Applied Soft Computing 66 (2018) 1–17

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Optimal Forecast Combination Based on Neural Networks for Time Series Forecasting

Lin Wang [a], Zhigang Wang [a], Hui Qu [b], Shan Liu [c,*]

[a] School of Management, Huazhong University of Science and Technology, Wuhan 430074, China
[b] School of Economics and Management, Hubei University of Technology, Wuhan 430068, China
[c] School of Management, Xi'an Jiaotong University, Xi'an 710049, China

ARTICLE INFO

Article history:
Received 24 September 2017
Received in revised form
31 December 2017
Accepted 3 February 2018
Available online 10 February 2018

Keywords:
Time series forecasting
forecast combination
artificial neural networks

ABSTRACT

Research indicates that forecast combination is one of the most important and effective approaches for time series forecasting. The success of forecast combination depends on how well component models are selected and combination weights are determined. A forecast combination model resulting from a new **n**eural **n**etwork**s**-based **l**inear **e**nsemble **f**ramework (NNsLEF) is proposed in this study. The principle of the proposed framework adheres to three primary aspects. (a) Four kinds of neural network models, namely, back-propagation neural network, dynamic architecture for artificial neural network, Elman artificial neural network, and echo state network, are selected as component forecasting models. (b) An **i**nput-**h**idden **s**election **h**euristic (IHSH) is designed to determine the input-hidden neuron combination for each component neural network. (c) An **i**n-sample **t**raining–**v**alidation **p**air-based **n**eural **n**etwork **w**eighting (ITVPNNW) mechanism is studied to generate the associated combination weights. In particular, the four neural network models are applied to impart their superior performance to the combination approach while maintaining their diversity. Meanwhile, IHSH is investigated to improve the performance of each component neural network model by attempting to solve the familiar overfitting problem of networks. Lastly, the ITVPNNW mechanism is studied to search for a set of appropriate combination weights that will primarily affect the accuracy of the linear ensemble framework.
Results: from experiments performed on eight time series data sets show that NNsLEF outperforms the four component neural network models and other well-recognized models.

1. Introduction

In the past few decades, time series analysis has become a popular research topic and has attracted a great deal of attention. Time series analysis has established itself as a powerful tool for characterizing complex systems from observed data [1–4]. Researchers have applied time series analysis in many fields, such as clustering [5], pattern recognition [6], classification [7] and prediction [8,9]. As a branch of time series analysis, time series forecasting plays an important role in practice applications. Examples include diverse forecasting applications in energy [10], finance [11], tourism [12,13], and electricity load [14,15]. However, improving the performance of forecasting is an important yet frequently difficult task. A substantial number of studies have been conducted and several approaches, such as autoregressive integrated moving average, support vector machine, and artificial neural networks (ANNs), have been proposed to address this issue. For situations where no dominant approach has been identified, forecast combination has been one of the most important, effective, and popular research perspectives applied since its introduction by Bates and Granger [16] in the 1960s. A theoretical justification of forecast combination can be established by viewing the problem from the perspective of Bayesian model averaging [17]. That is, several forecasting models can be tested and their forecasts are averaged according to the probabilities of the component models, in which knowledge of the precise data to generate a time series process is lacking. Forecast combination is motivated by an impressive result, which shows that this approach can generally yield more accurate and reliable results than single forecasting methods, as evidenced in the literature review [18–20].

A growing consensus suggests that forecast combination has advantages over a single model not only in terms of accuracy and error variability but also in simplifying model building and selec-

* Corresponding author.
E-mail addresses: wanglin@hust.edu.cn (L. Wang), zgwang2014@hust.edu.cn (Z. Wang), qhui733@gmail.com (H. Qu), shan.l.china@gmail.com (S. Liu).

https://doi.org/10.1016/j.asoc.2018.02.004
1568-4946/

tion, and the forecasting process as a whole [21]. In general, forecast combination can achieve superior performance based on the following four points. (a) Each component forecast may be insufficient because the best-trained model in the in-sample training period may not be the best for forecasting future values. (b) Component forecasts typically characterize the data-generating process of the time series from different and somewhat complementary perspectives. (c) Adaptive strategies for forecast combination can provide a complete picture from a number of partial solutions in a cooperative modular form, i.e., forecast combination allows component forecasts to "cover considerable grounds." (d) Forecast combination may mitigate structural breaks, model uncertainties, and model misspecifications, thereby improving the accuracy of forecasts. In summary, forecast combination can compensate for the drawbacks of individual forecasts, benefit from the interactions among individual forecasts, and mitigate the risks of using a single forecast.

1.1. Challenges of effective forecast combination

The superior performance of forecast combination has motivated scholars to research different combination approaches, and valuable knowledge has accumulated over the years. Despite all these works, two considerable challenges should be overcome.

1.1.1. Selection of the suitable component forecasting models

The first considerable challenge in forecast combination is selecting the component forecasting models to combine from an extensive pool of potential models. Forecast combination can inherit its forecasting performance from its component forecasting models, and thus, it is considered more superior than its component models. That is, forecast combination can achieve superior performance when the underlying components can provide good forecast. An implied condition for selection is to maintain the diversity of the models. The diversity of component forecasting models functions as a safeguard against focusing on a narrow specification. Such diversity can stem from using different forecasting models, model specifications, time series preprocessing methods, or exogenous input variables.

At present, it is difficult to achieve a universal guideline which is available for determining the best approach to select component forecasting models; however, different combination techniques have been proposed. One of the major techniques is to model the combination using ANNs. ANNs are flexible nonlinear data-driven models with attractive properties for forecasting. They have become one of the most accurate and widely used forecasting models [22,23]. The major advantage of ANNs is their flexible modeling capability that exhibits a self-adaptive and data-driven mechanism. ANNs have been proven to be universal and powerful approximators [24] that can suggest an appropriate data-generating process for both linear and nonlinear time series with different forms. Simultaneously, the ANN literature has strongly argued, with supporting empirical evidence and large-scale comparison studies, that instead of selecting a single ANN that may fail to achieve the desired accuracy, considering a combination of different ANN models or a combination of ANN models with other models is preferable [9,21,22,25,26].

Although the powerful approximation capabilities of ANNs enable these models to achieve tremendous success in modeling time series data, these capabilities also substantially complicate model specification. Selecting optimal architecture and parameters are important to achieve success in neural network applications [27,28]. Most widely used ANNs are multilayer perceptrons (MLPs). MLPs use the McCulloch-Pitts neuron model, which is based on an additive aggregation function. Thus, in an MLP designed for time series forecasting, certain design factors, such as the architecture of a neural network, and the number of input and hidden neurons significantly influence the forecast accuracy of a neural network [23]. The number of output neurons is typically set to 1. It is to say that neural networks encounter problems in specifying optimal topology and parameters although they are a promising alternative to traditional forecasting techniques. Therefore, numerous neural networks, each of which has different architecture and parameter values, should be built before finding a satisfactory model. Numerous studies have been performed and good suggestions have been offered. For example, more than one hidden layer can be found in a neural network; however, a neural network with a single hidden layer is preferable [23,24,29,30]. The number of input neurons helps indicate the relation between observations; hidden neurons define the properties within data and help establish a nonlinear relationship between the input and the output. These factors are fairly effective in improving the performance of a neural network [31–33]. However, these factors may change depending on the problem being addressed. Therefore, determining these factors is a vital and difficult issue for neural network applications. Although a systematic theory remains unavailable, several heuristic approaches have been proposed in the literature [31–37]. Nevertheless, none of the options can work efficiently for all neural networks.

1.1.2. Determination reasonable combination weights

Another essential challenge is weighting the contribution of each component model in a combination scheme. That is, determining the combination weights of component models, which will primarily affect the accuracy of the combination model. To solve this problem, a number of weight combination approaches have been proposed. The simplest approach is the static type; in this approach, a unique weighting vector is estimated and applied over the entire forecast horizon based on historical data [38]. Statistical averaging techniques, such as simple average (mean), trimmed mean, and median, used in an unweighted or weighted manner, are the most traditional and fundamental weight combination approaches because they do not explicitly determine combination weights. The simple average is the easiest and simplest weight combination approach that assigns equal weights to all component forecasts. Research evidence shows that the simple average can achieve remarkably good accuracy and outperform some other advanced linear combination approaches [19,39,40]. However, simple average is highly sensitive to extreme values. The median is also a frequently used statistical averaging technique [41,42]. This technique may filter outliers included in the distribution of forecasts, which may negatively affect the performance of mean-based ensemble forecasts. The trimmed mean is another widely used combination approach [42]. This approach computes the simple average by discarding an equal number of smallest and largest component forecasts. Both simple average and median are special cases of the trimmed mean that corresponds to no trimming and full trimming, respectively. A trimmed mean is feasible only when more than three component models are used. That is, the number of component models n should satisfy the constraint $n \geq 3$. For $n = 3, 4$, a trimmed mean is the same as a median. Although these statistical averaging techniques have been widely used in different forecasting applications, researchers have suggested that significant differences do not exist among them [43]. Research has shown that there are some other statistical combination techniques can be used to determine combination weights as documented in the literature [44]. These techniques can be described as minimizing error sum of squares, minimizing the sum of absolute error, minimizing the maximum absolute error, arithmetic average value method, the reciprocal prediction error sum of squares method, the reciprocal mean square error method, simple weighted average method and binomial coefficient method. Relative to the static type of weight combination approaches, other techniques have also

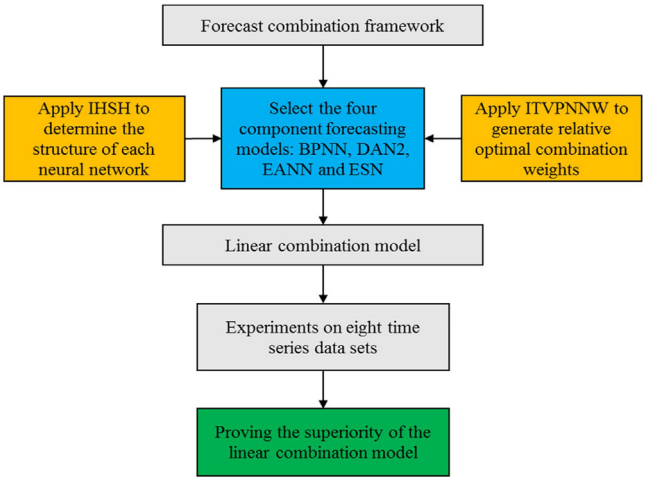

Fig. 1. Flowchart for forecast combination used in this study.

been investigated. Unlike static approaches, these techniques can be considered "dynamic" to a certain extent. The common characteristic of these approaches is a weight generation framework that is applied to dynamically determine the corresponding optimal weights. The main advantage of these approaches is the capability to relax in-sample performance dependence and abstract statistical complexity. Research has shown that these dynamic approaches can improve forecasting accuracy as documented in the literature [20,29,45–47].

1.2. Contributions

On the basis of the preceding discussions, a forecast combination model resulting from a new **n**eural **n**etwork**s**-based **l**inear **e**nsemble **f**ramework (NNsLEF) is proposed for time series forecasting in this study. The flowchart of this study is shown in Fig. 1. The light blue part, orange parts and green part show the difficulties, novelty and efficiency of the proposed approach, respectively.

The proposed framework can merge the advantages of component neural networks and dynamic weight combination approaches to improve forecasting performance. The main contributions of the present study are threefold:

(a) Four widely used neural networks, namely, back-propagation neural network (BPNN), dynamic architecture for ANN (DAN2), Elman ANN (EANN), and echo state network (ESN), which belong to four different categories of neural networks, are selected as the component models of the combination approach. The selection rationales of these models are also detailedly analyzed.
(b) An **i**nput-**h**idden **s**election **h**euristic (IHSH) is designed to determine the number of input and hidden neurons for each component neural network.
(c) A dynamic weight combination approach, called the **i**n-sample **t**raining–**v**alidation **p**air-based **n**eural **n**etwork **w**eighting (ITVPNNW), is used to generate relative optimal combination weights.

In a word, contributions (a) and (b) propose a feasible and potential solution for the selecting the underlying component forecasting models. Meanwhile, contribution (c) mainly helps overcome the challenge of reasonable weighting the contribution of each component model in the combination scheme. The results of the experiments conducted on eight time series data sets show that the proposed combination approach outperforms the four component models and other well-recognized models.

The rest of the paper is organized as follows. In Section 2, the concepts of the four selected neural networks, namely, BPNN, DAN2, EANN, and ESN, as well as that of an artificial intelligence approach, namely, the backtracking search optimization algorithm (BSA), are reviewed. Section 3 explains the linear forecast combination methodology and introduces the proposed forecast combination approach, namely, NNsLEF. Section 4 describes a series of experiments that are intended to test and verify the effectiveness of the proposed approach. Section 5 presents the conclusions and possible future extensions of this work.

2. Time series forecasting models

A time series is defined as a sequential collection of observations that is recorded in consecutive time periods. It can be classified into two kinds: univariate and multivariate. The former is an important data type of time series that is widely researched in the literature. A univariate time series can be represented as $\mathbf{Y} = [y_1, y_2, \ldots, y_N]^T$, where y_t is the observation at time period t. Forecasting is one of the major tasks required for a time series because it is a basic knowledge-intensive task. Conceptually, time series forecasting involves projecting the desired number of future values through models. The risk in using only one model is relatively considerable because even an "optimal" model can be prone to misspecification and inadequacy. However, combining several "non-optimal" models can closely approximate the actual data generation process as stated in the literature [48–51]. That is, forecast combination, as a successful alternative to individual forecast techniques, can overcome a number of inevitable limitations in using individual models. A forecast combination system inherits its forecasting performance from its component forecasting models, and thus, selecting appropriate models to combine from an extensive pool of candidates is essential. Although each component model is not required to be optimal, the selected models should neither include models with extremely poor performance nor discard any potentially superior models. Moreover, model diversity should be maintained. Determining the number of component models is another factor that significantly affects the accuracy of forecast combination. However, this factor is difficult to determine because of the lack of theoretical guidelines in this regard. From experience and the experimental results, four or five component models in a forecast combination approach are good options in improving accuracy [29,40,49,52].

In this study, four computational intelligence-based neural networks from diverse categories are selected to verify the superior performance and the diversity of the proposed combination approach (NNsLEF). These models are BPNN, DAN2, EANN, and ESN. Diversity is also achieved by considering different combinations of input and hidden neurons for each neural network. In the following sections, the basic concepts and modeling approaches of these neural networks are briefly reviewed. Meanwhile, the four models are compared with one another, their respective pros and cons are presented, and the rationales behind their selection are discussed.

2.1. BPNN model

The BPNN model, introduced by Rumelhart and McClelland [53], is one of the most utilized feedforward ANNs (FANNs) for time series forecasting [23,30,48]. It is a critically acclaimed model because of its nonparametric and nonlinear modeling capacity, strong adaptability, and parallel computing capability. The standard BPNN is composed of an input layer, one or more hidden layers, and an output layer. In general, a BPNN with a single hidden layer (Fig. 2) can generate the desired accuracy for time series forecasting applications [30].

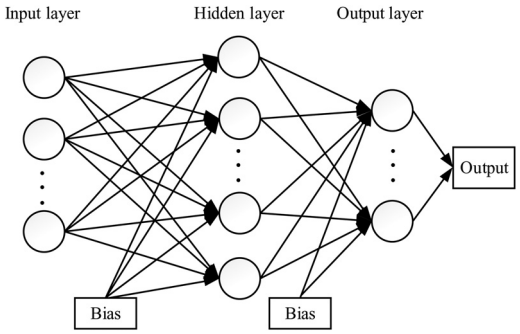

Fig. 2. Single hidden layer BPNN network structure.

BPNN is well-known for its error back-propagation learning algorithm while information is transmitted forward. This model is a mentor learning algorithm of gradient descent. However, one flaw of this learning algorithm is that the final training result easily falls into the local minimum instead of into the global optimum [30]. A slight misspecification of the parameters may yield disastrous outputs. Moreover, the BPNN model is ineffective in modeling linear time series and its forecasting accuracy is sensitive to the underlying architecture [54,55].

2.2. DAN2 model

The DAN2 model belongs to the category of FANN models but uses a relatively different architecture from those of traditional FANNs [56]. DAN2 can produce good results in time series forecasting [56,57]. The general philosophy of the DAN2 model is based on the principle of learning and accumulating knowledge at each layer, propagating and adjusting this knowledge forward to the next layer, and repeating these steps until the desired network performance criteria are achieved. That is, both error and information are transmitted forward in DAN2. The general DAN2 architecture is presented in Fig. 3.

Although the structure of DAN2 appears complicated, it is actually similar to the structure of classic FANN. The DAN2 architecture essentially comprises an input layer, one or more hidden layers, and an output layer. The input layer accepts external data into the model. Once the input nodes (neurons) have been identified, all the samples in the training data set are simultaneously used to train the network to minimize a specified training error measure. This feature differs from those of traditional FANNs. Each hidden layer is fixed with four nodes: a "C" node, a "CAKE" node, and two "CURNOLE" nodes. The final "CAKE" node represents the output. The training process begins with a special layer where the "CAKE" node captures the linear component of the input data. Thus, the input of the "CAKE" node is a linear combination (weighted sum) of the input variables and a "C" node. These weights are easily obtained through classical linear regression or artificial intelligence approaches (e.g., BSA). Hidden layers are sequentially and dynamically generated until a series of stopping criteria are achieved. The model can be simplified into a linear model if the desired accuracy level is reached during the first training step.

Both the BPNN and DAN2 models belong to the category of FANN. However, DNA2 uses an architecture that differs from that of the BPNN model in several ways, including the utilization of input records, the choice of the transfer function, the structure and number of hidden layers, and the relationships among interior nodes [57]. Moreover, DAN2 adopts a training mechanism that differs from that of BPNN, in which the training process begins with a special layer where the CAKE node captures the linear component of all the training samples. If the desired accuracy level is reached during the first training step, then the relationship is concluded to be linear and the training process is terminated. Subsequently, the DAN2 model degenerates into a linear model. This outcome has the additional effect of providing extra diversification away from nonlinear models.

In this work, BSA is briefly introduced. This novel differential evolution (DE) algorithm was proposed by Civicioglu [58]. The unique mechanism and simple structure of BSA enable it to solve numerical optimization problems quickly and successfully. The initial weights and global optimal parameters for DAN2 are optimized using BSA during the training process in this study.

2.3. EANN model

FANN models, including BPNN and DAN2, recognize the spatial relationship among successive observations but ignore the temporal relation. A recurrent neural network (RNN) provides a good concept to overcome this limitation by creating an internal state of the network that will allow it to exhibit a dynamic temporal behavior.

The EANN model is a local recurrent neural network that presents a context layer and feedback connections which make the network dynamic and able to perform mappings of input and target patterns. It was first proposed by Elman [59] to address the speech signal processing problem, but it could simultaneously provide promising outcomes for time series forecasting [29,50,60,61]. A Standard EANN model mainly consists of four layers (Fig. 4): an input layer, a context layer, a hidden layer, and an output layer. The extra context layer, which makes a copy of the hidden layer output in the previous time steps, preserves the records of previous network operations. This network possesses massive parallel connections not only between the hidden and output layers but also between the hidden and input layers and the context nodes. The

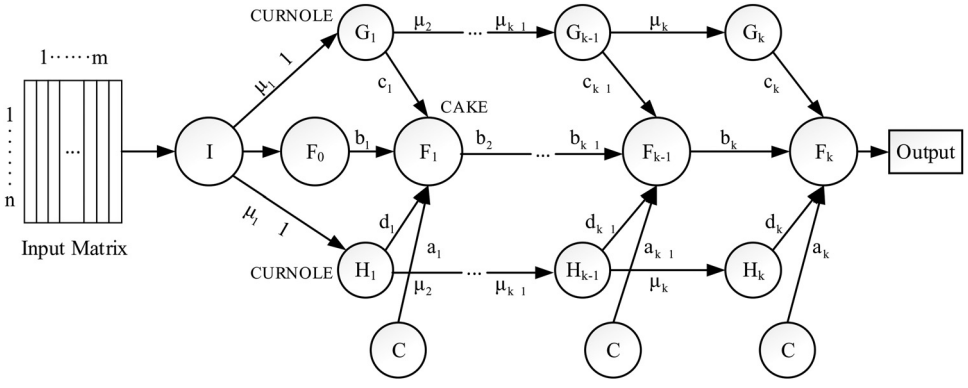

Fig. 3. DAN2 network architecture.

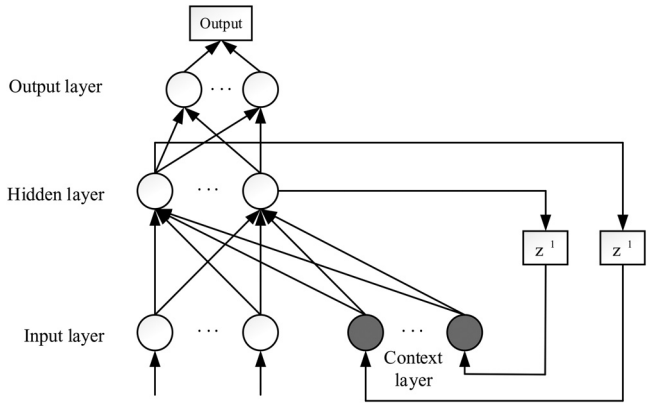

Fig. 4. EANN architecture.

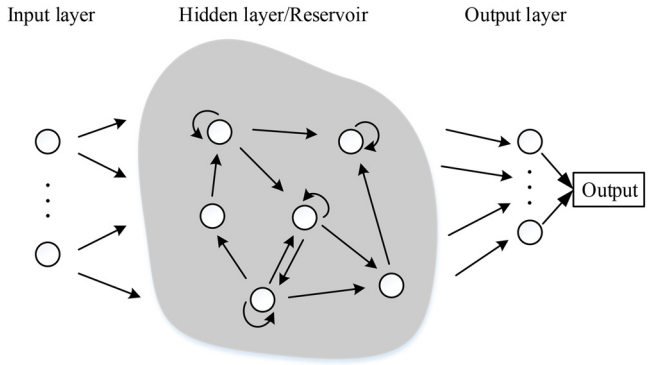

Fig. 5. ESN network architecture.

self-connections make the network sensitive to historical input, thereby allowing it to perform nonlinear time-varying mappings of input and target patterns [60,61]. Minimal work has been conducted on EANN architecture selection, but a widely agreed fact is that EANNs require considerably more hidden nodes than FANNs to adequately model temporal relations which will increases the computational cost [50].

2.4. ESN model

Along with EANN, the ESN model, proposed by Jaeger [62], is also a type of RNN. ESN is coupled with a "time parameter," and thus, it is highly effective for time series forecasting [48,63–66]. Relative to the aforementioned three neural networks, ESN has the simplest network structure. The widely used ESN can be partitioned into three components: an input layer, a hidden layer, and an output layer, as illustrated in Fig. 5. The hidden layer is also called the reservoir, which is randomly initialized. In general, the relevant information of the inputs can be represented through the internal states of a reservoir that is generated when inputs are fed into ESN. Accordingly, ESN can perform nonlinear time-varying mappings of input and target patterns by utilizing the dynamics of the reservoir instead of those of the original inputs.

The most distinctive characteristic of ESN is that only the connection weights from the reservoir neurons to the output layer neurons are required to be trained, whereas all other connection weights are randomly fixed and remain unchanged [62]. In addition, the adaptation can be regarded as a simple linear regression problem given that echo states have satisfactory richness. This feature considerably simplifies the training process. The problems in BPNN, DAN2, and EANN, including network construction and connection weight adaptation problems, are nearly completely evaded; ESN facilitates the practical application of RNNs and has been proven to exhibit superior performance in time series forecasting [48,63–66].

3. Proposed linear combination method

3.1. Linear combination of time series forecasts

Combining multiple time series forecasts into forecast combination is a successful alternative to using only a single model for time series forecasting [20]. In a linear ensemble framework, let $\boldsymbol{Y} = \left[y_1, y_2, \ldots, y_{N_0}\right]^{\mathrm{T}}$ be the index sequence of the forecasting objects and $\hat{Y}^{(i)} = \left[\hat{y}_1^{(i)}, \hat{y}_2^{(i)}, \ldots, \hat{y}_{N_0}^{(i)}\right]^{\mathrm{T}}$ $(i = 1, 2, \ldots, n)$ denote the forecasted output of the ith component forecasting model. The final combination forecast $\hat{Y}$ from the linear combination is generally calculated according to Eq. (1) as follows:

$$\hat{Y} = \sum_{i=1}^{n} w_i \hat{Y}^{(i)}, \tag{1}$$

where n is the number of component forecasting models, and N_0 is the number of desired forecasting objects. The terms $w_i\,(i = 1, 2, \ldots, n)$ denote the combination weights, which are frequently assumed to be unbiased and non-negative, i.e., $\sum_{i=1}^{n} w_i = 1$ and $w_i \geq 0\ \forall\, i$, to ensure that the combination model is reasonable. To achieve the maximum benefit of a linear ensemble framework, the linear combination forecast $\hat{Y}$ and the component forecast $\hat{Y}^{(i)}$ should satisfy the property described in Eq. (2) as follows [29]:

$$\lambda\left(\boldsymbol{Y}, \hat{Y}\right) \leq \lambda\left(\boldsymbol{Y}, \hat{Y}^{(i)}\right), \forall i (i = 1, 2, \ldots, n), \tag{2}$$

where λ is a forecasting error measure, e.g., one of the frequently used evaluation metrics such as the root mean square error (RMSE), mean absolute percentage error (MAPE), mean squared error (MSE), or mean absolute error (MAE).

3.2. Proposed linear combination method

The proposed forecasting model, i.e., NNsLEF, is a linear ensemble framework based on four neural network models. The block diagram of NNsLEF is shown in Fig. 6.

The principle of the proposed NNsLEF adheres to three primary aspects. (a) Four kinds of neural network models, namely, BPNN, DAN2, EANN, and ESN, are selected as component forecasting models of the linear ensemble framework. The four neural network models are applied to impart their superior performance while maintaining their diversity. (b) An input-hidden selection heuristic (IHSH) is designed to determine the input-hidden neuron combination for each component neural network. IHSH is investigated to improve the performance of each component neural network model. (c) An in-sample training–validation pair-based neural network weighting (ITVPNNW) mechanism is studied to generate the associated set of appropriate combination weights. It is because that the set of combination weights will primarily affect the accuracy of the linear ensemble framework.

NNsLEF can provide improved forecasting performance by merging the advantages of the component neural networks (BPNN, DAN2, EANN, and ESN) and the dynamic weight combination approach (ITVPNNW). It can be divided mainly into four phases as follows:

(a) Building in-sample training–validation pairs (Section 3.2.1);
(b) Selecting the optimal structure for each component model (Section 3.2.2);

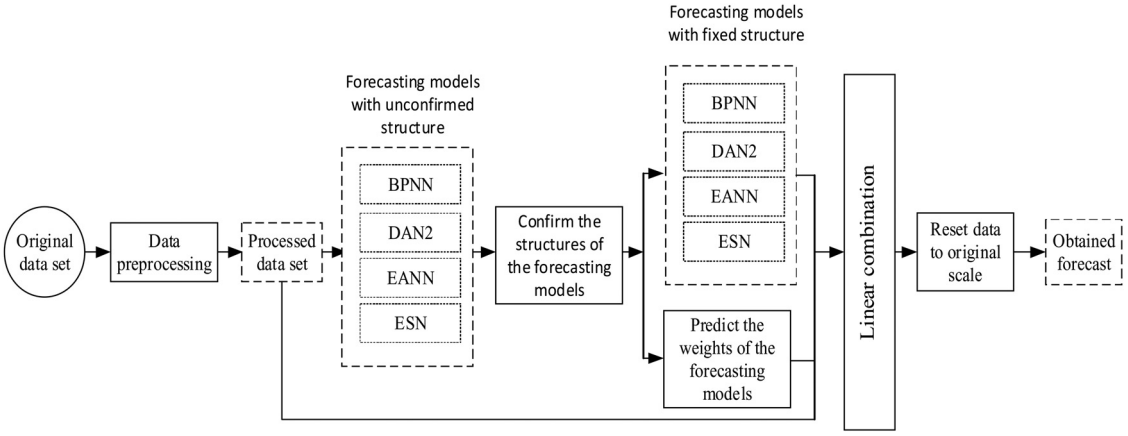

Fig. 6. Block diagram of the proposed linear ensemble framework.

(c) Determining the combination weights of the models (Section 3.2.3);
(d) Combining forecasting models (Section 3.2.4).

3.2.1. Building in-sample training–validation pairs

Dividing the in-sample training data into a number of consecutive in-sample training–validation pairs is necessary at the beginning of the proposed model. For example, the time series data set $\mathbf{Y} = [y_1, y_2, \ldots, y_N]^{\mathrm{T}}$ is first divided into the in-sample training data set $\mathbf{Y}_{\mathrm{tr}} = \left[y_1, y_2, \ldots, y_{N_{\mathrm{tr}}}\right]^{\mathrm{T}}$ and the out-of-sample testing data set $\mathbf{Y}_{\mathrm{ts}} = \left[y_{N_{\mathrm{tr}}+1}, y_{N_{\mathrm{tr}}+2}, \ldots, y_{N_{\mathrm{tr}}+N_{\mathrm{ts}}}\right]^{\mathrm{T}}$. In these data sets, N, N_{tr}, and N_{ts} are the number of observations in the time series, training, and testing data sets, which satisfy the relationship $N = N_{\mathrm{tr}} + N_{\mathrm{ts}}$. Then, the in-sample training data set $\mathbf{Y}_{\mathrm{tr}} = \left[y_1, y_2, \ldots, y_{N_{\mathrm{tr}}}\right]^{\mathrm{T}}$ is subdivided into a suitable number M ($M \ll N_{\mathrm{tr}}$) of consecutive in-sample training–validation pairs $\left(\mathbf{Y}_{\mathrm{tr}}^{(j)}, \mathbf{Y}_{\mathrm{vd}}^{(j)}\right)$ $(j = 1, 2, \ldots, M)$ through Eq. (3) as follows [29]:

$$\begin{aligned} &\mathbf{Y}_{\mathrm{tr}}^{(j)} = \left[y_1, y_2, \ldots, y_{base+j\text{-}1}\right]^{\mathrm{T}}, \\ &\mathbf{Y}_{\mathrm{vd}}^{(j)} = \left[y_{base+j}, y_{base+j+1}, \ldots, y_{base+j+(N_{\mathrm{vd}}\text{-}1)}\right]^{\mathrm{T}}, \\ &\forall j = 1, 2, \ldots, M, \end{aligned} \tag{3}$$

where N_{vd} is the number of observations in the in-sample validation subset, and the term $base$ ($base = N_{\mathrm{tr}} - N_{\mathrm{vd}} - M + 1$) is the number of observations in the first in-sample training subset $\mathbf{Y}_{\mathrm{tr}}^{(1)}$. The sum number of observations in the last in-sample training and validation subsets is essentially equal to N_{tr}. From Eq. (3), a new in-sample training–validation pair $\left(\mathbf{Y}_{\mathrm{tr}}^{(j+1)}, \mathbf{Y}_{\mathrm{vd}}^{(j+1)}\right)$ is produced through the following steps: (a) adding an observation y_{base+j} into the end of the previous in-sample training subset $\mathbf{Y}_{\mathrm{tr}}^{(j)}$ and (b) adding an observation $y_{base+j+N_{\mathrm{vd}}}$ into the previous in-sample validation subset $\mathbf{Y}_{\mathrm{vd}}^{(j)}$ while deleting the same observation y_{base+j} from its beginning. Therefore, the number of observations in the training subset increases by one at each step, whereas the number of observations in the validation subset remains constant throughout the process. An example for building in-sample training–validation pairs is shown in Fig. 7. In this example, the corresponding parameters are set as $N_{\mathrm{tr}} = 20$, $N_{\mathrm{vd}} = 4$, and $M = 4$. The restriction $M \ll N_{\mathrm{tr}}$ attempts to guarantee that the subsequent training process will not be sensitive to the outliers in the time series.

$$\begin{aligned} j = 1: &\quad \mathbf{Y}_{\mathrm{tr}}^{(1)} = \left[y_1, y_2, \ldots, y_{13}\right]^{\mathrm{T}}, \quad \mathbf{Y}_{\mathrm{vd}}^{(1)} = \left[y_{14}, y_{15}, y_{16}, y_{17}\right]^{\mathrm{T}}; \\ j = 2: &\quad \mathbf{Y}_{\mathrm{tr}}^{(2)} = \left[y_1, y_2, \ldots, y_{14}\right]^{\mathrm{T}}, \quad \mathbf{Y}_{\mathrm{vd}}^{(2)} = \left[y_{15}, y_{16}, y_{17}, y_{18}\right]^{\mathrm{T}}; \\ j = 3: &\quad \mathbf{Y}_{\mathrm{tr}}^{(3)} = \left[y_1, y_2, \ldots, y_{15}\right]^{\mathrm{T}}, \quad \mathbf{Y}_{\mathrm{vd}}^{(3)} = \left[y_{16}, y_{17}, y_{18}, y_{19}\right]^{\mathrm{T}}; \\ j = 4: &\quad \mathbf{Y}_{\mathrm{tr}}^{(4)} = \left[y_1, y_2, \ldots, y_{16}\right]^{\mathrm{T}}, \quad \mathbf{Y}_{\mathrm{vd}}^{(4)} = \left[y_{17}, y_{18}, y_{19}, y_{20}\right]^{\mathrm{T}}. \end{aligned}$$

Fig. 7. Example for building training–validation pairs.

3.2.2. Selecting the optimal structure for each component model using IHSH

Selecting an optimal structure is important to achieve success in neural network applications. Most traditional and universal methods for selecting the structure of a neural network is based on the in-sample training–validation pair $\left(\mathbf{Y}_{\mathrm{tr}}^{(M)}, \mathbf{Y}_{\mathrm{vd}}^{(M)}\right)$. For example, the structure with the smallest error on the validation subset $\mathbf{Y}_{\mathrm{vd}}^{(M)}$ is believed to be optimal, and thus, is selected from all the candidates for a given neural network. Such selection is based on the assumption that this "optimal" structure can provide the best solution for the current problem. Although this traditional selection strategy has been popular for many years, the structure obtained using this method may overfit the training phase and may be peculiar to the validation subset $\mathbf{Y}_{\mathrm{vd}}^{(M)}$, thereby resulting in poor performance during the testing phase.

To overcome this familiar overfitting problem in a given neural network, searching the space of all the structures of the neural network and conducting trials in each structure repeated with consecutive in-sample training–validation pairs $\left(\mathbf{Y}_{\mathrm{tr}}^{(j)}, \mathbf{Y}_{\mathrm{vd}}^{(j)}\right)$ $(j = 1, 2, \ldots, M)$ can be directed to find the optimal structure. An IHSH is designed following this concept to determine the optimal structures of the selected models. The basic principle of IHSH is based on the findings obtained from previously conducted studies in which input neurons are considered more important than hidden neurons in the performance of neural networks [31–33,67,68]. For clarity, a structure for the selected model, i.e., BPNN, EANN, or ESN, comprises a combination of input and hidden neurons. The same concept applied to the DAN2 model means a combination of input neurons and hidden layers. The pseudocode of IHSH is presented in Algorithm 1.

Algorithm 1. IHSH: Confirm the structure of the ith forecasting model.

Input: in-sample training–validation pairs $\left(\mathbf{Y}_{\text{tr}}^{(j)}, \mathbf{Y}_{\text{vd}}^{(j)}\right)(j=1,2,\ldots,M)$, validation subset size N_{vd}, total number κ_i of structures for the ith model, and trail times τ
Output: optimal structure for the ith forecasting model
1. **for** $j=1$ to M **do**
2. **for** $m=1$ to κ_i **do**
3. Fit $\mathbf{Y}_{\text{tr}}^{(j)}$ into the ith model with the mth candidate structure (denoted as $\Gamma_i^{(m)}$) and use the fitted model to predict $\mathbf{Y}_{\text{vd}}^{(j)}$ for τ times, then calculate the average $\hat{\mathbf{Y}}_{\text{vd}}^{(j)(m)}$.
4. Compute the error $\lambda\left(\hat{\mathbf{Y}}_{\text{vd}}^{(j)(m)}, \mathbf{Y}_{\text{vd}}^{(j)}\right)$ using the error measure RMSE (denoted as $\text{RMSE}_j^{(m)}$). Use it as the error of the forecasting model with the mth candidate structure for $\mathbf{Y}_{\text{vd}}^{(j)}$, and then add it to the set Θ_i^j.
5. **end**
6. Determine the minimum RMSE_j^{best} from the set Θ_i^j, and add the corresponding structure Γ_i^{best} to the set Ω_i.
7. **end**
8. Apply **Heuristic 1** to choose the most optimal structure from Ω_i as the final confirmed structure for the ith forecasting model.

The main procedure of IHSH for selecting the optimal structure of the ith forecasting model can be divided into two steps. The first step is to train the component model with consecutive in-sample training–validation pairs $\left(\mathbf{Y}_{\text{tr}}^{(j)}, \mathbf{Y}_{\text{vd}}^{(j)}\right)$ ($j = 1, 2, \ldots, M$). For example, for each pair $\left(\mathbf{Y}_{\text{tr}}^{(j)}, \mathbf{Y}_{\text{vd}}^{(j)}\right)$, the mth ($m = 1, 2, \ldots, \kappa_i$) candidate structure of the ith model is trained on the training subset $\mathbf{Y}_{\text{tr}}^{(j)}$ and used to forecast the corresponding validation subset $\mathbf{Y}_{\text{vd}}^{(j)}$ for τ times, collect the average forecasted values $\hat{\mathbf{Y}}_{\text{vd}}^{(j)(m)}$, and add it into the set Θ_i^j. In this case, κ_i is the number of candidate structures designed for the ith forecasting model. After the set Θ_i^j is obtained completely, an error measure is applied to each $\left(\hat{\mathbf{Y}}_{\text{vd}}^{(j)(m)}, \mathbf{Y}_{\text{vd}}^{(j)}\right)$ ($m = 1, 2, \ldots, \kappa_i$) in the set Θ_i^j to acquire a space of error. Then, for all candidate structures of the ith model, the $\left(\hat{\mathbf{Y}}_{\text{vd}}^{(j)(m)}, \mathbf{Y}_{\text{vd}}^{(j)}\right)$ that generates the smallest error is selected, and the corresponding structure $\Gamma_i^{(m)}$ is regarded as a local optimal structure and added into the set Ω_i. The currently applied error measure is the frequently used evaluation metric RMSE, which is defined in Eq. (4) as follows:

$$\text{RMSE} = \sqrt{\frac{1}{k}\sum_{t=1}^{k}(\hat{y}_t - y_t)^2}, \tag{4}$$

where $\hat{y}_t$ and y_t are the forecasted value and actual value, respectively; and k is the number of forecasting objects. In addition to RMSE, other error measures, namely, MAPE, MSE, and MAE, will be employed in the following paragraphs. These error measures exhibit the following advantages and disadvantages. MAE and MSE are effective in evaluating the total absolute forecasting error, but are highly sensitive to extreme errors; RMSE has properties similar to those of MAE, but is considerably less sensitive to extreme errors and is more stable; MAPE evaluates the percentage of the average absolute error and is independent of the measurement scale [29]. These error measures are defined in Eqs. (5), (6) and (7) as follows:

$$\text{MAE} = \frac{1}{k}\sum_{t=1}^{k}|\hat{y}_t - y_t|, \tag{5}$$

$$\text{MSE} = \frac{1}{k}\sum_{t=1}^{k}(\hat{y}_t - y_t)^2, \tag{6}$$

$$\text{MAPE} = \frac{1}{k}\sum_{t=1}^{k}\frac{|\hat{y}_t - y_t|}{y_t}. \tag{7}$$

The second step is to select the global optimal structure for the component model. A local optimal structure set Ω_i for the ith model has been obtained, and thus, the final optimal structure

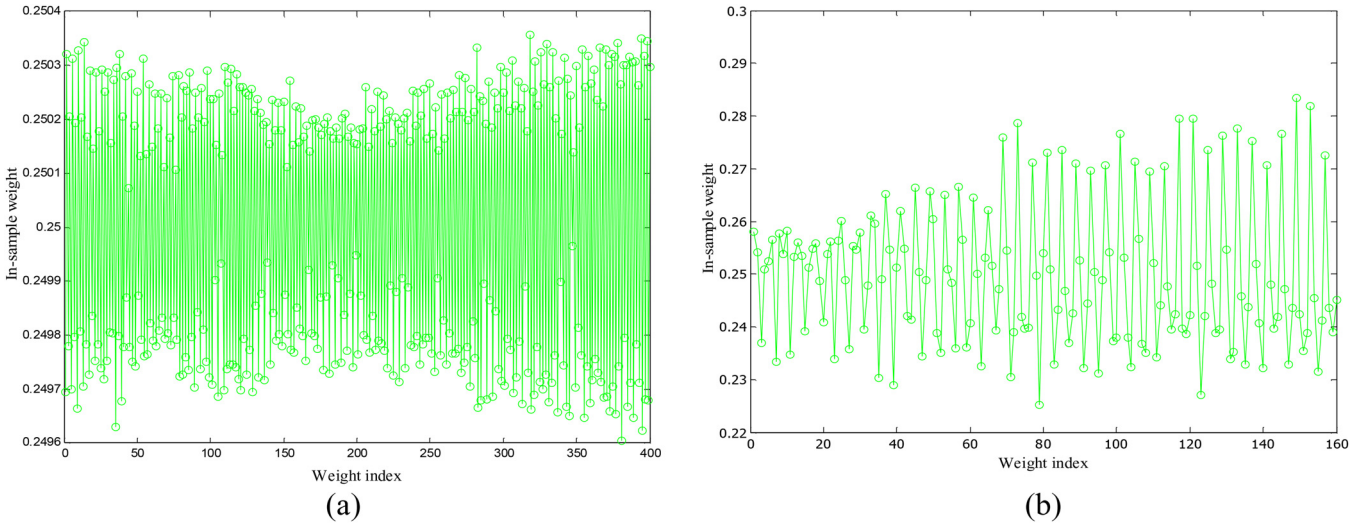

Fig. 8. Recurring in-sample weight patterns for the time series: (a) River flow and (b) Vehicles.

can be selected from the set for later use. A heuristic is designed for this purpose, the pseudocode of which is shown in Heuristic 1. The basic principle of the heuristic is based on the notion that the number of input neurons has a relative priority over that of hidden neurons. The process of Heuristic 1 is as follows. First, the elements in the set Ω_i is classified into different subsets according to the number of input neurons. That is, the structures with the same number of input neurons are collected in the same subset $\tilde{\Omega}_i^k$. Second, the process is applied to find the subset(s) with the maximum size $\eta_i^{\max}$. Third, the process is applied to denote the subset that has the same size as $\eta_i^{\max}$ with $\tilde{\Omega}_i^{best}$, subdivide it into a series of sub-subsets according to the hidden neurons/hidden layers, determine the sub-subset with the maximum size $\upsilon_i^{t\max}$, and denote the corresponding structure of the sub-subset with Δ_i^{tbest}. Fourth, the current maximum size $\upsilon_i^{\max}$ is compared with $\upsilon_i^{t\max}$. If $\upsilon_i^{t\max}$ is larger than $\upsilon_i^{\max}$, then $\upsilon_i^{t\max}$ and Δ_i^{tbest} will be assigned to $\upsilon_i^{\max}$ and Δ_i^{best}, respectively; otherwise, no step is performed. Fifth, the third and fourth steps are repeated until all the subsets with the same maximum size $\eta_i^{\max}$ are checked. Finally, the optimal structure Δ_i^{best} is obtained.

Heuristic 1. Select the global optimal structure from the structure set Ω_i.

Input: local optimal structure set Ω_i and the number of in-sample training–validation pairs M

Output: optimal structure Δ_i^{best}

1. $\sigma = 1$, $\tilde{\Omega}_i^{\sigma} = \{\ \}$.
2. **for** $j = 1$ to M **do**
3. Choose the jth element of Ω_i, which is denoted as $\Omega_i^{(j)}$.
4. **for** $k = 1$ to σ **do**
5. **if** $\Omega_i^{(j)}$ have the same input neurons with the elements in the subset $\tilde{\Omega}_i^k$
6. Add $\Omega_i^{(j)}$ into $\tilde{\Omega}_i^k$. // The structures with the same input neurons are collected in the same sub-subset.
7. **else**
8. $\sigma = \sigma + 1$.
9. Create a new empty subset $\tilde{\Omega}_i^{\sigma} = \{\ \}$, and add $\Omega_i^{(j)}$ into $\tilde{\Omega}_i^{\sigma}$.
10. **end**
11. **end**
12. **end**
13. Calculate the size η_i^k of each subset $\tilde{\Omega}_i^k (k = 1, 2, \ldots, \sigma)$ and record the maximum η_i^k as η_i^{max}.
14. $\upsilon_i^{max} = 0$, $\Delta_i^{best} = [\ ]$.
15. **for** $k = 1$ to σ **do**
16. **if** η_i^k is equal to η_i^{max}
17. Denote the corresponding subset $\tilde{\Omega}_i^k$ with $\tilde{\Omega}_i^{best}$ and subdivide it into a series of sub-subsets according to the hidden neurons/hidden layers. // The structures with the same hidden neurons/hidden layers are collected in the same sub-subset.
18. Determine the sub-subset with the maximum size υ_i^{tmax} and denote the corresponding structure of the sub-subset with Δ_i^{tbest}. // If more than one sub-subsets have the same maximum size, then choose one sub-subset randomly.
19. **if** $\upsilon_i^{tmax} > \upsilon_i^{max}$
20. $\Delta_i^{best} = \Delta_i^{tbest}$, $\upsilon_i^{max} = \upsilon_i^{tmax}$.
21. **else**
22. **continue**.
23. **end**
24. **else**
25. **continue**.
26. **end**
27. **end**
28. Obtain the optimal structure Δ_i^{best}.

3.2.3. Determining the combination weights using ITVPNNW

For combination weights, the simplest and most traditional statistical weight generation techniques that are solely based on historical forecast errors are considerably cited in the literature and present "hard to beat" results. However, the lack of an auxiliary model to update information along the forecast horizon may be considered a limitation. A direct alternative to traditional techniques considers a dynamic procedure where combination weights vary over the forecast horizon.

In practice, a dynamic generation scheme is not always guaranteed to outperform a static one, but "some time-variation or adaptive adjustment in the combination weights (or perhaps in the underlying models being combined) can often improve forecasting performance" [20]. However, developing an explicit model is a highly intricate task that does not guarantee performance increase. Therefore, the main objective of this section is to develop weighting models that can enhance traditional weighting procedures, relax historical forecast performance dependence, and abstract model complexity. Then, a weight generation framework called ITVPNNW is studied. ITVPNNW regards the in-sample set of weights as a new time series as pioneered by [29]. The inherent pattern of the new time series is then identified and learned through a BSA–BPNN model. The desired combination weights are then predicted from this fitted BSA–BPNN model.

The procedure for ITVPNNW can be divided into two phases. In the first phase, a set of in-sample weights is generated. For example, each component forecasting model with an optimal structure obtained in Section 3.2.2 is again trained on the consecutive in-sample training subsets and used to forecast their respective validation subsets. Some of the aforementioned absolute error measures are used to evaluate the forecasting efficiency of the model. A common consensus is that a component model with a larger error should receive less weight in a combination system. On the basis of this consensus, an *error-based* combination scheme assigns each weight as the normalized unbiased inverse absolute forecasting error of the model, such that

$$w_i = \frac{e_i^{-1}}{\sum_{i=1}^{n} e_i^{-1}}, \forall i = 1, 2, \cdots, n \tag{8}$$

In Eq. (8), e_i denotes the forecasting error of the ith model. In this study, the weight to the ith forecasting model for the jth training–validation pair is assigned based on an extended form of an *error-based* combination scheme shown in Eq. (9) as follows:

$$w_i^j = \frac{\exp(\nu_i^j)}{\sum_{i=1}^{n} \exp(\nu_i^j)}, \forall i = 1, 2, \cdots, n; j = 1, 2, \cdots, M \tag{9}$$

where $\nu_i^j = \left(\mathrm{MAE}_i^j + \mathrm{RMSE}_i^j + \mathrm{MAPE}_i^j\right)^{-1}$. The three terms, namely, MAE, RMSE, and MAPE, represent the respective errors of the ith model for the jth validation subset. The reasons for combining these three error measures have been mentioned in the previous paragraph. This assignment mechanism uses the SOFTMAX distribution, and these three error measures are inversely proportional to the combined effect. Evidently, each weight in a specific in-sample forecasting trial is non-negative and unbiased. An in-sample forecasting trial involves training the model and verifying its forecasting accuracy using an in-sample training–validation pair. Let $\mathbf{w}^j = \left[w_1^j, w_2^j, \ldots, w_n^j\right]^{\mathrm{T}}$ $(j = 1, 2, \ldots, M)$ be the column weight vector of the n models for the jth forecasting trial. X^{T} denotes the transposed form of X. A series of M weight vectors can be obtained after all the in-sample forecasting trials have been performed. Then, all the obtained weight vectors are concatenated with one another to form a $B \times 1$ $(B = n \times M)$ dimensional column vector $\mathbf{w} = \left[\left(\mathbf{w}^1\right)^{\mathrm{T}}, \left(\mathbf{w}^2\right)^{\mathrm{T}}, \ldots, \left(\mathbf{w}^M\right)^{\mathrm{T}}\right]^{\mathrm{T}}$, which can be regarded as a new time series of weights. Two examples of the new time series of weights are shown in Fig. 8. The vertical and horizontal axes repre-

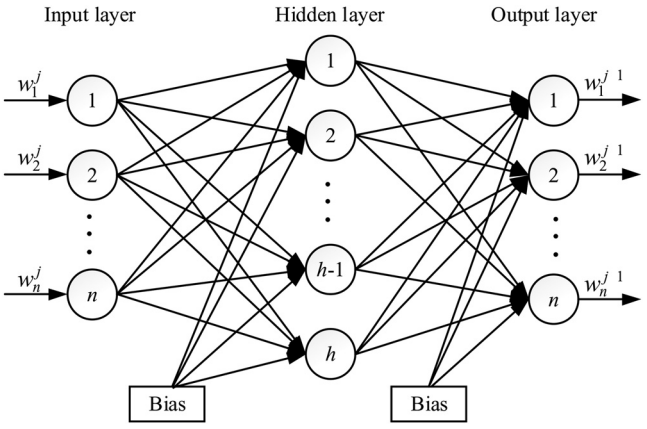

Fig. 9. Single hidden layer BPNN structure with the same input and output neurons.

sent the normalized weights w_i^j $(i = 1, 2, \ldots, n; j = 1, 2, \ldots, M)$ and their indices in the column weight vector $\mathbf{w}$, respectively. Notably, the weight time series exhibits regular recurring patterns. The primary reason for such recurrence is the *sliding window mechanism* used to formulate the consecutive pairs of training–validation subsets.

A neural network-based model, called BSA–BPNN, is proposed to identify and understand these regular recurring patterns in the second phase. The BSA–BPNN model caters to the following three points. (a) The characteristics of the whole data set can be approximately reflected by the training data set. (b) BPNNs are highly powerful and efficient in recognizing the recurring pattern in a data set, and this model has a simple architecture in which the numbers of input and output neurons are both equal to the number of component models. Moreover, this architecture only requires selecting the number of hidden neurons. (c) BSA is rapidly and successfully applied to solve the optimization problem, and it can be used to preliminarily search for the global optimal initial connection weights and the thresholds of BPNN. Thus, the designed BSA–BPNN model has an $n \times h \times n$ neural network structure, which is depicted in Fig. 9. The term h indicates the number of neurons in the hidden layer. $(M-1)$ input and output patterns, which are constituted by the weight vectors $\mathbf{w}^j = \left[w_1^j, w_2^j, \ldots, w_n^j\right]^{\mathrm{T}}$ and $\mathbf{w}^{j+1} = \left[w_1^{j+1}, w_2^{j+1}, \ldots, w_n^{j+1}\right]^{\mathrm{T}}$ $(j = 1, 2, \ldots, M-1)$, respectively, are used to train the BSA–BPNN model during the model fitting phase. Moreover, the last weight vector $\mathbf{w}^M$ is fed as the input to the fitted BSA–BPNN model to obtain the final predicted weights as $\mathbf{w} = [w_1, w_2, \ldots, w_n]^{\mathrm{T}}$. The pseudocode of the ITVPNNW process is presented in Algorithm 2.

Algorithm 2. Determining the combination weights of the models

Input: in-sample training–validation pairs $\left(\mathbf{Y}_{\mathrm{tr}}^{(j)}, \mathbf{Y}_{\mathrm{vd}}^{(j)}\right)(j = 1, 2, \ldots, M)$, validation subset size N_{vd}, $n(n = 4)$ identified forecasting models, and trail times τ
Output: predicted weight vector $\mathbf{w}$ for the forecasting models
1. $\mathbf{w} = [\ ]$. // Initialize $\mathbf{w}$ as the empty column vector
2. **for** $j = 1$ to M **do**
3. **for** $i = 1$ to n **do**
4. Fit $\mathbf{Y}_{\mathrm{tr}}^{(j)}$ into the ith individual model and use the fitted model to predict $\mathbf{Y}_{\mathrm{vd}}^{(j)}$ for τ times, and then calculate the average $\hat{\mathbf{Y}}_{\mathrm{vd}}^{(j)(i)}$.
5. Calculate the errors of $\lambda\left(\hat{\mathbf{Y}}_{\mathrm{vd}}^{(j)(i)}, \mathbf{Y}_{\mathrm{vd}}^{(j)}\right)$ using three error measures (MAE, RMSE, and MAPE) and then record the results as MAE_i^j, RMSE_i^j, and MAPE_i^j, respectively.
6. **end**
7. Compute the weights $\mathbf{w}^j = \left[w_1^j, w_2^j, \ldots, w_n^j\right]^{\mathrm{T}}$ using Eq. (9).
8. $\mathbf{w} = \left[\mathbf{w}^{\mathrm{T}}, \left(\mathbf{w}^j\right)^{\mathrm{T}}\right]^{\mathrm{T}}$ // Append $\mathbf{w}^j$ to $\mathbf{w}$
9. **end**
10. Fit an appropriate $n \times h \times n$ BPNN model into $\mathbf{w}(1 : n(M-1))$ and use BSA to optimize its connection weights and thresholds.
11. Provide $\mathbf{w}\left((n(M-1)+1) : nM\right) = \left[w_1^M, w_2^M, \ldots, w_n^M\right]^{\mathrm{T}}$ as the input to the fitted BSA–BPNN and obtain the predicted weight vector as $\mathbf{w} = \left[w_1, w_2, \ldots, w_n\right]^{\mathrm{T}}$.

Some features presented in this section are similar to those used in the literature; however, a new and different feature that extends

the previous work has been introduced in the literature [29]. In particular, the feature that concerned the training process of BPNN is included. This feature is facilitated by adopting a novel evolutionary computational technique, namely, BSA. In this manner, BSA can be applied to optimize the initial connection weights and the thresholds of BPNN, which will overcome the shortcoming of BPNN, i.e., the training results easily fall into the local minimum point instead of into the global optimum.

3.2.4. Combining forecasting models

The identified structure and the corresponding weight for each forecasting model are obtained after the aforementioned three phases. The training process of the proposed NNsLEF is completed and the fitted ensemble framework can be used for forecasting. The whole in-sample training data set $\mathbf{Y}_{tr}$ is fed into each identified forecasting model to forecast the out-of-sample data set $\mathbf{Y}_{ts}$, and the forecasted output is recorded as $\hat{Y}^{(i)}$. The final forecasted output $\hat{Y}$ can be obtained using Eq. (1) by combining the forecasted outputs $\hat{Y}^{(i)}\,(i = 1, 2, \ldots, n)$ with the weight vector $\mathbf{w}$. The pseudocode of NNsLEF is presented in Algorithm 3.

Algorithm 3. Process of the proposed linear ensemble framework

Input: training subset $\mathbf{Y}_{tr} = \left[y_1, y_2, \ldots, y_{N_{tr}}\right]^{T}$, testing subset size N_{ts}, $n(n=4)$ forecasting models (BPNN, DAN2, EANN, and ESN), and trail times τ
Output: combined forecast $\hat{Y} = \left[\hat{y}_{N_{tr}+1}, \hat{y}_{N_{tr}+2}, \ldots, \hat{y}_{N_{tr}+N_{ts}}\right]^{T}$
1. Algorithm 1. Confirm the structures of the forecasting models.
2. Algorithm 2. Obtain the predicted weights $\mathbf{w} = \left[w_1, w_2, \ldots, w_n\right]^{T}$.
1. **for** $i = 1$ to n **do**
2. Fit $\mathbf{Y}_{tr}$ into the ith model and use the fitted model to predict $\mathbf{Y}_{ts}$ for τ times, and then calculate the average predicted $\hat{Y}^{(i)} = \left[\hat{y}^{(i)}_{N_{tr}+1}, \hat{y}^{(i)}_{N_{tr}+2}, \ldots, \hat{y}^{(i)}_{N_{tr}+N_{ts}}\right]^{T}$.
3. **end**
4. Obtain the final combined forecast as $\hat{Y} = \sum_{i=1}^{n} w_i \hat{Y}^{(i)}$.

4. Empirical results and discussions

4.1. Experimental setup

4.1.1. Data sets

The proposed approach is empirically studied for the time series forecasting problem. The designed experiments are performed on eight data sets to demonstrate the effectiveness of the proposed NNsLEF. All the eight data sets are available at the *Time Seires Data Library* (*TSDL*) [69]. The detailed information of these eight data sets are (a) *Lynx* records the number of lynx trapped per year in Mackenzie River district of Northern Canada from 1821 to 1934; (b) *Sunspots* records annual visible spots observed on the face of the sun from 1700 to 1987; (c) *River flow* records the monthly flow in cms of the Clearwater river at Kamiah, Idaho, USA from 1911 to 1965; (d) *Vehicles* records monthly sales of vehicles in USA from January 1971 to December 1991; (e) *RGNP* records the real GNP values of USA from 1890 to 1974 in billions of dollars; (f) *Wine* records the monthly sales of red wine of Australian from January 1980 to July 1995 in thousands of liters; (g)*Airline* records the monthly number of international airline passengers from January 1949 to December 1960 in thousands; (h) Industry records quarterly industrial observations of USA from the first quarter of 1977 to the last quarter of 1992.

Table 1
Description of the eight data sets.

Time series	Type	Total size	Training size	Testing size
Lynx	Stationary, nonseasonal	114	100	14
Sunspots	Stationary, nonseasonal	288	253	35
River flow	Stationary, nonseasonal	600	500	100
Vehicles	Nonstationary, nonseasonal	252	200	52
RGNP	Nonstationary, nonseasonal	85	70	15
Wine	Monthly seasonal	187	132	55
Airline	Monthly seasonal	144	132	12
Industry	Quarterly seasonal	64	48	16

These data sets are the same as those used in the literature [29] to ensure effective comparison. Table 1 presents the basic information of these data sets. The first column lists the names of all the data sets. The second column presents the characteristic of each data set. The total size, training size, and testing size of each data set are provided in the third, fourth, and fifth columns, respectively. All the experiments are implemented using MATLAB 2014a and performed on a personal computer with Intel Core 4 GHz CPU and 8 GB main memory.

4.1.2. Data preprocessing

A data preprocessing process is performed for each data set to improve the performance of the proposed approach at the beginning of the experiments. Each data set is linearly scaled within the range [0, 1] by adopting linear transformation through Eq. (10):

$$x = \frac{x - x_{\min}}{x_{\max} - x_{min}}, \tag{10}$$

where x_{max} and x_{min} are the maximum and minimum values of the data set, respectively. Among all the data sets, "Lynx" is preprocessed using logarithms (to the base of 10) to achieve a good fit [29], and then the data preprocessing process is applied. This process is implemented to avoid the phenomenon in which the attributes in the wide numeric ranges dominate those in the small numeric ranges.

4.1.3. Parameter setting tuning test

The performance of the proposed approach primarily depends on the performances of the suggested ANN models, which are influ-

Table 2
Requisite modeling information.

Data sets	BPNN	DAN2	EANN	ESN	h
Lynx	$3 \times 9 \times 1$	$8 \times 0 \times 1$	$3 \times 16 \times 1$	$3 \times 30 \times 1$	14
Sunspots	$8 \times 12 \times 1$	$8 \times 0 \times 1$	$8 \times 16 \times 1$	$3 \times 40 \times 1$	13
River flow	$8 \times 9 \times 1$	$3 \times 1 \times 1$	$8 \times 9 \times 1$	$8 \times 80 \times 1$	16
Vehicles	$3 \times 11 \times 1$	$7 \times 2 \times 1$	$3 \times 9 \times 1$	$3 \times 30 \times 1$	12
RGNP	$5 \times 16 \times 1$	$4 \times 0 \times 1$	$3 \times 15 \times 1$	$8 \times 60 \times 1$	12
Wine	$8 \times 8 \times 1$	$3 \times 1 \times 1$	$8 \times 11 \times 1$	$8 \times 30 \times 1$	12
Airline	$11 \times 12 \times 1$	$12 \times 3 \times 1$	$11 \times 12 \times 1$	$11 \times 90 \times 1$	17
Industry	$4 \times 17 \times 1$	$4 \times 0 \times 1$	$4 \times 16 \times 1$	$11 \times 60 \times 1$	13

enced by the selection of their structures and selected according to a series of experiments. In general, all the four neural network models are designed with one neuron in the output layer in this study. More details about specific setting process of each model are as follows. The BPNN and EANN models are designed with only one hidden layer and are fitted into the neural network toolbox through the training function *trainlm* [70]. The numbers of input and hidden neurons for each individual BPNN model are selected from the set $\{3, 4, \ldots, 12\}$ and the set $\{12, 13, \ldots, 24\}$, respectively. The selections of input and hidden neurons for the EANN models are the same as those for the BPNN models. The number of input nodes is also selected from the set $\{3, 4, \ldots, 12\}$ for each individual DAN2 model and ESN model; however, the selection of hidden structures for the DAN2 and ESN models is relatively different from that for the BPNN and EANN models. In particular, the DAN2 models are designed to have less than five hidden layers and the number of hidden neurons for each hidden layer is fixed to four. The number of hidden neurons (i.e., the reservoir size) is selected from the set $\{30, 40, \ldots, 120\}$ for each individual ESN model. Thus, a total of $\kappa_1 = 100$ BPNN models, $\kappa_2 = 100$ EANN models, $\kappa_4 = 100$ ESN models, and a maximum of $\kappa_3 = 50$ DAN2 models have been investigated for each in-sample training–validation pair of each time series. For the BSA–BPNN model, the numbers of input and output neurons are both equal to the number of component models; thus, only the number of hidden neurons h is required to be determined. h is selected from the set $\{12, 13, \ldots, 24\}$, and the optimal value is chosen based on the experiments. Based on the IHSH introduced in Section 3.2.2, the requisite modeling information for all the eight time series data sets is obtained and presented in Table 2, in which $p \times q \times 1$ for DAN2 indicates that q hidden layers exist in the model.

Other parameters also play important roles in the performance of the proposed approach, such as parameters M used for the training phase, N_{vd} and τ used for the entire phase, parameters involved in ESN, and parameters used in BSA. An extremely large value of M may lead the training process be sensitive to the outliers and increases the size of the in-sample weight set as well as increases the time complexity of the algorithm; however, a small value of Mmay lead to a biased combination of the most recent in-sample observations, and thus, may be inadequate. The determination of M is based on a series of experiments. Different values of M have been tested based on the formula $M = \lceil coeff \times N_{tr} \rceil$, where *coeff* is a coefficient ranging from 0.1 to 0.3, and N_{tr} is the number of observations in the training data set $\mathbf{Y}_{tr}$ of each data set. $\lceil C \rceil$ represents the rounding of C to the nearest integer higher than or equal to C. The experimental results suggest that $M = \lceil 0.2 \times N_{tr} \rceil$ is an appropriate choice for nearly all the data sets. Another parameter is N_{vd}, which is considered equal to the number of observations in the out-of-sample testing data set for each data set, i.e., $N_{vd} = N_{ts}$. Meanwhile, the parameter trial times τ is set to 5. The two parameter N_{vd} and τ are not so important to affect the experimental results, and thus are empirically set.

ESN involves the selection of reservoir parameters, such as the number of input nodes, reservoir size (*SN*), sparseness (*SP*), and spectral radius (*SR*) of the internal connection matrix *W*. The sug-

Table 3
Best forecasting results for different models.

Data sets	Error measures	Adhikari's models									Our proposed models						
		Individual models				Combination methods					Individual models				Combination methods		
		Box-Jenkins	SVM	FANN	EANN	Avg.	Median	EB	LSR	Outperf.	AIW	*AsM*	BPNN	DAN2	EANN	ESN	NNsLEF
Lynx	MAE	0.103	0.173	0.154	0.156	0.112	0.133	0.097	0.133	0.107	0.110	0.068	0.091	0.125	0.086	0.095	0.067
	MSE	0.015	0.053	0.032	0.036	0.018	0.026	0.013	0.027	0.015	0.017	0.006	0.013	0.024	0.012	0.016	0.006
Sunspots	MAE	14.91	20.06	16.55	18.93	15.96	14.98	13.78	14.2	13.75	13.84	13.49	13.242	16.997	15.1	13.106	**11.451**
	MSE	348.5	630.3	526.4	663.6	371.8	428.4	352.3	393.4	375.4	371.8	311	305.354	509.055	401.057	302.93	**210.485**
River flow	MAE	1.26	0.687	0.66	1.036	0.751	0.676	0.712	0.736	0.665	0.749	**0.638**	0.654	1.54	0.834	0.684	0.723
	MSE	2.606	1.172	1.217	2.19	1.158	1.138	1.197	1.245	1.068	1.156	0.978	1.001	5.19	1.688	1.107	**1.372**
Vehicles	MAE	2.174	2.239	2.128	2.173	2.087	2.139	2.06	2.059	2.011	2.071	2.001	2.099	2.073	2.177	1.955	**1.878**
	MSE	7.142	6.794	7.018	6.989	6.188	6.683	6.011	7.508	6.088	6.175	5.531	6.428	6.407	6.89	6.191	**5.397**
RGNP	MAE	17.01	20.89	16.2	15	11.74	13.67	10.1	12.25	13.23	11.72	9.903	16.873	11.747	14.513	29.46	**9.567**
	MSE	470	719.2	359.4	322.2	197.3	265.9	155.2	278.5	260.6	196.6	139	678.162	206.696	315.293	1133	**131.085**
Wine	MAE	2.776	2.451	2.923	3.009	2.075	2.173	2.057	2.466	2.008	2.372	**1.923**	2.766	2.954	2.893	2.737	2.303
	MSE	12.19	10.03	17	18.2	9.233	10.05	9.014	10.07	7.712	9.204	7.524	16.201	15.859	13.106	12.185	**9.113**
Airline	MAE	12.49	10.85	16.17	15.24	11.63	11.73	10.85	10.68	10.85	10.22	7.434	14.083	12.5	14.583	12.167	**7**
	MSE	291	176.9	378	332.5	181.4	176.5	157.8	158.3	152.5	143.1	86.63	244.417	260.667	264.25	230.5	**64.833**
Industry	MAE	1.526	1.958	1.388	1.822	1.386	1.37	1.678	1.365	1.452	1.386	1.272	1.419	1.391	1.335	1.667	**1.011**
	MSE	4.842	5.532	3.162	4.606	2.888	2.869	5.239	2.974	3.764	2.892	2.576	2.355	2.624	2.266	3.186	**1.302**

gested preparation of the reservoir is presented in the study of Jaeger [62]. In the current simulation, the number of input nodes and reservoir size (*SN*) are set as mentioned earlier. All the other parameters are set according to the recommendation of existing literature. According to the study [7], the sparseness (*SP*) of *W* is selected as $10/SN$ and the spectral radius (*SR*) is set to 0.8. Connection weights matrices *W* and W^{in}, as well as the noise matrix *bias*, are randomly fixed and unchanged. The range of elements in *W*, W^{in}, and *bias* is specified within [0, 1][62]. The parameters used in BSA are set based on a series of experiments and recommendation of existing literature. From the experimental results, the BSA parameters are specified as follows: (a) the population size is set to 100, (b) the maximum iteration is set to 100, and (c) the control parameter *mixrate* is set to 1.0 [58]. In summary, all the parameters used in this study can be determined according to the experiments, the recommendations of existing literature and empirical setting.

4.2. Analysis of results

In this work, the performance of the proposed approach is compared with those of different forecasting models, including the four proposed component models and Adhikari's models [29]. Note that Adhikari's models include the four individual models (Box-Jenkins, SVM, FANN and EANN) and seven combination models (Avg., Median, EB, LSR, Outperf., AIW and the linear combination model). Among the seven combination models, the linear combination model, denoted as *AsM*, is the best-known model. For more details, we refer to the reference [29]. MAE and MSE error measures are selected to evaluate forecasting accuracies, as used in Adhikari's models [29], to provide a fair idea on the performances of the different models.

Table 3 presents the obtained forecasting results of the proposed models and Adhikari's models, where the best results are shown in boldface. In particular, the results of the "River flow," "Wine," and "Industry" time series are given in transformed scales, such that the original MAE is equal to the obtained MAE multiplied by 10^2 and the original MSE is equal to the obtained MSE multiplied by 10^4. A diagrammatic depiction of the forecasting performance of the proposed models on the eight data sets is provided in Fig. 10. The actual and forecasted values obtained from the BPNN, DAN2, EANN, ESN, and NNsLEF models are represented in each plot.

The mean ratio of relative errors is used as a measure across data sets to have an idea regarding relative performance [7,71]. For the two methods, *A* (base method) and *B* (comparison method) with errors δ_A and δ_B, the relative error is $\frac{\delta_B-\delta_A}{\delta_A}$. A relative error is a popular measure that indicates superior performance between two compared methods. The lower the relative error, the better the performance of the comparison method than that of the base method.

In this study, the proposed NNsLEF model is set as the comparison method, whereas the proposed four individual models, and Adhikari's four individual models, as well as Adhikari's seven combination models are set as the base methods. Tables 4 and 5 show the relative errors of the pair of models where NNsLEF is set as comparison method and an individual model or a combination method is set as base method. The mean ratio of the relative error is the average of the relative errors over all the data sets. A value of the mean ratio of the relative error that is less than 0 represents an improvement relative to the base method.

From Tables 3–5, the performance of the proposed NNsLEF model is relatively superior to those of the four proposed component models (BPNN, DAN2, EANN, and ESN), Adhikari's four individual models (Box-Jenkins, SVM, FANN and EANN), and Adhikari's seven combination models (Avg., Median, EB, LSR, Outperf., AIW and *AsM*), as evidenced by examining either the cases alone or the overall cases.

(a) Improvement in the forecasting performance occurs when the NNsLEF model is compared with the eight individual models, including Box-Jenkins, SVM, FANN, Adhikari's EANN, BPNN, DAN2, the proposed EANN and ESN. Examining the cases alone, the NNsLEF model is superior to the eight individual models in cases with eight, seven, seven, eight, seven, eight, eight, and seven data sets in term of MSE error measure. Hence, the NNsLEF model can achieve better forecasting accuracy than the eight component models for nearly all of the eight data sets. For the "River flow" data set, the performance of the NNsLEF model is relatively unsatisfactory compared with the individual models of SVM, FANN, BPNN and ESN. This information is shown in italic in Table 4. The reason for this unsatisfactory result is multifaceted. One reason for this outcome may be attributed to the limited computational complexity and running time given that the notable characteristic of this data set is the large number of samples.

From another aspect, i.e., examining the overall cases, the NNsLEF model can also obtain better forecasting accuracy. The mean ratios of the relative error, in terms of both MAE and MSE error measures, are −31.61% and −52.45% for the Box-Jenkins and NNsLEF pair, −32.4% and −48.69% for the SVM and NNsLEF pair, −29.44% and −50.4% for the FANN and NNsLEF pair, −37.33% and −59.15% for the Adhikari's EANN and NNsLEF pair, −22.37% and −38.31% for the BPNN and NNsLEF pair, −31.68% and −53.45% for the DAN2 and NNsLEF pair, −25.51% and −43.1% for the EANN and NNsLEF pair, and −25.69% and −40.82% for the ESN and NNsLEF pair, respectively. This information is shown as "MEAN" in the last row of Table 4. The results that all the mean ratios of the relative error, in terms of both MAE and MSE error measures, are less than 0 have indicated that the NNsLEF model has an improvement relative to the eight individual models.

(b) Improvement in the forecasting performance is observed when the NNsLEF model is compared with Adhikari's seven combination models, including Avg., Median, EB, LSR, Outperf., AIW and *AsM*. The proposed model also outperforms the seven combination models in two aspects. For the cases alone, the NNsLEF model can obtain better forecasting accuracy in the cases with seven, six, six, eight, six, eight and six of the eight data sets in term of MSE error measure. In particular, the NNsLEF model clearly does not perform well in the "River flow" and "Wine" data sets. This information is shown in italic in Table 5. More specifically, for the "River flow" data set, the performance of the NNsLEF model is relatively unsatisfactory compared with the combination models of Median, EB, Outperf., and *AsM*. The reason for this unsatisfactory outcome may be attributed to the results generated from the proposed model do not satisfy Eq. (2), which may lead to the unsatisfactory performance. As for the "Wine" data set, the NNsLEF clearly does not perform well when compared with the combination models of Avg., Median, EB, and *AsM*. It may be because of the limited performance of the NNsLEF model. After all, a model cannot produce good results for all data sets.

Although the NNsLEF model generates unsatisfactory results in the "River flow" and "Wine" data sets, it does not indicate that the performance of the NNsLEF model is poor. Evidence can be obtained when attention is directed toward the performance of the overall cases. The reduction values of the proposed NNsLEF model, in term of both MAE and MSE error measures, are 19.57% and 32.3%, 21.12% and 38.05%, 15.46% and 29.77%, 21.06% and 39.99%, 16.36 and 30.16%, 18.62% and 30.75%, and 2.42% and 6.71% when com-

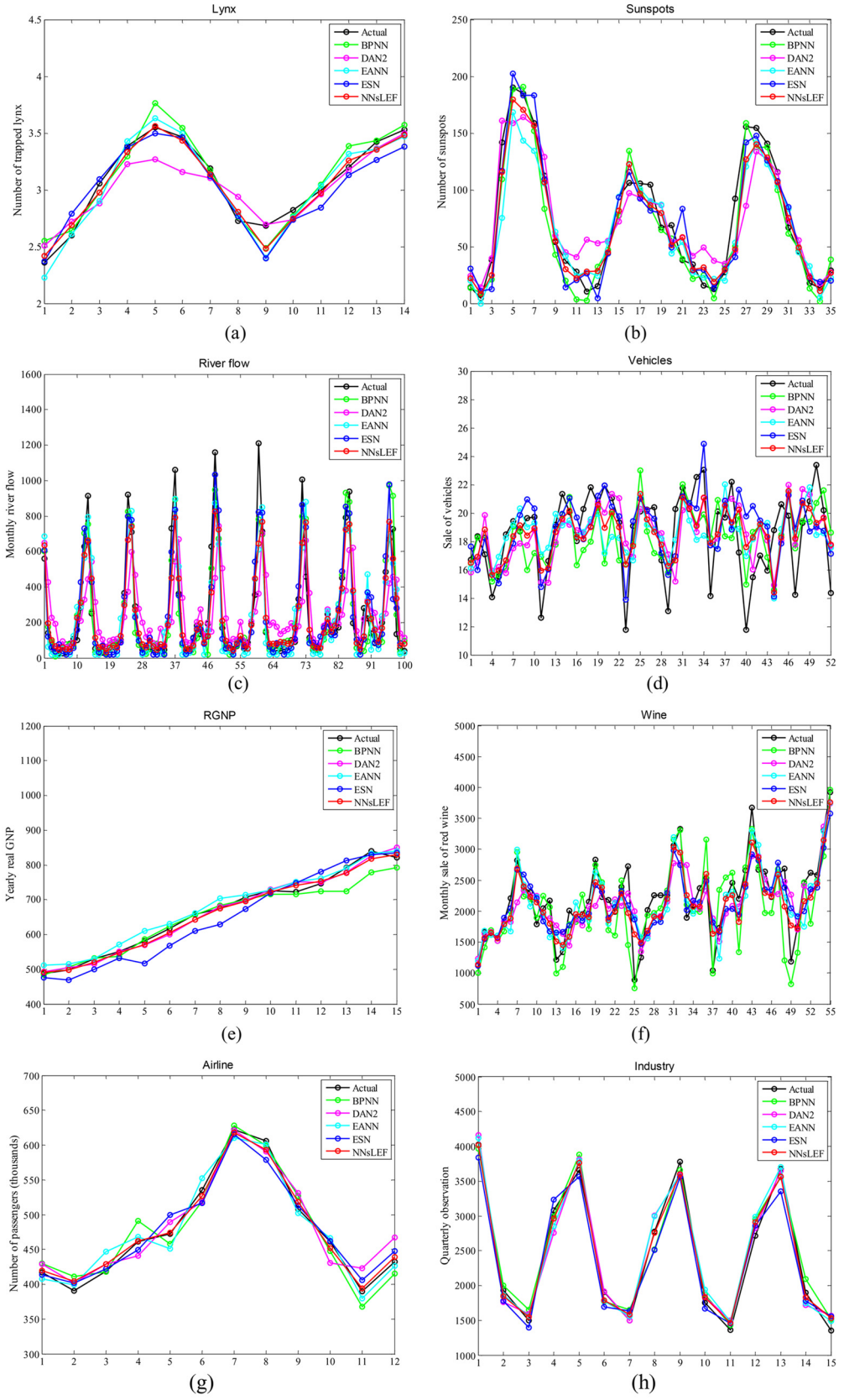

Fig. 10. Forecasting performance of the eight data sets: (a) Lynx, (b) Sunspots, (c) River flow, (d) Vehicles, (e) RGNP, (f) Wine, (g) Airline, and (h) Industry.

pared with the models Avg., Median, EB, LSR, Outperf., AIW and *AsM*, respectively. Note that the performance of the NNsLEF model is better than that of the best-known model *AsM* and has MAE and MSE improvements of 2.42% and 6.71%. This information is shown in boldface as "MEAN" in the last row of Table 5.

Table 4
Relative error rates of different models (NNsLEF is set as comparison method and the eight individual models are respectively set as base method).

Data sets	Error measures	Adhikari's models				Our proposed models			
		Box-Jenkins	SVM	FANN	EANN	BPNN	DAN2	EANN	ESN
Lynx	MAE	-0.3495	-0.6127	-0.5649	-0.5705	-0.2637	-0.464	-0.2209	-0.2947
	MSE	-0.6	-0.8868	-0.8125	-0.8333	-0.5385	-0.75	-0.5	-0.625
Sunspots	MAE	-0.232	-0.4292	-0.3081	-0.3951	-0.1353	-0.3263	-0.2417	-0.1263
	MSE	-0.396	-0.666	-0.6	-0.6828	-0.3107	-0.5865	-0.4752	-0.3052
River flow	MAE	-0.4262	*0.0524*	*0.0955*	-0.3021	*0.1055*	-0.5305	-0.1331	*0.0570*
	MSE	-0.4735	*0.1706*	*0.1274*	-0.3735	*0.3706*	-0.7356	-0.1872	*0.2394*
Vehicles	MAE	-0.1362	-0.1612	-0.1175	-0.1358	-0.1053	-0.0941	-0.1373	-0.0394
	MSE	-0.2443	-0.2056	-0.231	-0.2278	-0.1604	-0.1576	-0.2167	-0.1283
RGNP	MAE	-0.4376	-0.542	-0.4094	-0.3622	-0.433	-0.1856	-0.3408	-0.6753
	MSE	-0.7211	-0.8177	-0.6353	-0.5932	-0.8067	-0.3658	-0.5842	-0.8843
Wine	MAE	-0.1704	-0.0604	-0.2121	-0.2346	-0.1674	-0.2204	-0.2039	-0.1586
	MSE	-0.2524	-0.0914	-0.4639	-0.4993	-0.4375	-0.4254	-0.3047	-0.2521
Airline	MAE	-0.4396	-0.3548	-0.5671	-0.5407	-0.5029	-0.4400	-0.5200	-0.4247
	MSE	-0.7772	-0.6335	-0.8285	-0.805	-0.7347	-0.7513	-0.7547	-0.7187
Industry	MAE	-0.3375	-0.4837	-0.2716	-0.4451	-0.2875	-0.2732	-0.2427	-0.3935
	MSE	-0.7311	-0.7646	-0.5882	-0.7173	-0.4471	-0.5038	-0.4254	-0.5913
MEAN	MAE	-0.3161	-0.324	-0.2944	-0.3733	-0.2237	-0.3168	-0.2551	-0.2569
	MSE	-0.5245	-0.4869	-0.504	-0.5915	-0.3831	-0.5345	-0.431	-0.4082

Table 5
Relative error rates of different models (NNsLEF is set as comparison method and the seven combination models are respectively set as base method).

Data sets	Error measures	Adhikari's models						
		Avg.	Median	EB	LSR	Outperf.	AIW	*AsM*
Lynx	MAE	-0.4017	-0.4962	-0.3093	-0.4962	-0.3738	-0.3909	-0.0147
	MSE	-0.6667	-0.7692	-0.5385	-0.7778	-0.6	-0.6471	0
Sunspots	MAE	-0.2825	-0.2356	-0.169	-0.1936	-0.1672	-0.1723	-0.1511
	MSE	-0.4339	-0.5087	-0.4025	-0.465	-0.4393	-0.4339	-0.3232
River flow	MAE	-0.0373	*0.0695*	*0.0155*	-0.0177	*0.0872*	-0.0347	*0.1332*
	MSE	0.1848	0.2056	0.1462	0.102	0.2846	0.1869	0.4029
Vehicles	MAE	-0.1001	-0.122	-0.0884	-0.0879	-0.0661	-0.0932	-0.0615
	MSE	-0.1278	-0.1924	-0.1021	-0.2812	-0.1135	-0.126	-0.0242
RGNP	MAE	-0.1851	-0.3001	-0.0528	-0.219	-0.2769	-0.1837	-0.0339
	MSE	-0.3356	-0.507	-0.1554	-0.5293	-0.497	-0.3332	-0.0569
Wine	MAE	*0.1099*	*0.0598*	*0.1196*	-0.0661	*0.1469*	-0.0291	*0.1976*
	MSE	-0.013	-0.0932	0.011	-0.095	0.1817	-0.0099	0.2112
Airline	MAE	-0.3981	-0.4032	-0.3548	-0.3446	-0.3548	-0.3151	-0.0584
	MSE	-0.6426	-0.6327	-0.5891	-0.5904	-0.5749	-0.5469	-0.2516
Industry	MAE	-0.2706	-0.262	-0.3975	-0.2593	-0.3037	-0.2706	-0.2052
	MSE	-0.5492	-0.5462	-0.7515	-0.5622	-0.6541	-0.5498	-0.4946
MEAN	MAE	-0.1957	-0.2112	-0.1546	-0.2106	-0.1636	-0.1862	-0.0243
	MSE	-0.323	-0.3805	-0.2977	-0.3999	-0.3016	-0.3075	-0.0671

Table 6
Wilcoxon signed-rank test.

Compared methods	R^+	R^-	*p*-value	Diff?
NNsLEF versus BPNN	34	2	0.025	Yes
NNsLEF versus DAN2	36	0	0.012	Yes
NNsLEF versus EANN	36	0	0.012	Yes
NNsLEF versus ESN	34	2	0.025	Yes
NNsLEF versus *AsM*	28	8	0.161	No

Nonparametric statistical tests are performed on the experimental results using the MAE error measure alone to further test whether significant performance differences exist between the proposed NNsLEF model and the four component models (BPNN, DAN2, EANN, and ESN) and the reference method (*AsM*). The Friedman test shows that significant differences exist among the compared methods with a *p*-value of 0.001 (the *p*-value is less than 0.05). Furthermore, the Wilcoxon signed-rank test outcomes shown in Table 6 indicate that the NNsLEF model is statistically better than the BPNN, DAN2, EANN, and ESN models with respective *p*-values of 0.025, 0.012, 0.012, and 0.025 and is insignificantly better than the *AsM* model with a *p*-value of 0.161.

In summary, the performance of the proposed model is superior to those of the proposed four individual models (BPNN, DAN2, EANN, and ESN), and Adhikari's four individual models, as well as Adhikari's seven combination models by examining either the cases alone or the overall performance.

5. Conclusions and future work

This study proposes a new and effective linear ensemble framework, called NNsLEF, which aims to improve the accuracy of time series forecasting. The proposed NNsLEF could be an effective forecast combination model because of the following reasons: (a) four widely used neural networks (BPNN, DAN2, EANN, and ESN) are selected as the underlying component forecasts, and an IHSH mechanism is accordingly designed to determine their structures. Neural networks are flexible nonlinear data-driven models with attractive properties for forecasting. Thus, NNsLEF could inherit its forecasting performance from the four component neural network models. (b) a dynamic weight combination approach, namely, ITVPNNW, is proposed to generate the relative optimal combination weights. The main advantage of ITVPNNW is the capability to relax in-sample performance dependence and abstract statistical complexity. It can improve forecasting accuracy of forecast combination model. Thus, NNsLEF can merge the advantages of component neural networks and a dynamic weight combination

approach for better performance. This merging is the main contribution of this work.

A series of experiments has been performed to verify the effectiveness of the proposed forecast combination. Under the criterion of minimizing the sum of absolute errors in terms of MAE and MSE, the experimental results obtain the following main conclusions. (a) The proposed forecast combination approach is superior to the four component models (BPNN, DAN2, EANN, and ESN) in cases with seven, eight, eight, and seven data sets, with MAE and MSE presenting the best/worst average improvements of 31.68%/22.37% and 53.54%/38.31%, respectively. (b) The proposed approach is more promising compared with the best-known *AsM* combination method, as reflected in better forecasting accuracies for three-fourths of the eight data sets, with average improvements of 2.42% for MAE and 6.71% for MSE for all the data sets. (c) The proposed approach adapts well to different data set types, including stationary/nonstationary and seasonal/nonseasonal, without showing evident signs of overfitting. Thus, the proposed NNsLEF model is a potential tool for addressing complex and interesting time series forecasting problems in a variety of fields.

However, the approach is limited in terms of computational complexity, particularly when data sets with a large number of samples are used to test the proposed proposal. These underlying drawbacks would limit the approach to the cases where additional training time is of low relevance for the concerned application. Therefore, future research can involve reducing the computational complexity and running time of NNsLEF to address these limitations. Other extensions may comprise (a) combining different forecasting models, (b) analyzing other input-hidden neuron determination mechanisms, and (c) applying new combination weight-generating mechanisms such as Bayes combination and genetic optimization. Efforts should also be exerted to develop novel combination approaches that can be easily implemented with satisfactory performance under complex decision environments [72–76].

Acknowledgments

The authors are very grateful for the constructive comments of editors and referees. This research is partially supported by the National Natural Science Foundation of China (71771095; 71701064).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.asoc.2018.02.004.

References

[1] A. Lempel, J. Ziv, On the complexity of finite sequences, IEEE Transactions on Information Theory 22 (1) (1976) 75–81.
[2] J.W. Kantelhardt, S.A. Zschiegner, E. Koscielny-Bunde, S. Havlin, A. Bunde, H.E. Stanley, Multifractal detrended fluctuation analysis of nonstationary time series, Physica A 316 (2002) 87–114.
[3] Y. Lai, N. Ye, Recent developments in chaotic time series analysis, International Journal of Bifurcation & Chaos 13 (06) (2003) 1383–1422.
[4] J. Zhang, X. Luo, T. Nakamura, J. Sun, M. Small, Detecting temporal and spatial correlations in pseudoperiodic time series, Physical Review E Statistical Nonlinear & Soft Matter Physics 75 (2) (2007) 016218.
[5] A. Cherif, H. Cardot, R. Boné, SOM time series clustering and prediction with recurrent neural networks, Neurocomputing 74 (11) (2011) 1936–1944.
[6] Y. Wan, X. Gong, Y. Si, Effect of segmentation on financial time series pattern matching, Applied Soft Computing 38 (2016) 346–359.
[7] L. Wang, Z. Wang, S. Liu, An effective multivariate time series classification approach using echo state network and adaptive differential evolution algorithm, Expert Systems with Applications 43 (2016) 237–249.
[8] Y.C. Hu, Electricity consumption prediction using a neural-network-based grey forecasting approach, Journal of the Operational Research Society 68 (10) (2017) 1259–1264.
[9] S. Ben Taieb, G. Bontempi, A.F. Atiya, A. Sorjamaa, A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition, Expert Systems with Applications 39 (8) (2012) 7067–7083.
[10] H.Z. Li, S. Guo, C.J. Li, J.Q. Sun, A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm, Knowledge-Based Systems 37 (2013) 378–387.
[11] Y. Bodyanskiy, S. Popov, Neural network approach to forecasting of quasiperiodic financial time series, European Journal of Operational Research 175 (3) (2006) 1357–1366.
[12] G. Li, H. Song, Tourism demand modelling and forecasting-A review of recent research, Tourism Management 29 (2) (2008) 203–220.
[13] C.F. Chen, M.C. Lai, C.C. Yeh, Forecasting tourism demand based on empirical mode decomposition and neural network, Knowledge-Based Systems 26 (2012) 281–287.
[14] X. Lu, J. Wang, Y. Cai, J. Zhao, Distributed HS-ARTMAP and its forecasting model for electricity load, Applied Soft Computing 32 (2015) 13–22.
[15] Y.R. Zeng, Y. Zeng, B. Choi, L. Wang, Multifactor-influenced energy consumption forecasting using enhanced back-propagation neural network, Energy 127 (2017) 381–396.
[16] J.M. Bates, C.W.J. Granger, The combination of forecasts, Operational Research Quarterly 20 (4) (1969) 451–468.
[17] J.A. Hoeting, D. Madigan, A.E. Raftery, C.T. Volinsky, Bayesian model averaging: a tutorial, Statistical Science 14 (4) (1999) 382–417.
[18] R.T. Clemen, Combining forecasts: a review and annotated bibliography, International Journal of Forecasting 5 (1989) 559–583.
[19] L.M. de Menezes, D.W. Bunn, J.W. Taylor, Review of guidelines for the use of combined forecasts, European Journal of Operational Research 120 (2000) 190–204.
[20] A. Timmermann, Forecast combinations, in: G. Elliott, C.W.J. Granger, A. Timmermann (Eds.), Handbook of Economic Forecasting, North-Holland, San Diego, 2006.
[21] N. Kourentzes, D.K. Barrow, S.F. Crone, Neural network ensemble operators for time series forecasting, Expert Systems with Applications 41 (9) (2014) 4235–4244.
[22] S.F. Crone, M. Hibon, K. Nikolopoulos, Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction, International Journal of Forecasting 27 (3) (2011) 635–660.
[23] G.P. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with artificial neural networks: The state of the art, International Journal of Forecasting 14 (1) (1998) 35–62.
[24] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[25] M. Khashei, M. Bijari, A novel hybridization of artificial neural networks and ARIMA models for time series forecasting, Applied Soft Computing 11 (2) (2011) 2664–2675.
[26] A. Jain, A.M. Kumar, Hybrid neural network models for hydrologic time series forecasting, Applied Soft Computing 7 (2) (2007) 585–592.
[27] T. Hill, M. O'Connor, M. Remus, Neural network models for time series forecasting, Management Science 42 (7) (1996) 1082–1092.
[28] I.M. Galvan, P. Isasi, Multi-step learning rule for recurrent neural models: an application to time series forecasting, Neural Processing Letters 13 (2) (2001) 115–133.
[29] R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing 157 (2015) 231–242.
[30] L. Wang, Y. Zeng, T. Chen, Back propagation neural network with adaptive differential evolution algorithm for time series forecasting, Expert Systems with Applications 42 (2) (2015) 855–863.
[31] C.H. Aladag, A new architecture selection method based on tabu search for artificial neural networks, Expert Systems with Applications 38 (4) (2011) 3287–3293.
[32] S. Aras, I.D. Kocakoç, A new model selection strategy in time series forecasting with artificial neural networks: IHTS, Neurocomputing 174 (2016) 974–987.
[33] G. Lachtermacher, J.D. Fuller, Backpropagation in time-series forecasting, Journal of Forecasting 14 (4) (1995) 381–393.
[34] C.H. Aladag, E. Egrioglu, S. Gunay, M.A. Basaran, Improving weighted information criterion by using optimization, Journal of Computational and Applied Mathematics 233 (10) (2010) 2683–2687.
[35] U. Anders, O. Korn, Model selection in neural networks, Neural Networks 12 (2) (1999) 309–323.
[36] E. Egrioglu, U. Yolcu, C.H. Aladag, E. Bas, Recurrent multiplicative neuron model artificial neural network for non-linear time series forecasting, Neural Processing Letters 41 (2) (2015) 249–258.
[37] S. Heravi, D.R. Osborn, C.R. Birchenhall, Linear versus neural network forecasting for european industrial production series, International Journal of Forecasting 20 (3) (2004) 435–446.
[38] V.L.M. Martins, L. Werner, Forecast combination in industrial series: a comparison between individual forecasts and its combinations with and without correlated errors, Expert Systems with Applications 39 (13) (2012) 11479–11486.
[39] T.E. Clark, M.W. McCracken, Improving forecast accuracy by combining recursive and rolling forecasts, International Economic Review 50 (2) (2009) 363–395.
[40] S. Makridakis, R.L. Winkler, Averages of forecasts: Some empirical results, Management Science 29 (9) (1983) 987–996.
[41] C.E. Agnew, Bayesian consensus forecasts of macroeconomic variables, Journal of Forecasting 4 (4) (1985) 363–376.

[42] V.R.R. Jose, R.L. Winkler, Simple robust averages of forecasts: Some empirical results, International Journal of Forecasting 24 (2008) 163–169.
[43] N. Kourentzes, F. Petropoulos, J.R. Trapero, Improving forecasting by estimating time series structural components across multiple frequencies, International Journal of Forecasting 30 (2) (2014) 291–302.
[44] S. Yin, L. Liu, J. Hou, A multivariate statistical combination forecasting method for product quality evaluation, Information Sciences 355 (2016) 229–236.
[45] S.G. Hall, J. Mitchell, Combining density forecasts, International Journal of Forecasting 23 (2007) 1–13.
[46] L.L. Pauwels, A.L. Vasnev, A note on the estimation of optimal weights for density forecast combinations, International Journal of Forecasting 32 (2) (2016) 391–397.
[47] R.D.O. Valle Dos Santos, M.M.B.R. Vellasco, Neural Expert Weighting: A NEW framework for dynamic forecast combination, Expert Systems with Applications 42 (22) (2015) 8625–8636.
[48] R.R. Andrawis, A.F. Atiya, H. El-Shishiny, Forecast combinations of computational intelligence and linear models for the NN5 time series forecasting competition, International Journal of Forecasting 27 (3) (2011) 672–688.
[49] J.S. Armstrong, Principles of Forecasting: A Handbook for Researchers and Practitioners, vol. 30, Kluwer Academic Publishers, Boston, USA, 2001.
[50] C. Lemke, B. Gabrys, Meta-learning for time series forecasting and forecast combination, Neurocomputing 73 (10) (2010) 2006–2016.
[51] C.C. Lin, C.L. Lin, J.Z. Shyu, Hybrid multi-model forecasting system: A case study on display market, Knowledge-Based Systems 71 (2014) 279–289.
[52] Z.H. Zhou, J. Wu, W. Tang, Ensembling neural networks: Many could be better than all, Artificial Intelligence 137 (1) (2002) 239–263.
[53] D.E. Rumelhart, J.L. McClelland, Parallel distributed processing – explorations in the microstructure of cognition, MIT Press, Cambridge, MA, 1986.
[54] G.P. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing 50 (2003) 159–175.
[55] L. Wang, Y.R. Zeng, J.L. Zhang, W. Huang, Y.K. Bao, The criticality of spare parts evaluating model using an artificial neural network approach, Lecture Notes in Computer Science 3991 (2006) 728–735.
[56] M. Ghiassi, H. Saidane, A dynamic architecture for artificial neural network, Neurocomputing 63 (2005) 397–413.
[57] M. Ghiassi, H. Saidane, D.K. Zimbra, A dynamic artificial neural network model for forecasting time series events, International Journal of Forecasting 21 (2) (2005) 341–362.
[58] P. Civicioglu, Backtracking search optimization algorithm for numerical optimization problems, Applied Mathematics and Computation 219 (15) (2013) 8121–8144.
[59] J.L. Elman, Finding structure in time, Cognitive Science 14 (2) (1990) 179–211.
[60] R. Chandra, M.J. Zhang, Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction, Neurocomputing 86 (2012) 116–123.
[61] J. Zhao, X. Zhu, W. Wang, Y. Liu, Extended Kalman filter-based Elman networks for industrial time series prediction with GPU acceleration, Neurocomputing 118 (2013) 215–224.
[62] H. Jaeger, The echo state approach to analysing and training recurrent neural networks. GMD Report 148, GMD-German National Research Institute for Computer Science, 2001.
[63] F.M. Bianchi, S. Scardapane, A. Uncini, A. Rizzi, A. Sadeghian, Prediction of telephone calls load using Echo State Network with exogenous variables, Neural Networks 71 (2015) 204–213.
[64] H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science 304 (5667) (2004) 78–80.
[65] N. Chouikhi, B. Ammar, N. Rokbani, A.M. Alimi, PSO-based analysis of Echo State Network parameters for time series forecasting, Applied Soft Computing 55 (2017) 211–225.
[66] L. Shen, J. Chen, Z. Zeng, J. Yang, J. Jin, A novel echo state network for multivariate and nonlinear time series prediction, Applied Soft Computing 62 (2018) 524–535.
[67] T. Jasic, D. Wood, Neural network protocols and model performance, Neurocomputing 55 (3–4) (2003) 747–753.
[68] G.P. Zhang, B.E. Patuwo, M.Y. Hu, A simulation study of artificial neural networks for nonlinear time-series forecasting, Computers and Operations Research 28 (4) (2001) 381–396.
[69] R.J. Hyndman, Time Series Data Library (TSDL), 2013, Available online at url: http://robjhyndman.com/TSDL/. (01-01-13).
[70] H. Demuth, M. Beale, M. Hagan, Neural Network Toolbox User's Guide, The MathWorks, Natic, MA, 2010.
[71] E. Bauer, R. Kohavi, An experimental comparison of voting classification algorithms: bagging, boosting and variants, Machine Learning 36 (1999) 105–139.
[72] L. Wang, H. Qu, S. Liu, C. Chen, Optimizing the joint replenishment and channel coordination problem under supply chain environment using a simple and effective differential evolution algorithm, Discrete Dynamics in Nature and Society (2014) 1–12, Article ID 709856.
[73] S. Liu, L. Wang, W.W. Huang, Effects of process and outcome controls on business process outsourcing performance: Moderating roles of vendor and client capability risks, European Journal of Operational Research 260 (3) (2017) 1115–1128.
[74] L. Wang, R. Liu, S. Liu, An effective and efficient fruit fly optimization algorithm with level probability policy and its applications, Knowledge-Based Systems 97 (2016) 158–174.
[75] R. Liu, Y.R. Zeng, H. Qu, L. Wang, Optimizing the new coordinated replenishment and delivery model considering quantity discount and resource constraints, Computers & Industrial Engineering 116 (2018) 82–96.
[76] B. Gao, X. Li, S. Liu, D. Fang, How power distance affects online hotel ratings: The positive moderating roles of hotel chain and reviewers' travel experience, Tourism Management 65 (2018) 176–186.