithm belongs to the incremental learning paradigm, which is specially designed for TSP.

After generating T_k base models on each dataset S_k , $k = 1, \dots, K$, all the base models generated so far are integrated by using the weighted majority voting rule to obtain the final composite model H_f . The specific form of the final composite model H_f is as follows:

$$H_f = \arg \max_y \sum_{k=1}^K \sum_{t: |h_t(\mathbf{x}) - y| < \delta} \log(1/\Gamma_t) \quad (23)$$

The time complexity of the proposed DIL algorithm is $O(KT_k(C_H + C_h))$, where K represents the number of data

subsets after dataset division, and T_k represents the number of iterations. C_h and C_H , respectively, represent the number of base models and compound models discarded during one iteration.

Note that the DIL algorithm preserves all of the generated base models; therefore, the previous data can be discarded to save storage space, without forgetting the previous knowledge. In addition, DIL has another important feature, i.e., the independence from the base learner. That is to say, any appropriate weak learner could be chosen as the base model for DIL, which has only a little effect on the overall performance of the algorithm. However, choosing the corresponding base learner according to the specific problems is helpful for DIL to achieve more desirable prediction results. Generally, the DIL algorithm is able to achieve good prediction effect for various TSP problems, which is propitious to solve many problems in reality.

3.3 A discussion about the extensions of DIL algorithm with respect to deep learning

In recent years, with the development of machine learning theory, deep learning (DL), as a new research branch, has emerged. A lot of researches and applications have verified the powerful performance of deep learning [45–47]. Deep learning is essentially a nonlinear combination of multi-level representation learning methods. Representation learning [48] refers to learning the feature representation from data, in order to extract useful information from the data for the purpose of classifying or forecasting. Starting from raw data, DL paradigm transforms each layer’s features into higher layers and more abstract features, so as to discover intricate structures in high-dimensional data.

In the field of DL, various deep structures have been put forward. Among them, several most classic deep structures are deep belief network (DBN) [49], deep Boltzmann machine (DBM) [50] and stacked autoencoder (SAE) [51]. DBN and DBM are obtained by stacking restricted Boltzmann machines [52]. SAE is formulated by stacking

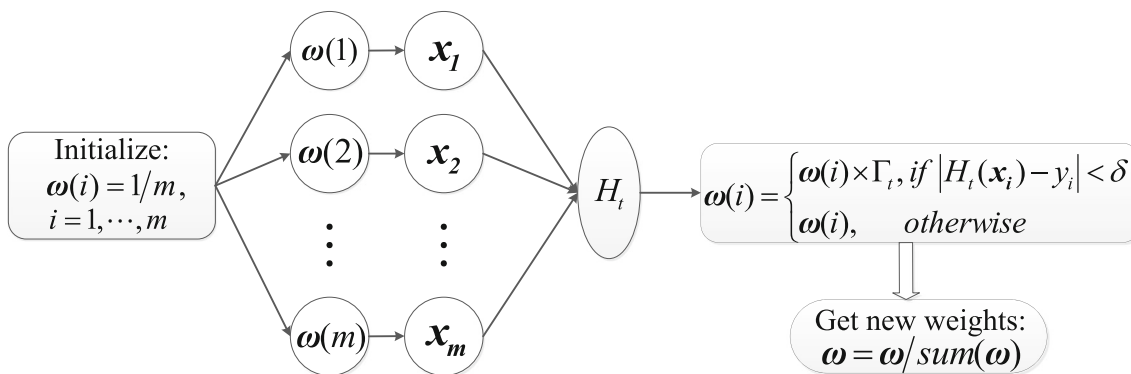


Fig. 3 The weights update rule in DIL

autoencoders [53]. Furthermore, researchers have proposed some excellent deep structures, recently. For example, Zhang et al. [54] developed a character-level sequence-to-sequence learning method, i.e., RNNembed, for neural machine translation.

About the extensions of the proposed DIL algorithm with respect to deep learning, we have three ideas. The first one is that, inspired by SAE, multiple incremental SVMs could be stacked together to get a deep incremental support vector machine, which could be used to replace the original base learner of the DIL algorithm to further improve its performance.

The second idea is, firstly, building a deep neural network for feature extraction from data, and then, feeding the obtained feature representation through unsupervised learning into the base models of the DIL algorithm, i.e., incremental SVMs, so that the DIL algorithm could be used to predict the trend of data.

The third thought is, it might be desirable to integrate incremental learning with deep learning paradigm, such that deep neural networks can learn data incrementally. For example, a deep neural network is used for feature learning, while an incremental learning algorithm integrates existing feature sets with newly acquired features in a particular way.

4 Numerical experiments

In order to evaluate the performance of the proposed DIL algorithm, simulation experiments based on several benchmark synthetic and real-world datasets are conducted. And the experimental results of DIL on each dataset are compared with those state-of-the-art algorithms proposed in other literatures, with the detailed experimental results and discussions given in Sect. 4.2. The experimental results have demonstrated the significant improvement to the predictive performance achieved by the proposed algorithm.

4.1 Datasets and experimental setup

4.1.1 Datasets

Simulation experiments on six benchmark datasets have been conducted in this work, including two synthetic datasets and four real-world datasets. The details of the six benchmark datasets are described in turn as below.

(A) Mackey–Glass database

Originally, the Mackey–Glass equation was presented as a model for regulating blood cell. One of the major features of the Mackey–Glass dataset is its chaotic nature; therefore, it is one of the classical datasets in the field of chaotic TSP. The time series is generated by the following nonlinear differential equation:

$$\frac{dx}{dt} = \frac{ax(t - \tau)}{1 + x^c(t - \tau)} - bx(t). \tag{24}$$

If $\tau > 16.8$, then the time series is chaotic. According to the literatures [7, 55, 56], the parameters selected for generating the time series are $a = 0.2$, $b = 0.1$, $c = 10$, and $\tau = 17$. According to Eq. (24), a chaotic time series dataset with the length of 10,000 is generated, with the initial value being set to 1.2, that is, $x(0) = 1.2$. The first 8000 values are discarded and the last 2000 values are kept for the experiments.

(B) Lorenz database

The Lorenz time series is a three-dimensional dynamical system that exhibits chaotic flow and was found by Edward Lorenz. The equation for generating the Lorenz time series is as follows:

$$\begin{aligned} \frac{dx(t)}{dt} &= \sigma[y(t) - x(t)] \\ \frac{dy(t)}{dt} &= x(t)[r - z(t)] - y(t) \\ \frac{dz(t)}{dt} &= x(t)y(t) - bz(t). \end{aligned} \tag{25}$$

where σ , r and b are the dimensionless parameters. The parameters used to generate the time series are set according to the literatures [7, 55, 56], where $\sigma = 10$, $r = 28$ and $b = 8/3$. In this group of experiments, the x -coordinate of the Lorenz time series is taken as the experimental dataset. A chaotic time series dataset with the length of 10,000 can be generated according to Eq. (25). Similarly, in order to reduce the transient effect, we discard the first 8000 values and keep the last 2000 values for the experiments.

(C) Sunspot database

The Sunspot time series is a time series that regularly records the number of sunspots, which is an important indicator for the study of the solar cycle. The solar cycle has a significant impact on the Earth’s climate, the operation of the satellite, and so on; therefore, it is of great practical significance to predict the sunspots number. However, the prediction of sunspot numbers is still a challenging task, because of its own complexity. The monthly smoothed Sunspot time series used in this paper is obtained from Sunspot Index World Data Center (SIDC) [57]. To compare the performance of the proposed DIL algorithm with the other algorithms in the literatures, the Sunspot time series from November 1834 to June 2001 is selected as our dataset, which contains 2000 data values.

(D) Three financial datasets

In addition to the three benchmark datasets introduced above, there are three important stock index datasets, respectively, N225, DJI and SSE. These three stock indexes have a more important impact on the international financial markets, thus, it is necessary to do some research on them. The N225 dataset in the fourth group of experiments consists of monthly sampled data, which includes all the closing prices from April 1988 to March 2015, containing 324 data points [58]. The monthly closing prices of DJI from February 1985 to March 2015 are selected as the experimental data for the fifth group of experiments, including 352 data values [58]. Similarly, in the sixth group of experiments, the monthly closing prices of SSE from December 1990 to January 2015 are selected as the experimental data, containing 290 data points [58].

Furthermore, in order to facilitate comparison, it is necessary to normalize the data, that is, to adjust the data to the range of [0, 1]. The normalization formula is as follows:

$$x'_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \tag{26}$$

where x'_i is the normalized data value, x_i is the original value, x_{\max} and x_{\min} are the maximum and minimum values in the original data, respectively.

4.1.2 Experimental setup

In order to carry out the experiments of this work, some parameters are required to be preset appropriately. The specific parameters of the DIL algorithm are shown in Table 1.

For the time window size tw , it is found by trial-and-error that, the feasible value range of the time window tw is [4, 10] for datasets with different sizes. For data partition, it is important to determine an appropriate size for the data subset $S_k, k = 1, \dots, K$, which is usually related to the size of the original dataset. After repeated experiments, it is found that, for datasets with different sizes, when the number of subsets K is set within the range [2, 6], the subset $S_k, k = 1, \dots, K$ will have a more appropriate size. In this paper, the original dataset is evenly divided into several subsets, that is, the sizes of S_k ’s are identical. However, in fact, the DIL algorithm does not force the sizes of the subsets S_k ’s to be the same. The DIL algorithm would generate T_k base models on each $S_k, k = 1, \dots, K$. T_k should not be too large, because blindly increasing T_k is not much helpful to upgrade the model performance, and would lead to inefficiency; however, T_k should also not be too small, because a too small T_k will reduce the model performance. As shown in Table 1, the value of T_k in this work is set as 20.

In order to evaluate the performance of the proposed DIL algorithm and to compare it with the algorithms in the literatures, we need proper measurements to measure their prediction performance. In the field of TSP, there are several commonly used prediction performance measurements, among which the following four error measures, i.e., root mean squared error (RMSE), normalized mean squared error (NMSE), mean absolute error (MAE), and absolute error (Error), are employed in this work. Their specific formulas are presented as follows:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2}, \tag{27}$$

$$NMSE = \left(\frac{\sum_{i=1}^N (x_i - \hat{x}_i)^2}{\sum_{i=1}^N (x_i - \bar{x})^2} \right), \tag{28}$$

Table 1 The parameters of DIL

Parameters	Time window size tw	Number of data subsets K	Number of iterations T_k
Values	[4, 10]	[2, 6]	20

$$\text{MAE} = \frac{\sum_{i=1}^N |x_i - \hat{x}_i|}{N}, \quad (29)$$

$$\text{Error} = x_i - \hat{x}_i, \quad (30)$$

where x_i and \hat{x}_i , $i = 1, \dots, N$ represent the original data and the predicted data, respectively. \bar{x} represents the average of the original data, and N denotes the size of the testing dataset.

In order to reduce the contingency of the experimental results, twenty repetitive experiments are performed on each dataset, and the average values of the repetitive experimental results are taken as the final results.

Finally, the operating environment of the experiments in this work is MATLAB R2009a. The hardware configuration is one PC with 2.4 GHz CPU and 12 GB RAM.

4.2 Results and discussion

In this section, the experimental results of the proposed algorithm on the six benchmark datasets are given, and the relevant experimental results graphics are drawn. Furthermore, the prediction performance of the DIL algorithm is compared with other excellent algorithms in the literatures.

Tables 2 and 3 show the maximum, minimum and average values of RMSE and the run times (CPU time) of the DIL algorithm in 20 independent and repetitive runs on the six benchmark datasets, which are what they have in common on each dataset. The difference between Tables 2 and 3 lies in that, Table 2 lists out the maximum, minimum and average values of NMSE on the Mackey–Glass, Lorenz and Sunspot datasets, while Table 3 displays those values of MAE on the N225, DJI and SSE datasets. The reason why the experimental results are listed out in this way is because the performance of DIL is required to be compared with other state-of-the-art algorithms. However,

the comparative algorithms in some literatures provide their RMSE and NMSE results, while the comparative algorithms in other literatures provide their RMSE and MAE results.

As can be seen from Tables 2 and 3, the DIL algorithm is capable to complete training and prediction on the six datasets in a short period of time. In addition, according to the minimum, maximum, mean of the RMSEs, NMSEs and MAEs, we can see that the error range of DIL on five datasets is very compact with slight fluctuation, except those on the Mackey–Glass dataset. Although the error fluctuation on the Mackey–Glass dataset is slightly larger, it is also within acceptable limits. And it can be seen from the subsequent chapters that, it is caused by very few examples. From these experimental results, a conclusion could be drawn that, the DIL algorithm possesses a relatively stable performance in most cases.

As can be seen from Figs. 4, 5, 6, 7, 8 and 9 that, the DIL algorithm has good predictive performance on all the six benchmark datasets. Figures 4a, 5a, and 6a compare the predictive values with the original values on the Mackey–Glass, Lorenz, and Sunspot datasets, respectively, which indicate that the differences between these two values are very small. Figures 7a, 8a, and 9a compare the predictive values with the original values on the N225, DJI, and SSE datasets, respectively. On these three datasets, the high degree of unpredictability of financial data causes slightly larger prediction errors. But, the forecast trend is basically consistent with the actual trend, and the prediction performance of the DIL algorithm on these three datasets also exceeds many existing algorithms.

Correspondingly, Figs. 4b, 5b, 6b, 7b, 8b, and 9b show the prediction errors (absolute errors) on the six datasets, respectively. The results show that, the absolute errors on the six datasets at most reach the value of 10^{-2} or reduce to even

Table 2 RMSE, NMSE and CPU time of the DIL algorithm on the first three datasets

Datasets	RMSE			NMSE			Time (s)
	Max	Min	Mean	Max	Min	Mean	
Mackey	9.66E−04	3.65E−04	5.69E−04	1.41E−05	2.01E−06	5.18E−06	913.51
Lorenz	3.79E−04	3.15E−04	3.46E−04	3.11E−06	2.15E−06	2.60E−06	199.77
Sunspot	7.26E−03	6.41E−03	6.72E−03	6.24E−04	4.07E−04	4.86E−04	67.58

The way of expression of 9.66E−04 is scientific notation, that is, 9.66E−04 stands for 9.66×10^{-4} . The same markings in the following tables have the same meaning

Table 3 RMSE, MAE and CPU time of the DIL algorithm on the last three datasets

Datasets	RMSE			MAE			Time (s)
	Max	Min	Mean	Max	Min	Mean	
N225	3.00E−02	2.30E−02	2.54E−02	2.27E−02	1.75E−02	1.96E−02	2.95
DJI	2.03E−02	1.77E−02	1.91E−02	1.48E−02	1.35E−02	1.42E−02	4.41
SSE	2.54E−02	2.04E−02	2.23E−02	2.03E−02	1.55E−02	1.73E−02	2.75

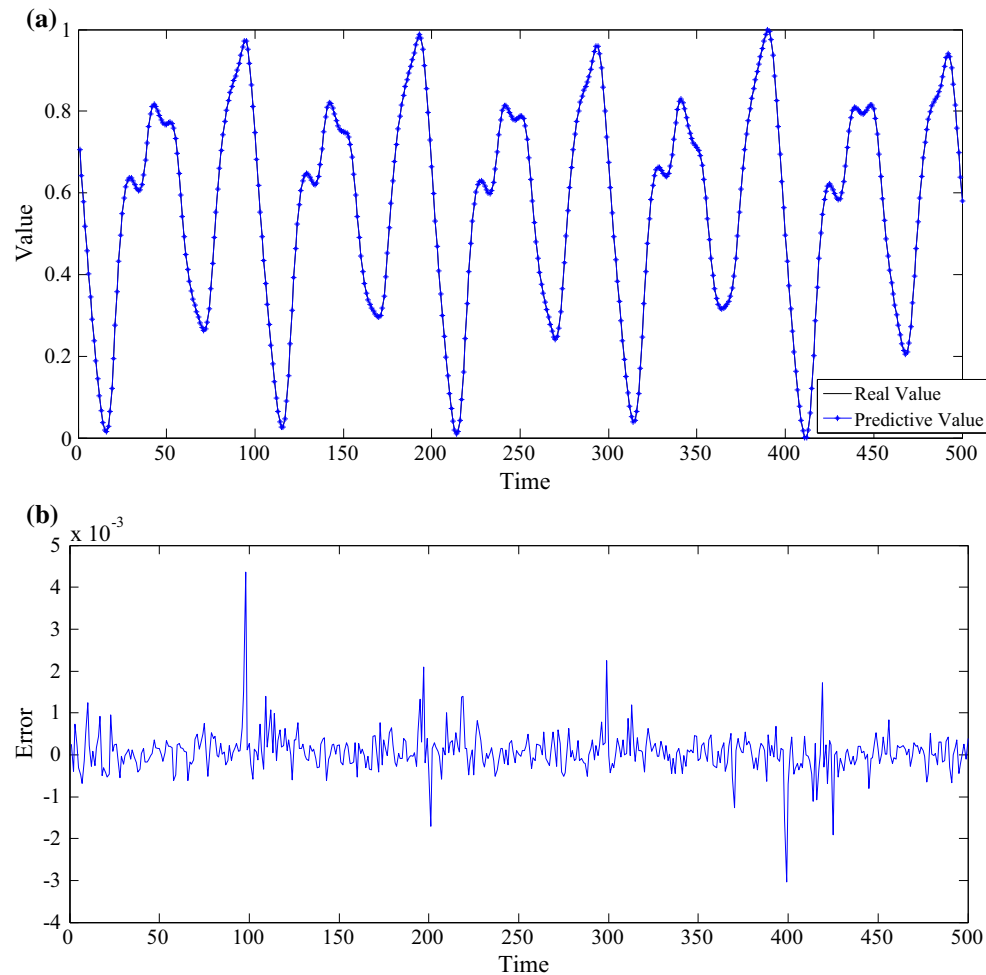


Fig. 4 **a** The prediction results graph of the DIL algorithm on Mackey–Glass. **b** The prediction errors of the DIL algorithm on Mackey–Glass

lower values. Figure 4b shows that, very few data points fluctuate much, which confirms the results in Table 2. From these prediction errors graphs, we can conclude that, the DIL algorithm has less error fluctuation on the five datasets, and its prediction performance is stable.

As can be seen from the preceding introduction, the DIL algorithm integrates the incremental learning paradigm together with incremental SVM. In order to analyze which part has a greater impact on the overall performance of the DIL algorithm, we split the DIL algorithm into two parts. The first part is an incremental learning framework, in which, the traditional SVM is chosen as base learner, replacing the incremental SVM. This algorithm is called single incremental learning (SIL) algorithm in this paper. The second part is the base learner of the DIL algorithm, i.e., incremental SVM, which is abbreviated as ISVM.

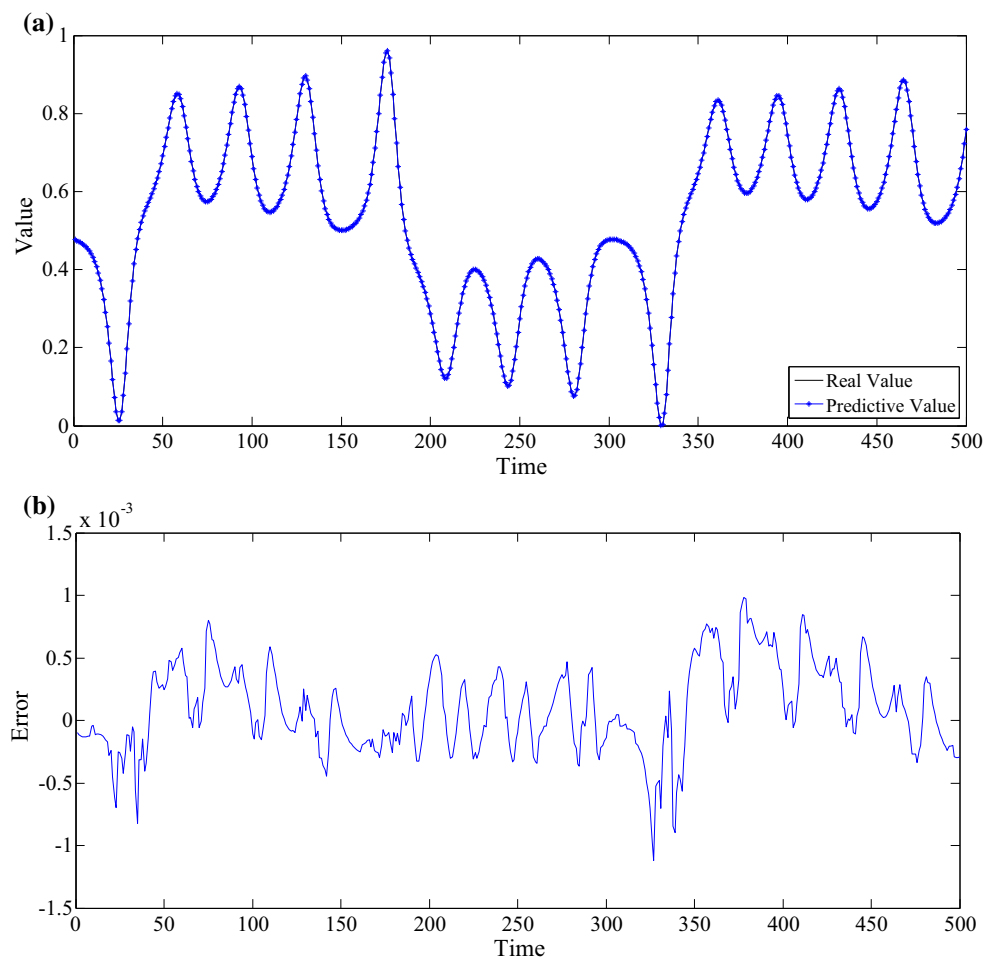
Tables 4 and 5 show the experimental results of DIL, SIL and ISVM algorithms on the six datasets, respectively. These results are obtained by averaging the results of twenty repeated experiments. As can be seen from these two tables, the performances of SIL and DIL are relatively

close to each other, while the performance of ISVM is quite different from the former two. This situation illustrates that the incremental learning framework contributes more to the overall performance of the proposed DIL algorithm, while the base learner has little impact on it.

There are some parameters in the proposed DIL algorithm, among which, the time window size tw , the number of data subsets K , and the number of iterations T_k are the three most important parameters. In order to analyze the impact of these parameters on the performance of the DIL algorithm, repeated experiments are performed on six data sets. In the course of the experiments, each parameter takes various values to carry on the experiments in turn. Through these experiments, we can determine a suitable range of each parameter values. These experiments also further verify the values intervals of the parameters setting in Sect. 4.1.2 are reasonable.

Tables 6, 7 and 8 show the results of DIL algorithm with different values of time window size tw on six datasets, while the values of the other two parameters K and T_k are set as 4 and 20, respectively, by trial-and-error. It is not

Fig. 5 **a** The prediction results graph of the DIL algorithm on Lorenz. **b** The prediction errors of the DIL algorithm on Lorenz



difficult to find from these three tables that, the prediction results of DIL algorithm are similar when the values of tw are within the interval [4, 10]. If the tw 's value is not within the range of [4, 10], the prediction effect of DIL algorithm will be worse. This shows that, the value of tw should not be too large nor too small. If tw is too small, samples will contain too little valid information, which might lead to poor predictive performance. If the value of tw is too large, samples will contain too much noise, thus affecting the learner's learning.

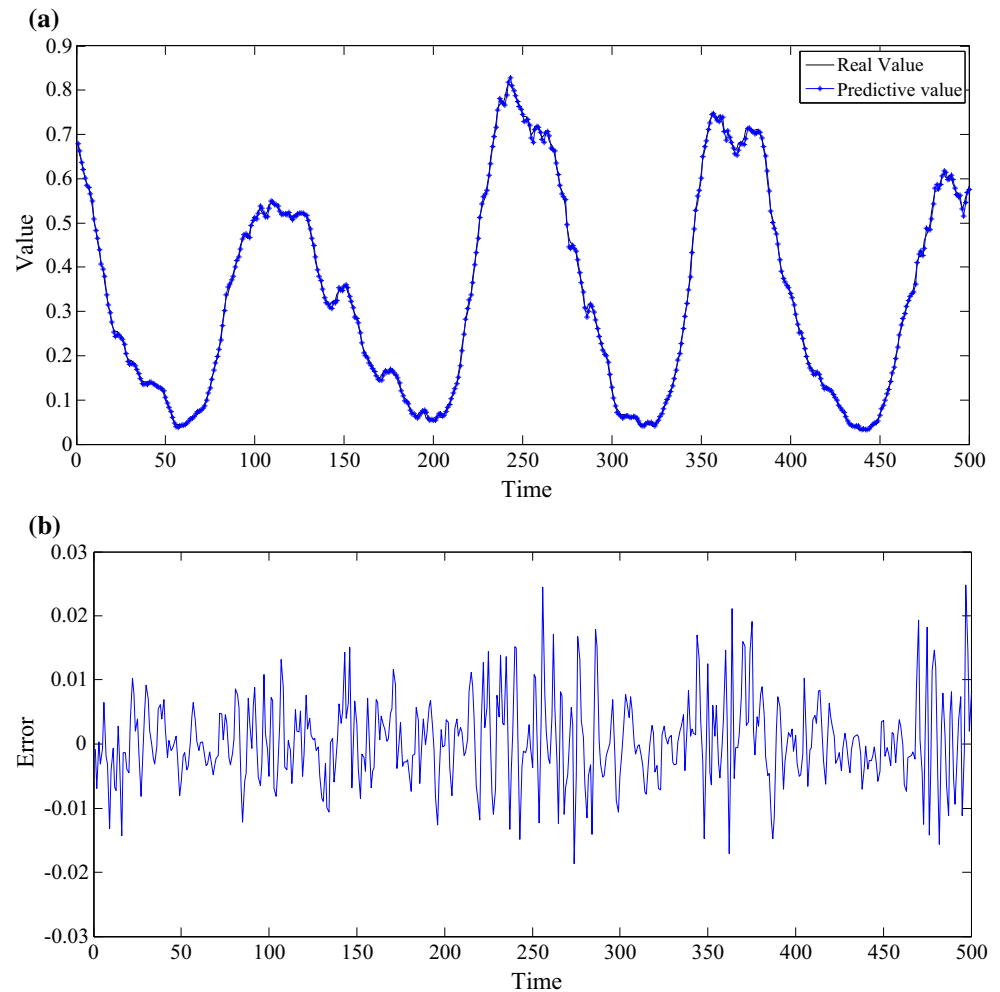
The experimental results of the DIL algorithm with different values of the number of data subsets K on six datasets are shown in Tables 9, 10 and 11, while the other two parameters tw and T_k take the values of 7 and 20, respectively, by trial-and-error. From these three tables, we can draw a conclusion similar to the previous one. The three tables show that, proper partitioning of the datasets can improve the overall performance of the algorithm. However, the size of the data subset should not be too small, since a single base model learned on an undersized dataset would have a poor overall description of the data.

This also has a certain impact on the overall performance of DIL algorithm.

Similarly, different values are set for the number of iterations T_k , and then experiments are conducted on the six datasets. In this experimental part, the values of the other two parameters tw and K are set as 7 and 4, respectively, by trial-and-error. The experimental results are presented in Tables 12, 13 and 14. It can be easily seen from the experimental results that, when the value of T_k is too small, that is, the generated base models are too little, the performance of the algorithm will be obviously decreased. This is because the number of base models is too small and the data have not been fully learned. It is not difficult to find that, when the value of T_k increases to a certain extent, if it continues to increase, the performance of the algorithm would not only not be significantly improved, but rather result in performance degradation. The reason might be that, when the value of T_k becomes too large, the diversity among the base models could decrease, while their accuracies might not be improved, simultaneously.

In order to show the superiority of the proposed algorithm, the predictive performance of DIL on the six time

Fig. 6 **a** The prediction results graph of the DIL algorithm on Sunspot. **b** The prediction errors of the DIL algorithm on Sunspot



series datasets are compared with the latest excellent algorithms in the literatures, as shown in Tables 15, 16, 17, 18, 19 and 20. For the sake of comparison, the best results are highlighted in bold. In addition, the results listed in the tables are the average values of the 20 repetitive experimental results obtained by DIL.

Table 15 shows the comparison between the algorithms reported in the literatures with the proposed DIL algorithm based upon RMSE and NMSE values on the Mackey–Glass dataset. It is easy to see from the table that, DIL has better performance in predicting this time series, compared with other algorithms. Obviously, compared with other algorithms, both RMSE and NMSE values have been significantly reduced by DIL.

Table 16 presents a comparison of the prediction errors on the Lorenz time series. The results show that, the DIL algorithm has much smaller RMSE and NMSE values, and more accurate prediction results.

In Table 17, the proposed DIL algorithm achieves smaller RMSE in predicting the Sunspot time series, when compared to other algorithms. The mean value of NMSE

for 20 runs of DIL is minimal, although only slightly smaller than the second optimal algorithm. Table 2 shows that, the maximum NMSE of DIL is $6.24\text{E}-04$ on the Sunspot dataset, which is worse than the Residual Analysis method using Hybrid Elman-NARX Neural Networks (HENNN-RA) [55] and Taguchi’s Design of Experiment (Taguchi’s DoE) [56] models. However, we can see that, the minimum and average NMSE values achieved by DIL are smaller than Taguchi’s DoE. In fact, 16 results of NMSEs are within the range of $(4.07\text{E}-04, 5.03\text{E}-04)$ in 20 repetitive experiments. The prediction NMSEs of the proposed model are better than the second optimal model in most cases. Combining with the results of NMSE and RMSE, we have sufficient reason to believe that the performance of the proposed DIL algorithm is better than all the other comparative models.

In Tables 18, 19 and 20, the main comparisons are conducted toward the results of RMSE and MAE. Table 18 shows that, the average values of RMSE and MAE obtained by DIL are the smallest when predicting the N225 time series, which is superior to the best existing neuron

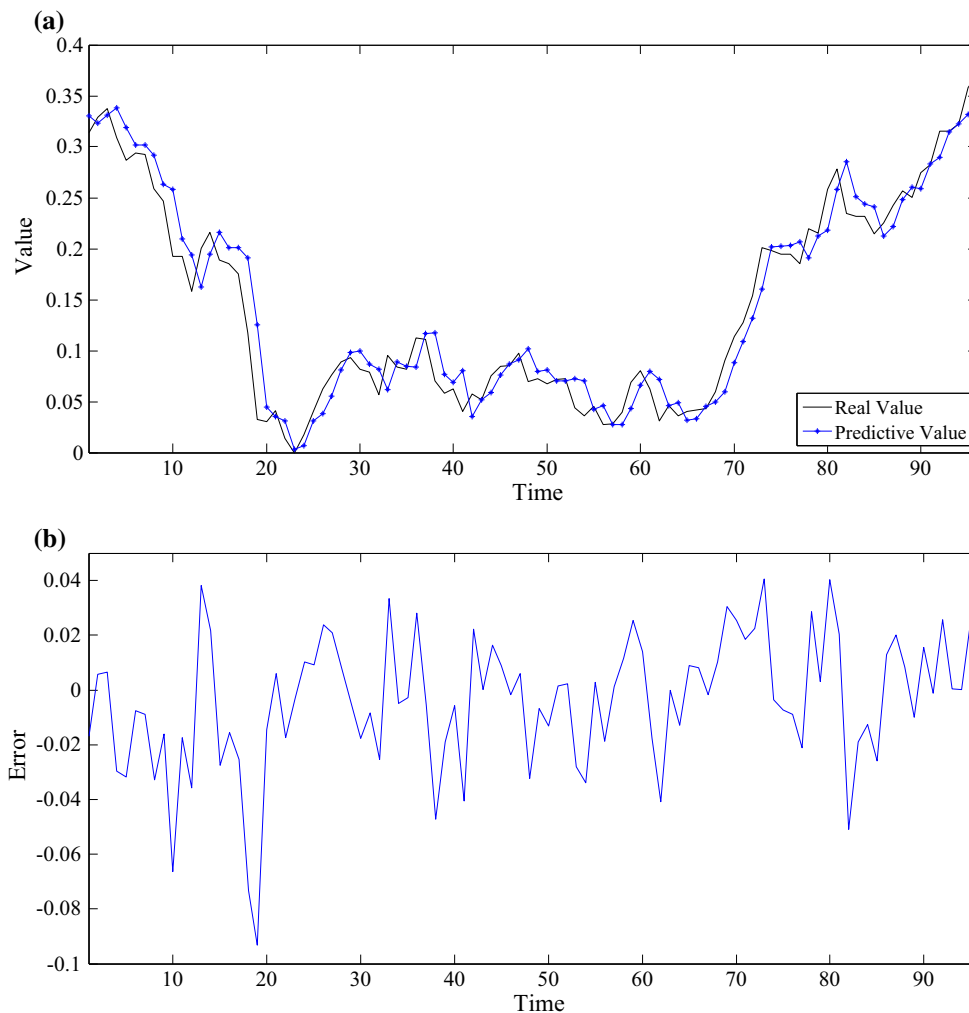


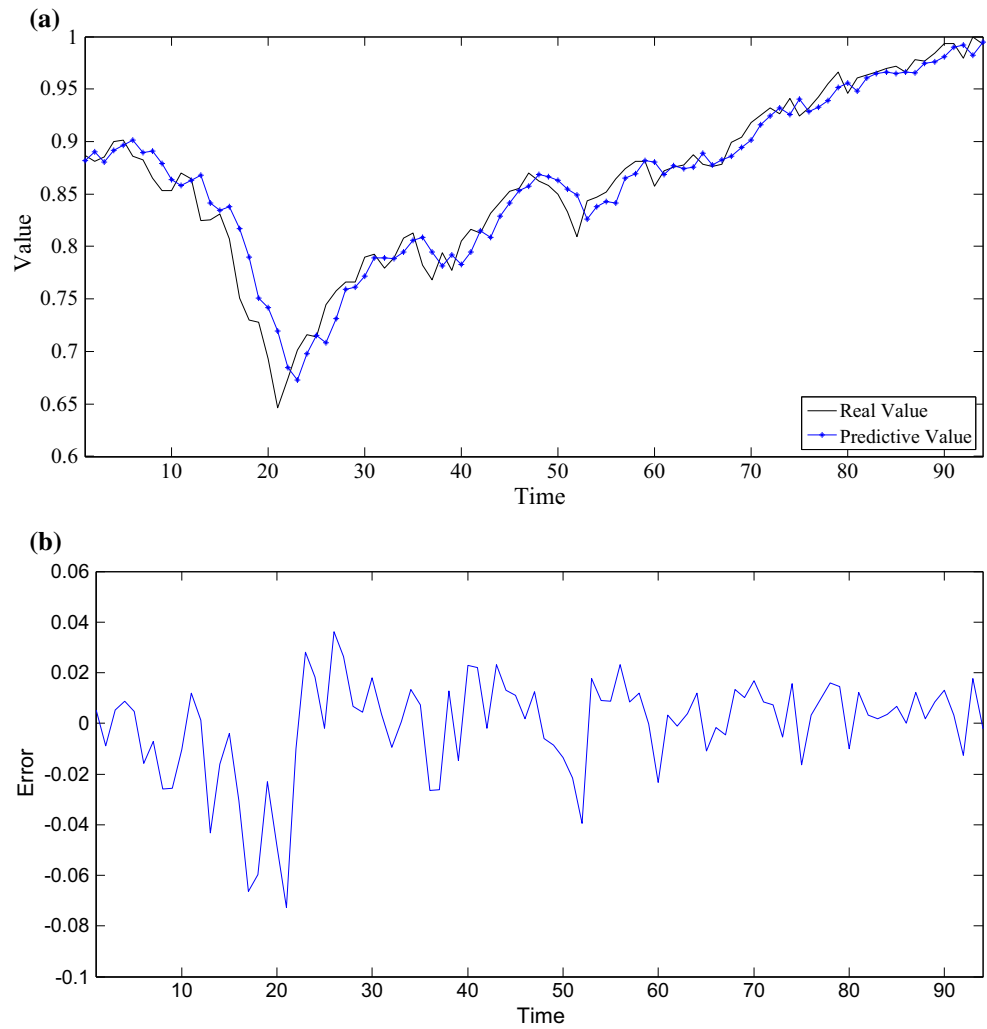
Fig. 7 **a** The prediction results graph of the DIL algorithm on N225. **b** The prediction errors of the DIL algorithm on N225

model with nonlinear interactions between excitation and inhibition of dendrites (NBDM) [58]. Furthermore, we can conclude that, DIL has improved performance by at least 25% compared to NBDM. As can be seen from Table 19, when the DJI time series is predicted, the proposed DIL algorithm achieves the smallest average values of RMSE and MAE, which are $1.91\text{E}-02$ and $1.42\text{E}-02$, respectively. Compared to the optimal NBDM, the average values of RMSE and MAE achieved by DIL are reduced by at least 45%. In Table 20, similarly, DIL achieves the optimal average results of RMSE and MAE when predicting the SSE time series. And the average values of RMSE and MAE obtained by DIL are decreased by 19.5 and 22.4%, respectively, compared with the best NBDM.

Next, we will further verify the computational complexity of the proposed DIL algorithm through experiments. Moreover, five comparative algorithms are selected to be compared with the proposed DIL algorithm, including locally linear neurofuzzy model with locally linear model

tree (LLNF-LoLiMot) algorithm [60], competitive island-based cooperative coevolution methods for two islands (CICC-two-island) [62], cooperative coevolution for training recurrent neural networks with synapse level encoding (SL-CCRNN) [7], orthogonal least squares learning algorithm for Radial Basis Function networks (RBF-OLS) [60], and cooperative coevolution for training recurrent neural networks with neuron level encoding (CCRNN-NL) [61], in terms of overall computational time on the six benchmark datasets. According to the previous analysis, the computational complexity of the proposed DIL algorithm is $O(KT_k(C_h + C_H))$. In the following, the computational complexities of the five selected comparative algorithms are briefly analyzed. $O(2Mp^4)$ is the time complexity of LLNF-LoLiMot, where M is the number of neurons, p is the dimension of the sample space. The computational complexity of SL-CCRNN is $O(nsT)$, where n means n generations, s is equal to the total number of synapses and biases, T is the number of iterations.

Fig. 8 **a** The prediction results graph of the DIL algorithm on DJI. **b** The prediction errors of the DIL algorithm on DJI



$O(T_1T_2D(S + N))$ is the computational complexity of CICC-two-island, where T_1 represents global evolution times, T_2 means island evolution times, D is the depth of n generations, N is the number of sub-populations at neuron level, S is the number of sub-populations at synapse level. The time complexity of CCRNN-NL is $O(ckm)$, where c is the number of cycles, k is the total number of hidden and output neurons, m is the number of generations. $O(Rp)$ is the time complexity of RBF-OLS, where R is the number of all the candidate regressors, p represents the number of steps set in the algorithm.

Table 21 shows the overall computational times of these algorithms on the six benchmark datasets, which are also obtained by averaging the results of twenty repeated experiments. Combining the results in Table 21 and the time complexities of the comparative algorithms, the following conclusions could be drawn easily. SL-CCRNN, CCRNN-NL and DIL have similar time complexities, and their running times are also relatively close. However, the predictive performance of the DIL algorithm is much better

than either of them. The time complexities of LLNF-LoLiMot and CICC-two-island are higher than DIL, and correspondingly, their running times are much longer than DIL. Moreover, the predictive performance of DIL is better than both of them. This situation shows that, DIL is superior to the two comparative algorithms in both efficiency and performance. For RBF-OLS algorithm, its time complexity is lower than DIL, and its running time is also shorter than DIL. However, as you can see from Tables 15, 16, 17, 18, 19 and 20, DIL’s predictive performance is better than RBF-OLS.

5 Conclusions and future works

In this paper, a novel integrated algorithm, i.e., the DIL algorithm, with promising performance for TSP is presented, where incremental SVM is employed as the base learner. In the first stage of DIL, several base models are generated, with each one generated based on a respective

Fig. 9 **a** The prediction results graph of the DIL algorithm on SSE. **b** The prediction errors of the DIL algorithm on SSE

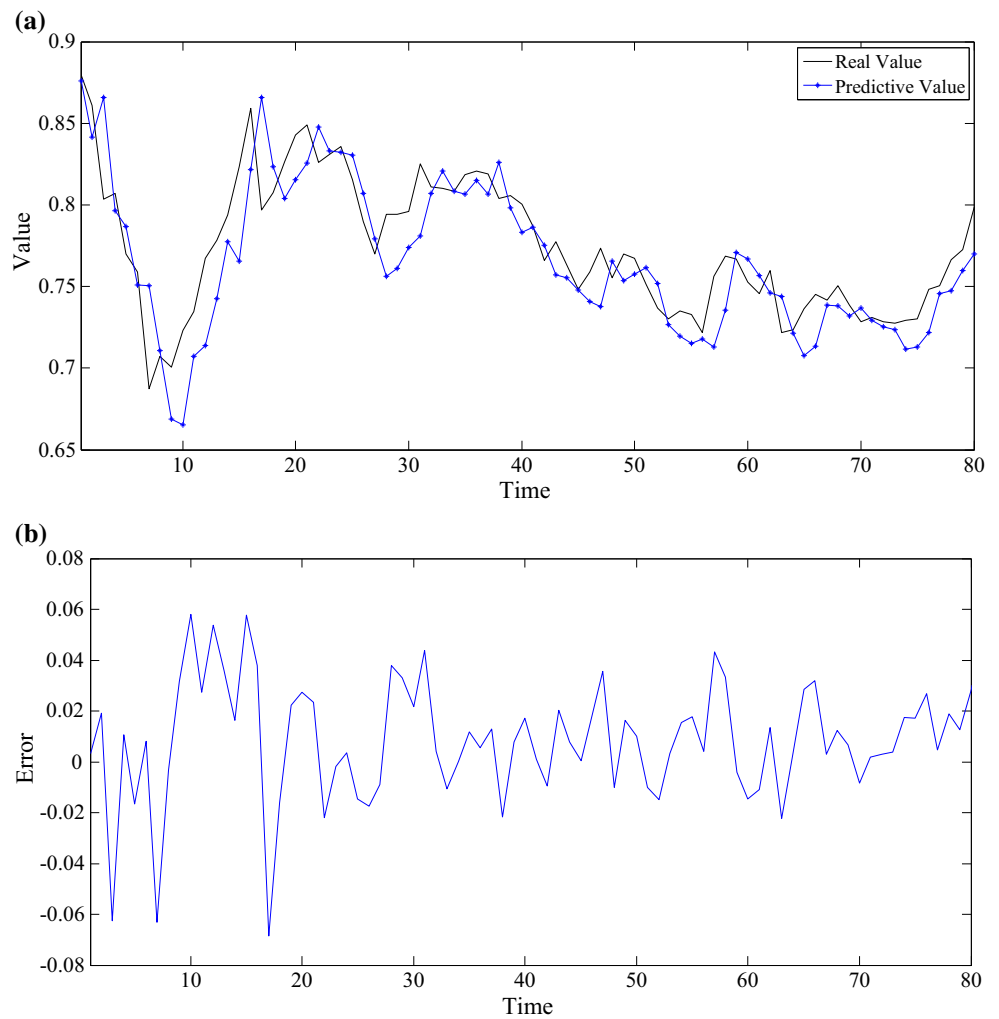


Table 4 The results of DIL, SIL and ISVM algorithm on the first three datasets

Datasets	RMSE			NMSE		
	DIL	SIL	ISVM	DIL	SIL	ISVM
Mackey	5.69E-04	6.19E-04	1.19E-03	5.18E-06	5.66E-06	1.02E-05
Lorenz	3.46E-04	3.87E-04	5.33E-03	2.60E-06	3.11E-06	7.86E-05
Sunspot	6.72E-03	7.01E-03	1.52E-02	4.86E-04	5.25E-04	2.24E-03

Table 5 The results of DIL, SIL and ISVM algorithm on the last three datasets

Datasets	RMSE			MAE		
	DIL	SIL	ISVM	DIL	SIL	ISVM
N225	2.54E-02	3.02E-02	5.63E-02	1.96E-02	2.43E-02	4.43E-02
DJI	1.91E-02	2.35E-02	1.33E-01	1.42E-02	1.91E-02	7.22E-02
SSE	2.23E-02	2.44E-02	6.61E-02	1.73E-02	2.01E-02	4.77E-02

training data subset. Then, during training, the weighted majority voting method is used to combine the base models into a composite model. According to the composite model, the weight of one specific sample is updated by

using a novel weight update rule. Finally, after all the base models have been learnt using all the training subsets, again, the weighted majority voting technique is utilized to combine them to get the final composite model. In the DIL

Table 6 RMSE of the DIL algorithm with different parameter tw on six datasets

Datasets	tw				
	2	4	7	10	15
Mackey	6.81E−03	5.64E−04	5.84E−04	6.12E−04	1.33E−03
Lorenz	8.96E−04	4.19E−04	3.44E−04	5.13E−04	1.60E−03
Sunspot	7.80E−03	6.71E−03	6.70E−03	7.02E−03	8.64E−03
N225	3.34E−02	2.55E−02	2.46E−02	2.63E−02	3.93E−02
DJI	2.83E−02	1.95E−02	1.91E−02	1.99E−02	3.32E−02
SSE	4.17E−02	2.23E−02	2.66E−02	2.79E−02	4.67E−02

Table 7 NMSE of the DIL algorithm with different parameter tw on the first three datasets

Datasets	tw				
	2	4	7	10	15
Mackey	6.87E−05	5.04E−06	5.58E−06	6.09E−06	2.65E−05
Lorenz	1.41E−05	3.44E−06	2.55E−06	4.46E−06	6.38E−05
Sunspot	6.78E−04	4.83E−04	4.78E−04	5.22E−04	8.45E−04

Table 8 MAE of the DIL algorithm with different parameter tw on the last three datasets

Datasets	tw				
	2	4	7	10	15
N225	2.95E−02	2.05E−02	1.92E−02	2.10E−02	3.35E−02
DJI	2.35E−02	1.46E−02	1.40E−02	1.55E−02	3.28E−02
SSE	3.12E−02	1.77E−02	1.92E−02	2.24E−02	4.11E−02

Table 9 RMSE of the DIL algorithm with different parameter K on six datasets

Datasets	K				
	2	4	6	8	10
Mackey	5.89E−04	5.74E−04	6.31E−04	7.81E−04	8.93E−04
Lorenz	3.80E−04	3.39E−04	4.12E−04	5.73E−04	6.91E−04
Sunspot	6.77E−03	6.67E−03	7.24E−03	1.18E−02	1.69E−02
N225	2.55E−02	2.48E−02	2.71E−02	3.32E−02	3.78E−02
DJI	2.01E−02	2.20E−02	2.43E−02	2.91E−02	3.53E−02
SSE	2.26E−02	2.37E−02	2.51E−02	3.17E−02	4.24E−02

Table 10 NMSE of the DIL algorithm with different parameter K on the first three datasets

Datasets	K				
	2	4	6	8	10
Mackey	5.38E−06	5.11E−06	5.83E−06	7.29E−06	8.32E−06
Lorenz	3.16E−06	2.56E−06	3.39E−06	5.73E−06	7.91E−06
Sunspot	5.27E−04	4.85E−04	5.93E−04	1.31E−03	1.95E−03

Table 11 MAE of the DIL algorithm with different parameter K on the last three datasets

Datasets	K				
	2	4	6	8	10
N225	2.03E−02	1.95E−02	2.21E−02	2.65E−02	3.17E−02
DJI	1.46E−02	1.62E−02	1.75E−02	2.28E−02	2.92E−02
SSE	1.69E−02	1.78E−02	1.90E−02	2.45E−02	3.47E−02

Table 12 RMSE of the DIL algorithm with different parameter T_k on six datasets

Datasets	T_k			
	10	20	30	40
Mackey	6.60E−04	5.63E−04	6.05E−04	6.14E−04
Lorenz	5.17E−04	3.56E−04	4.06E−04	4.01E−04
Sunspot	8.05E−03	6.61E−03	7.13E−03	7.31E−03
N225	2.88E−02	2.48E−02	2.44E−02	2.56E−02
DJI	2.38E−02	1.96E−02	1.92E−02	1.93E−02
SSE	2.66E−02	2.31E−02	2.15E−02	2.18E−02

Table 13 NMSE of the DIL algorithm with different parameter T_k on the first three datasets

Datasets	T_k			
	10	20	30	40
Mackey	6.33E−06	5.12E−06	5.92E−06	5.98E−06
Lorenz	3.43E−06	2.67E−06	2.85E−06	2.83E−06
Sunspot	6.71E−04	4.74E−04	5.56E−04	5.72E−04

Table 14 MAE of the DIL algorithm with different parameter T_k on the last three datasets

Datasets	T_k			
	10	20	30	40
N225	2.46E−02	1.92E−02	1.89E−02	2.03E−02
DJI	1.88E−02	1.43E−02	1.41E−02	1.42E−02
SSE	2.33E−02	1.84E−02	1.73E−02	1.75E−02

Table 15 Comparisons with the results from the literatures on Mackey–Glass

Prediction model	RMSE	NMSE
SL-CCRNN [7]	6.33E−03	2.79E−04
BPNN-GA-RA [59]	1.30E−03	/
RBF-OLS [60]	1.02E−03	/
LLNF-LoLiMot [60]	9.61E−04	/
CCRNN-NL [61]	1.07E−02	/
CICC-two-island [62]	8.47E−03	/
Proposed DIL	5.69E−04	5.18E−06

The mark “/” indicates that the contrastive algorithms have not listed results corresponding to the certain performance measurement in the reference paper

algorithm, the most critical part is the weight update rule, which is a brand-new and powerful rule. This rule has made a great contribution to the improvement of algorithm performance.

Table 16 Comparisons with the results from the literatures on Lorenz

Prediction model	RMSE	NMSE
SL-CCRNN [7]	6.36E−03	7.72E−04
ARMA-ANN [63]	8.76E−02	/
BPNN-GA-RA [59]	2.96E−02	/
CCRNN-NL [61]	1.36E−02	/
CICC-two-island [62]	9.15E−03	/
Proposed DIL	3.46E−04	2.60E−06

Table 17 Comparisons with the results from the literatures on Sunspot

Prediction model	RMSE	NMSE
SL-CCRNN [7]	1.66E−02	1.47E−03
ERNN [11]	1.29E−02	2.80E−03
HENNN-RA [55]	1.19E−02	5.90E−04
Taguchi’s DoE [56]	1.10E−02	5.04E−04
CCRNN-NL [61]	3.91E−02	/
CICC-two-island [62]	4.06E−02	/
Proposed DIL	6.72E−03	4.86E−04

Table 18 Comparisons with the results from the literatures on N225

Prediction model	RMSE	MAE
NBDM [58]	3.42E−02	2.97E−02
ELMAN [64]	4.21E−02	3.32E−02
MLP [65]	5.80E−02	4.56E−02
ANFIS [66]	1.27E−01	3.38E−02
SMN [67]	6.68E−02	5.39E−02
Proposed DIL	2.54E−02	1.96E−02

Table 19 Comparisons with the results from the literatures on DJI

Prediction model	RMSE	MAE
NBDM [58]	5.13E−02	2.74E−02
ELMAN [64]	6.14E−02	3.47E−02
MLP [65]	1.26E−01	7.05E−02
ANFIS [66]	1.97E−01	9.72E−02
SMN [67]	2.02E−01	1.14E−01
Proposed DIL	1.91E−02	1.42E−02

Table 20 Comparisons with the results from the literatures on SSE

Prediction model	RMSE	MAE
NBDM [58]	2.77E−02	2.23E−02
ELMAN [64]	5.74E−02	5.24E−02
MLP [65]	7.86E−02	5.89E−02
ANFIS [66]	4.78E−02	5.15E−02
SMN [67]	6.65E−02	4.85E−02
Proposed DIL	2.23E−02	1.73E−02

Table 21 Comparison of the average run times (s) of the corresponding algorithms on the six benchmark datasets over 20 repetitive runs

Datasets	Models					
	Mackey	Lorenz	Sunspot	N225	DJI	SSE
LLNF-LoLiMot [60]	3426.41	506.84	263.67	11.51	13.16	12.31
CICC-two-island [62]	2203.13	389.11	127.36	5.67	7.13	5.98
SL-CCRNN [7]	1009.74	221.06	81.21	3.05	4.77	3.12
RBF-OLS [60]	619.01	142.93	43.17	1.82	2.71	1.76
CCRNN-NL [61]	976.23	213.76	73.53	3.11	4.97	3.25
Proposed DIL	913.51	199.77	67.58	2.95	4.41	2.75

The previous description about the advantages of DIL, including promising performance, desirable robustness, and high stability, has been fully testified in the experimental part. In addition, as mentioned earlier, since DIL saves all the trained base models, the trained historical data can be discarded, so that the storage space will be saved. These characteristics indicate that, the proposed DIL model has a broad prospect in its application to TSP problems.

Compared with some of the other excellent algorithms in the literatures, the experimental results on six benchmark time series datasets, including two synthetic and four real-world datasets, fully confirm the superiority of the DIL algorithm.

With regard to the outlook for future works, firstly, the research of this work is only focused on single-step TSP problems, while in the future, our research work can be extended to multi-step TSP ones. Besides, in this work, the parameters of DIL are determined by the trial-and-error method, while in the future, we will consider the design of self-adaptive parameters for the implementation of the algorithm. Moreover, the proposed algorithm involves the generation of multiple base models. However, ensemble pruning techniques have not been employed. Therefore, in the future, we will appropriately incorporate ensemble pruning techniques into our developed algorithm to further boost its efficiency and efficacy.

Acknowledgements This work is supported by the National Natural Science Foundation of China under Grants No. 61473150.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

- Abdi J, Moshiri B, Abdulhai B, Sedigh AK (2013) Short-term traffic flow forecasting: parametric and nonparametric approaches via emotional temporal difference learning. *Neural Comput Appl* 23:141–159

- Aye GC, Balcilar M, Gupta R, Majumdar A (2015) Forecasting aggregate retail sales: the case of South Africa. *Int J Prod Econ* 160:66–79
- Li G, Wang S (2017) Sunspots time-series prediction based on complementary ensemble empirical mode decomposition and wavelet neural network. *Math Probl Eng* 2017:1–7
- Podsiadlo M, Rybinski H (2016) Financial time series forecasting using rough sets with time-weighted rule voting. *Expert Syst Appl* 66:219–233
- Gooijer JGD, Hyndman RJ (2006) 25 years of time series forecasting. *Int J Forecast* 22:443–473
- Chen D, Han W (2013) Prediction of multivariate chaotic time series via radial basis function neural network. *Complexity* 18:55–66
- Chandra R, Zhang MJ (2012) Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction. *Neurocomputing* 86:116–123
- Abiyev RH (2011) Fuzzy wavelet neural network based on fuzzy clustering and gradient techniques for time series prediction. *Neural Comput Appl* 20:249–259
- Castro JR, Castillo O, Melin P, Mendoza O, Rodríguezdiaz A (2010) An interval type-2 fuzzy neural network for Chaotic time series prediction with cross-validation and Akaike test. In: Kang JC, Schoch CL (eds) *Soft computing for intelligent control and mobile robotics*. Springer, Berlin, pp 269–285
- Lin CJ, Chen CH, Lin CT (2009) A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications. *IEEE Trans Syst Man Cybern Part C Appl Rev* 39:55–68
- Ma QL, Zheng QL, Peng H, Zhong TW, Xu LQ (2007) Chaotic time series prediction based on evolving recurrent neural networks. In: *Proceedings of 2007 international conference on machine learning and cybernetics*, vol 1–7, pp 3496–3500
- Donate JP, Li XD, Sanchez GG, de Miguel AS (2013) Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm. *Neural Comput Appl* 22:11–20
- Rivero CR (2013) Analysis of a Gaussian process and feed-forward neural networks based filter for forecasting short rainfall time series. In: *IEEE computational intelligence magazine*, pp 1–6
- Pucheta JA, Rodríguez Rivero CM, Herrera MR, Salas CA, Patiño HD, Kuchen BR (2011) A feed-forward neural networks-based nonlinear autoregressive model for forecasting time series. *Computación Y Sistemas* 14:423–435
- Babinec Š, Pospíchal J (2006) Merging echo state and feedforward neural networks for time series forecasting. In: Kollias SD, Stafylopatis A, Duch W, Oja E (eds) *Artificial neural networks – ICANN 2006*. ICANN 2006. Lecture Notes in Computer Science, vol 4131. Springer, Berlin, Heidelberg

16. Wang BH, Huang HJ, Wang XL (2013) A support vector machine based MSM model for financial short-term volatility forecasting. *Neural Comput Appl* 22:21–28
17. Miranian A, Abdollahzade M (2013) Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction. *IEEE Trans Neural Netw Learn Syst* 24:207–218
18. Wu Q (2010) The hybrid forecasting model based on chaotic mapping, genetic algorithm and support vector machine. *Expert Syst Appl* 37:1776–1783
19. Hansen JV, Nelson RD (1997) Neural networks and traditional time series methods: a synergistic combination in state economic forecasts. *IEEE Trans Neural Netw* 8:863–873
20. Vapnik VN (2000) *The nature of statistic learning theory*. Springer, Berlin
21. Suykens JAK, De Brabanter J, Lukas L, Vandewalle J (2002) Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing* 48:85–105
22. Laskov P, Gehl C, Kruger S, Muller KR (2006) Incremental support vector learning: analysis, implementation and applications. *J Mach Learn Res* 7:1909–1936
23. Ma JS, Theiler J, Perkins S (2003) Accurate on-line support vector regression. *Neural Comput* 15:2683–2703
24. Zhang YW (2009) Enhanced statistical analysis of nonlinear processes using KPCA, KICA and SVM. *Chem Eng Sci* 64:801–811
25. Cauwenberghs G, Poggio T (2000) Incremental and decremental support vector machine learning. In: *International conference on neural information processing systems*, pp 388–394
26. Zhou ZH, Chen ZQ (2002) Hybrid decision tree. *Knowl-Based Syst* 15:515–528
27. Hu LM, Shao C, Li JZ, Ji H (2015) Incremental learning from news events. *Knowl-Based Syst* 89:618–626
28. Xu X, Wang W, Wang JH (2016) A three-way incremental-learning algorithm for radar emitter identification. *Front Comput Sci* 10:673–688
29. Lange S, Zilles S (2012) Formal models of incremental learning and their analysis. In: *International joint conference on neural networks*, vol 4, pp 2691–2696
30. Giraud-Carrier C (2000) A note on the utility of incremental learning. *Ai Commun* 13:215–223
31. Xu SL, Wang JH (2016) A fast incremental extreme learning machine algorithm for data streams classification. *Expert Syst Appl* 65:332–344
32. Das RT, Ang KK, Quek C (2016) ieRSPOP: a novel incremental rough set-based pseudo outer-product with ensemble learning. *Appl Soft Comput* 46:170–186
33. Qin Y, Li D, Zhang A (2015) A new SVM multiclass incremental learning algorithm. *Math Probl Eng* 2015:1–5
34. Osorio FS, Amy B (1999) INSS: a hybrid system for constructive machine learning. *Neurocomputing* 28:191–205
35. Xing YL, Shi XF, Shen FR, Zhou K, Zhao JX (2016) A self-organizing incremental neural network based on local distribution learning. *Neural Netw* 84:143–160
36. Gu B, Sheng VS, Tay KY, Romano W, Li S (2015) Incremental support vector learning for ordinal regression. *IEEE Trans Neural Netw Learn Syst* 26:1403–1416
37. Hoya T, Constantinides AG (1998) An heuristic pattern correction scheme for GRNNs and its application to speech recognition. In: *Neural networks for signal processing VIII*, pp 351–359
38. Yamauchi K, Yamaguchi N, Ishii N (1999) Incremental learning methods with retrieving of interfered patterns. *IEEE Trans Neural Netw* 10:1351–1365
39. Tsoumakas G, Partalas I, Vlahavas I (2009) An ensemble pruning primer. In: Okun O, Valentini G (eds) *Applications of supervised and unsupervised ensemble methods*. *Studies in Computational Intelligence*, vol 245. Springer, Berlin, Heidelberg
40. Banfield RE, Hall LO, Bowyer KW, Kegelmeyer WP (2005) Ensemble diversity measures and their application to thinning. *Inf Fusion* 6:49–62
41. Zhou ZH, Wu JX, Tang W (2002) Ensembling neural networks: many could be better than all. *Artif Intell* 137:239–263
42. Vapnik V, Cortes C (1995) Support vector networks. *Mach Learn* 20:273–297
43. Drucker H, Burges CJC, Kaufman L, Smola A, Vapnik V (1997) Support vector regression machines. *Adv Neural Inf Process Syst* 9:155–161
44. Freund Y, Schapire RE (1995) A decision-theoretic generalization of on-line learning and an application to boosting. In: *Proceedings of the second European conference on computational learning theory*, pp 119–139
45. Sun Y, Wang XG, Tang X (2014) Deep learning face representation from predicting 10,000 classes. In: *IEEE conference on computer vision and pattern recognition*, pp 1891–1898
46. Lehman B, Sullins J, Daigle R, Combs R, Vogt K, Perkins L (2010) A time for emoting: when affect-sensitivity is and isn't effective at promoting deep learning. In: *International conference on intelligent tutoring systems*, pp 245–254
47. Graesser AC, Moreno KN, Marineau JC, Adcock AB, Olney AM, Person NK (2003) AutoTutor improves deep learning of computer literacy: is it the dialog or the talking head? *Artif Intell Educ* 97:47–54
48. Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. *IEEE Trans Pattern Anal Mach Intell* 35:1798–1828
49. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. *Science* 313:504–507
50. Salakhutdinov R, Hinton G (2009) Deep Boltzmann machines. *J Mach Learn Res* 5:1967–2006
51. Bengio Y, Lamblin P, Dan P, Larochelle H (2006) Greedy layer-wise training of deep networks. In: *International conference on neural information processing systems*, pp 153–160
52. Smolensky P (1986) Information processing in dynamical systems: foundations of harmony theory. In: Rumelhart DE, Group CP (eds) *Parallel distributed processing: explorations in the microstructure of cognition*, vol 1. MIT Press, Cambridge, pp 194–281
53. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S et al (2014) Generative adversarial nets. In: *International conference on neural information processing systems*, pp 2672–2680
54. Zhang HJ, Li JX, Ji YZ, Yue H (2017) Understanding subtitles by character-level sequence-to-sequence learning. *IEEE Trans Industr Inf* 13(2):616–624
55. Ardalani-Farsa M, Zolfaghari S (2010) Chaotic time series prediction with residual analysis method using hybrid Elman-NARX neural networks. *Neurocomputing* 73:2540–2553
56. Ardalani-Farsa M, Zolfaghari S (2013) Taguchi's design of experiment in combination selection for a Chaotic time series forecasting method using ensemble artificial neural networks. *Cybern Syst* 44:351–377
57. World Data Center for the Sunspot Index. <http://sidc.oma.be/>. Accessed May 2017
58. Zhou TL, Gao SC, Wang JH, Chu CY, Todo Y, Tang Z (2016) Financial time series prediction using a dendritic neuron model. *Knowl-Based Syst* 105:214–224
59. Ardalani-Farsa M, Zolfaghari S (2011) Residual analysis and combination of embedding theorem and artificial intelligence in Chaotic time series forecasting. *Appl Artif Intell* 25:45–73
60. Gholipour A, Araabi BN, Lucas C (2006) Predicting chaotic time series using neural and neurofuzzy models: a comparative study. *Neural Process Lett* 24:217–239

61. Chandra R, Chand S (2016) Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance. *Appl Soft Comput* 49:462–473
62. Chandra R (2015) Competition and collaboration in cooperative coevolution of elman recurrent neural networks for time-series prediction. *IEEE Trans Neural Netw Learn Syst* 26:3123–3136
63. Rojas I, Valenzuela O, Rojas F, Guillen A, Herrera LJ, Pomares H et al (2008) Soft-computing techniques and ARMA model for time series prediction. *Neurocomputing* 71:519–537
64. Elman JL (1990) Finding structure in time. *Cogn Sci* 14:179–211
65. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Anderson JA (ed) *Neurocomputing: foundations of research*. MIT Press, Cambridge, pp 318–362
66. Vairappan C, Tamura H, Gao S, Tang Z (2009) Batch type local search-based adaptive neuro-fuzzy inference system (ANFIS) with self-feedbacks for time-series prediction. *Neurocomputing* 72:1870–1877
67. Yadav RN, Kalra PK, John J (2007) Time series prediction with single multiplicative neuron model. *Appl Soft Comput* 7:1157–1163