Neurocomputing ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Neurocomputing

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

A neural network based linear ensemble framework for time series forecasting

Ratnadip Adhikari *

Department of Computer Science & Engineering, The LNM Institute of Information Technology, Jaipur, Rajasthan 302031, India

ARTICLE INFO

Article history:
Received 30 June 2014
Received in revised form
29 October 2014
Accepted 8 January 2015
Communicated by P. Zhang

Keywords:
Time series forecasting
Forecasting accuracy
Combining forecasts
Weights selection
Artificial neural networks

ABSTRACT

Combining time series forecasts from several models is a fruitful alternative to using only a single individual model. In the literature, it has been widely documented that a combined forecast improves the overall accuracy to a great extent and is often better than the forecast of each component model. The accuracy of a linear combination of forecasts primarily depends on the associated combining weights. Despite extensive research in this direction, finding out the most appropriate weights is still very challenging. This paper proposes a linear combination method for time series forecasting that determines the combining weights through a novel neural network structure. The designed neural network successively recognizes the weight patterns of the constituent models from their past forecasting records and then predicts the desired set of the combining weights. Empirical results from eight real-world time series show that our approach provides significantly better forecasting accuracies than the component models and other well recognized linear combination schemes. These findings are also verified through ranking methods and a non-parametric statistical test.

1. Introduction

Combining forecasts from different time series models started in the late sixties with the pioneering work of Bates and Granger [1] and since then this approach has been extensively analyzed in the forecasting literature. The combining methodology provides a much better alternative to using only a single model for performing the forecasts [2,3]. The default technique of forecasting is to test several potential models on the in-sample dataset and then select the best among them for generating the desired out-of-sample forecasts. Despite being the most intuitive, this approach of forecasting has a number of serious limitations. First, a time series seldom has the *independence and identical distribution (i.i.d.)* property that is a fundamental requirement of realistic statistical processes [4]. As a result, the model that performed best for the in-sample dataset might not always provide the best forecasts for the unseen future values. Second, a forecasting model is specific to the nature of the time series, i.e. whether the series is generated from a linear or nonlinear process, follows stationary or nonstationary distribution, contains trend, seasonal, or cyclical patterns, etc. Estimating the exact nature requires large number of historical observations, but in practice there is only a very small sample of available data and so the fitted model may be inappropriate. Third, a time series is a dynamical process that keeps on changing continuously with high degree of uncertainty and may even exhibit *regime switches*. This jeopardizes the validity of the forecasting model when new observations are added to the available data. Finally, a particular model is always prone to faulty assumptions, implementation biases and errors in parameter estimation, which considerably affect the desired forecasts [5]. These discouraging facts of single modeling approach motivated the exploration of various forecast combination techniques. A combination of forecasts benefits from the inter-model diversities, mitigates the risks of using an isolated model, and compensates the drawbacks of the individual models. It has been observed in numerous studies that a combination of multiple forecasts improves the forecasting accuracy to a large extent and often comes out as the superior to each constituent model [6,7].

During the past few decades, there has been an overwhelming interest in combining time series forecasts that consequently has led to the development of a large number of forecast combination techniques. A majority of them form a weighted linear combination of the component forecasts. The statistical averaging techniques, e.g. *simple average*, *trimmed mean*, *Winsorized mean*, *median*, etc. are the most basic ensemble methods, as they do not explicitly determine the combining weights. Many studies found that these fairly simple methods reasonably outperformed a number of more advanced combining schemes [8–10]. Jose and Winkler [8] have meticulously studied the performances of these four statistical ensemble techniques in combining forecasts for the widely popular M3 competition

* Tel.: +91 9660933276.
E-mail address: adhikari.ratan@gmail.com

http://dx.doi.org/10.1016/j.neucom.2015.01.012
0925-2312/

datasets [6]. The outcome of their research was that both trimmed as well as Winsorized mean are competent alternatives to simple average and median in combining forecasts. However, one major downside of these methods is that they do not consider the relative performances of the individual models and are mainly suitable when the component forecasts have comparable accuracies. There are various other sophisticated linear combination methods, which assign weights to the constituent forecasts on the basis of the past forecasting records of the respective models. A common approach is to select each individual weight to be the normalized unbiased inverse of the in-sample absolute forecasting error of the respective model. This scheme follows the intuitive notion that a model with more error should get less weight and vice versa. Granger and Ramanathan [11] interpreted the forecast combination methodology in a regression framework, where the time series observations and the individual forecasts are considered to be the dependent and explanatory variables, respectively. The combining weights are then determined through the *Ordinary Least Square (OLS)* regression method. Later, Aksu and Gunter [12] in their extensive study found that the performance of the OLS method is reasonably improved through considering unbiased weights. A well-known alternative to the OLS technique is the *minimum-variance* method [13] that determines the weights by minimizing the variance of the combined forecast error. The *outperformance* method, proposed by Bunn [14] adopts a Bayesian framework of subjective probabilities to assign the combining weights. It assumes that the weight to each component forecast is the probability that the respective model will outperform the others, i.e. produce the least error in the next trial. It is a robust non-parametric approach that is found to achieve reasonably good accuracy when there are few historical data or the weights are subjective to expert judgment [10].

Over the past decade, a lot of research interests has also been observed in forming *homogeneous ensemble* of Artificial Neural Networks (ANNs). Such a framework combines several ANNs with varying architectures in order to achieve better overall accuracy. Evidently, an ensemble of ANNs is beneficial only when there is a considerable amount of disagreements among the ANN outputs. Krogh and Vedelsby [15] formulated a method that relates this disagreement, termed as the *ensemble ambiguity* and the *generalization error* to construct successful ANN ensembles. In another important work, Zhou et al. [16] minutely analyzed the relationship between the ensemble accuracy and the constituent neural networks. They employed *genetic algorithm* to develop a method that selectively combines a class of neural networks from the available choices. Their work found that it is a better strategy to selectively combine many neural networks, instead of all. An intensive review on combining forecasts was provided by Clemen [17] that is little old now, but still very helpful in summarizing the development of the subject. Some good recent reviews in this domain are the works of Timmermann [7], De Gooijer and Hyndman [9], De Menezes et al. [10], and Lemke and Gabrys [18].

The majority of the works on combining forecasts are devoted to determine the optimal weights through minimizing the *Sum of Squared Error (SSE)*, calculated from the original dataset and its combined forecast. However, the actual SSE is unknown in advance and hence must be estimated from the available in-sample observations. The most common resolution is to estimate the unknown SSE from some in-sample training and validation sets. But, in this approach the determined weights are biased to the particular validation dataset and so the combined forecasts may be inappropriate if there are significant dissimilarities among the in-sample validation and future observations. Furthermore, the weights are highly sensitive to the changes in the validation dataset and so the obtained combined forecasts are often quite unstable. There are some sophisticated combining schemes in the literature those mitigate this problem to some extent. Among them, the *Recursive Least Squares (RLS)* [19,20] is quite well-known that recursively updates the least squares weights with addition of new observations. The common variants of this method are the *dynamic RLS* and *covariance addition RLS (RLS-CA)*, which actually belong to the broad class of Kalman filtering algorithms [19,20]. But, these methods are not straightforward to use and are also computationally quite expensive, which overshadow the achieved accuracy improvements through them. Bunn's outperformance method also imparts somewhat dynamic nature to weight estimation through performing a number of validation trials on the in-sample dataset. However, there are disputes regarding the interpretation, appropriateness, and validity of the associated outperformance probabilities [17,21].

This paper proposes a new linear combination method for time series forecasting in which the weights are determined from the forecasting results of the individual models on several in-sample datasets. The models are successively applied for a number of in-sample forecasting trials and weights are assigned to them on the basis of their obtained absolute errors, in an inversely proportionate manner. A set of in-sample weights of all models is formed and then an ANN is fitted to this set. To recognize the in-sample weight pattern, we use an ANN because of its flexible, data-driven, and model-free structure with remarkably good learning and generalization ability. Thus, the proposed approach interprets the in-sample set of weights as a new time series, whose inherent pattern is identified and learned through a novel ANN model. The desired combining weights are then predicted from this fitted ANN. In this manner, the distinguished learning and generalization ability of ANN is utilized to dynamically determine the combining weights from several past forecasting records of the component models. The effectiveness of the proposed combination method is tested with four individual models on eight real-world time series. The forecasting performances of the proposed method are compared with the individual models as well as other popular conventional linear combination schemes in terms of two well-known absolute error measures.

The remainder of the paper is organized as follows. Section 2 briefly describes the individual forecasting models and Section 3 explains the linear forecast combination methodology. Our proposed combination algorithm is described in Section 4. Section 5 reports the empirical results, followed by summary and conclusions in Section 6.

2. The time series forecasting models

A time series is a sequential collection of observations, recorded at consecutive time periods. Univariate time series with discrete values are most widely studied in the literature, where such a series is represented as $\mathbf{Y} = [y_1, y_2, \ldots, y_N]^{\mathrm{T}}$, y_t being the observation at time t. Time series forecasting is the projection of desired number of future values through some mathematical model. In usual forecasting paradigm, an appropriate model is identified from a particular class and it is then used to generate the future values. However, as discussed in the outset, this approach of using only a single model has a number of inevitable limitations, which encouraged combining forecasts as a fruitful alternative. A combination of forecasts is based on the fundamental rationale that any individual model is prone to misspecification and inadequacy, whereas several non-optimal models together can very closely approximate the actual data generation process [5,18]. But, a forecast combination can be effective only when there are considerable extent of diversities among the individual models [6,5,15,16]. Another factor that significantly affects the combined forecasting accuracy is the number of component models. Ideally, the ensemble should neither include models with very poor forecasting results nor discard any potentially good model. Identifying the exact number of component models is quite difficult, due

Please cite this article as: R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.01.012

to the lack of theoretical guidelines in this regard. In the extensive study with the *M-competition* time series datasets, Makridakis and Winkler [22] found that most of the possible error reduction is achieved through combining at most five models. But, the gain in accuracy rapidly diminished with inclusion of more models. Armstrong [5] also recommended the use of at most four or five models in the ensemble.

In view of the aforementioned facts, in this study we use four component models from diverse classes to form our ensemble. These are the *Box–Jenkins* [23,24], *Feedforward ANN* (*FANN*) [18,25,26], *Elman ANN* (*EANN*) [27,28], and *Support Vector Machine* (*SVM*) [29,30]. These four models, together with the differences among them and the rationale behind their selection are discussed in the next paragraphs.

2.1. The Box–Jenkins models

The *Autoregressive Integrated Moving Average (ARIMA)* models, pioneered by Box and Jenkins [24], are the backbone of almost all statistical forecasting methods. An ARIMA(p,d,q), commonly known as a Box–Jenkins model assumes that each future value of a time series is a linear function of several past observations and *white noise* terms [23,24,26]. It employs an ordinary differencing process of order d to make a time series stationary. For seasonal time series forecasting, the basic ARIMA model is extended to the *Seasonal ARIMA (SARIMA)*, represented as SARIMA$(p,d,q)\times(P,D,Q)^s$, "s" being the period of seasonality [31].

2.2. The FANN model

Over the past few decades, FANNs have emerged as a very successful alternative class to the traditional statistical models for time series forecasting [2,25,26]. They have gained wide recognition in the forecasting domain due to their non-parametric and nonlinear modeling skill, flexible and data-driven nature, good generalization ability, and reasonably good accuracy. Also, as proved by Hornik et al. [32], FANNs are *universal approximators*, as they can estimate any nonlinear continuous function to any desired accuracy. An FANN is most commonly represented as a $p\times h\times 1$ structure that consists of p *input nodes*, h *hidden nodes*, and one *output node*. The network weights and biases are identified in the training phase through the *backpropagation* algorithm [33,25]. The network size significantly affects the FANN performance, but it is difficult to appropriately estimate this parameter in advance. There are a few heuristic criteria in the literature for selecting the network size. In this study, we use the well-known *Bayesian Information Criterion (BIC)* [25,34,35] that controls the network size by penalizing for every increase in the number of network weights, given by $N_{p,h}=h(p+2)+1$.

2.3. The EANN model

Alongside FANNs, recurrent neural networks have also provided promising outcomes in time series forecasting. An EANN is a local recurrent neural network that differs from an FANN through introducing an extra *context layer* and *feedback connections* [27,28]. The feedforward structure is maintained among the input, hidden, and output layers, but a feedback structure is adopted between the hidden and context layers. The context nodes receive the reverting feedback connections from the hidden nodes at each step and as such, they preserve the records of previous network operations. This recurrence makes the network dynamic, thereby allowing it to perform nonlinear time-varying mappings of input and target patterns [28,36]. There has been little work on EANN architecture selection, but an widely agreed fact is that EANNs need much more hidden nodes than FANNs to adequately model the temporal relationships [18,37]. Commonly, the number of hidden nodes of an EANN is affixed beforehand by some rule and this approach is used in the present study, as well.

2.4. The SVM model

SVM, developed by Vapnik [29], has been successfully applied in many time series forecasting problems. It is a statistical learning theory, based on the *Structural Risk Minimization (SRM)* principle that attempts to obtain an optimal decision rule through minimizing the combination of the empirical risk and the capacity of the hypothesis space. The goal of SRM is achieved through a subset of training data points, known as *support vectors*. Time series forecasting is a branch of *Support Vector Regression (SVR)* that predicts real-valued outputs through constructing a maximum margin hyperplane, after nonlinearly mapping the original input space to a higher dimensional feature space. The explicit construction of the mapping is avoided through using a kernel function that satisfies the *Mercer's condition* [38,39]. In the present study, we use the *Radial Basis Function* (*RBF*) kernel, due to its impressive results in forecasting applications. This kernel is defined as $K(\mathbf{x},\mathbf{y})=\exp\left(-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2\right)$, where $\mathbf{x}$, $\mathbf{y}$ are the input vectors and σ is a tuning parameter [40].

2.5. Comparison of the models and rationale behind their selection

In the preceding paragraphs, we have briefly described four forecasting models, chosen from conceptually different domains. In this section, we discuss about the diversities among these four models, their respective pros and cons, rationale behind their selection, and their individual contributions in the ensemble.

- The Box–Jenkins models are remarkably efficient in forecasting time series, generated from purely linear processes. But, due to this linearity restriction, their accuracy is not so impressive for nonlinear time series datasets [26].
- Unlike Box–Jenkins models, FANNs are self-adaptive and data-driven, as they do not require any prior assumption about the data generation process of the time series [26,41]. In addition, they are very powerful in modeling nonlinear time series. But, FANNs are not so effective for linear processes and their forecasting accuracy is quite sensitive to the underlying architecture [26,42]. Even a slight misspecification of the parameters and training algorithm may yield disastrous outputs. Above all, FANNs always possess the inevitable risk of getting struck to a local minimum that significantly hampers the forecasting results [25].
- An FANN recognizes the spatial relationship among the successive observations, but ignores the temporal relation. An EANN overcomes this limitation through performing nonlinear time dependent mappings with introducing a context layer and feedback connections [18,28]. But, this enhancement comes with increased computational cost, as an EANN needs considerably more hidden nodes than its feedforward counterpart [18,36,37].
- SVM is based on the SRM philosophy that is entirely different from other models. Together with excellent out-of-sample generalization skill, SVM overcomes a major shortcoming of neural networks by always producing the unique global optimal solution [30]. In addition, it mitigates the problem of a low dimensional input space through kernel function. According to the nature of the kernel, SVM is capable of effectively modeling both linear and nonlinear time series. However, like all forecasting methods, SVM too has some crucial pitfalls. These include its high sensitiveness to the hyper-parameters, algorithmic complexity, slow speed, and extensive memory requirements for large datasets [43].

Please cite this article as: R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.01.012

The above discussion shows that there exists significant extent of diversities among the four models and each of them has a unique contribution in the ensemble. A combination scheme attempts to benefit from the strengths of these models, simultaneously mitigating their weaknesses. It is to be noted that none of the ensemble techniques which we discuss in the upcoming sections is solely dependent on these models. In fact, each ensemble method can be applied for any set of chosen component models.

3. Linear combination of time series forecasts

A linear ensemble is the most common approach of combining multiple forecasts [19,10]. For the dataset $\mathbf{Y}=[y_1,y_2,\ldots,y_N]^{\mathrm{T}}$ and its n forecasts $\hat{Y}^{(i)}=\left[\hat{y}_1^{(i)},\hat{y}_2^{(i)},\ldots,\hat{y}_N^{(i)}\right]^{\mathrm{T}}$ $(i=1,2,\ldots,n)$, the forecasts from a linear combination are given by

$$\hat{y}_k=w_0+w_1\hat{y}_k^{(1)}+w_2\hat{y}_k^{(2)}+\ldots+w_n\hat{y}_k^{(n)}\,\forall k=1,2,\ldots,N. \tag{1}$$

The terms w_i $(i=1,2,\ldots,n)$ are the combining weights, which are often assumed to be nonnegative, i.e. $w_i\geq 0\ \forall i$ and unbiased, i.e. $\sum_{i=0}^{n}w_i=1$. The introduction or non-introduction of the constant term w_0 has notable impact on the combined forecasting accuracy, but usually it is discarded. Omitting the constant term, the linear combination of forecasts can be conveniently expressed as follows:

$$\hat{Y}=\mathbf{Fw} \tag{2}$$

where, $\hat{Y}=[\hat{y}_1,\hat{y}_2,\ldots,\hat{y}_N]^{\mathrm{T}}$ is the combined forecast, $\mathbf{F}=\left[\hat{Y}^{(1)}|\hat{Y}^{(2)}|\ldots|\hat{Y}^{(n)}\right]$ is the $N\times n$ forecast matrix and $\mathbf{w}=[w_1,w_2,\ldots,w_n]^{\mathrm{T}}$ is the weight vector. The total absolute forecasting error of a linear combination of forecasts satisfies the property:

$$\lambda\left(\mathbf{Y},\hat{Y}\right)\leq\lambda\left(\mathbf{Y},\hat{Y}^{(i)}\right)$$
$$\forall i\in S,\quad S\subseteq\{1,2,\ldots,n\}. \tag{3}$$

Here, λ is a total absolute forecasting error function, e.g. the forecast SSE. The maximum benefit of a linear combination of forecasts is achieved, when the inequality (3) is satisfied for all individual models, i.e. when $S=\{1,2,\ldots,n\}$. A block diagram of the linear combination mechanism, used in this paper, is shown in Fig. 1.

The *simple average* is the easiest combination method that assigns equal weights to all component forecasts and as such, it is free from any kind of weight estimation errors. Research evidences show that the naïve simple average often achieved remarkably good accuracy and outperformed many other advanced linear combination methods [8,9,17]. One major downside of the simple average is its high sensitiveness to extreme values and in order to overcome this, other statistical averages have also been investigated in the literature. The *trimmed mean* computes the simple average by discarding an equal number of smallest and largest forecasts. The *median* has also been used for combining forecasts and it yielded mixed results [8,44]. The simple average and median both are in fact special cases of the trimmed mean, corresponding to no trimming and full trimming, respectively. It should be noted that a trimmed mean is feasible, only when there are three or more models, i.e. when $n\geq 3$. For $n=3,\ 4$, it is same as the median.

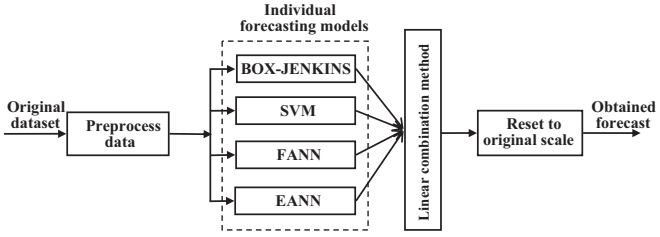

Fig. 1. The block diagram of our linear combination framework.

The statistical averaging techniques are easy to implement and interpret, but they follow predefined weight assignment rules, independent of the relative forecasting performances of the component models. As such, these methods are suitable when the individual models have similar accuracies. An *Error-based (EB)* combining scheme assigns each weight as the normalized unbiased inverse absolute forecasting error of the respective model, so that

$$w_i=e_i^{-1}\Big/\sum_{i=1}^{n}e_i^{-1}$$
$$\forall i=1,2,\ldots,n. \tag{4}$$

Here, e_i denotes the past forecasting error of the ith model that is usually calculated from an in-sample pair of training and validation datasets. This scheme is based on the simple principle that the forecast from a model with more in-sample error receives less weight and vice versa. A total absolute squared or percentage error measure is commonly used to evaluate the in-sample forecasting precisions of the component models [5,35].

The *Least Square Regression (LSR)* is another popular method that attempts to find the optimal weights by minimizing the combined forecast SSE. Using the LSR principle, the optimal weight vector for the linear combination framework (2) can be obtained as $\mathbf{w}=\mathbf{F}^{+}\mathbf{Y}$, where, $\mathbf{F}^{+}$ is the *Moore–Penrose pseudo-inverse* of $\mathbf{F}$ [19,35]. Evidently, $\mathbf{F}^{+}=(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}$ if $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ is invertible, otherwise $\mathbf{F}^{+}$ can be obtained from the *singular value decomposition* of $\mathbf{F}$ [45]. It can be seen that the calculation of the optimal weights requires the knowledge of the actual future dataset that is to be forecasted and obviously this dataset is unknown in advance. The most common resolution is to estimate the weights from an in-sample validation set. However, the obtained weights in this manner depend on the specific validation set and may not be the appropriate weights, those actually minimize the combined forecast SSE.

Bunn [14] adopted a Bayesian probabilistic framework that assigns weight to each component model on the basis of the number of times it attained the best accuracies in the past in-sample forecasting experiments. After M forecasting trials, the outperformance probability k_i $(i=1,2,\ldots,n)$ is calculated as a fraction of M for which the ith model achieved the best accuracies. These probabilities are reasonably assumed to follow the multivariate beta distribution. The *outperformance method* is a robust non-parametric combination approach with notably good results, although there are debates regarding the appropriateness and validity of the outperformance probabilities [17,21].

There are various other linear combination methods in the literature which adopt different approaches for estimating the combining weights. The central motive of all these methods is to determine the weights in such a manner that they give closest estimation of the data generation process of the time series [7,9,10].

4. The proposed linear combination method

We propose a method for linearly combining time series forecasts that attempts to determine the combining weights after analyzing their patterns in successive in-sample forecasting trials. The proposed approach can be divided into the following three phases. First, the individual models are applied to consecutive in-sample forecasting trials and weights are assigned to them on the basis of their inverse absolute forecasting errors. Then, a set of weights is formed that is a sequential collection of the assigned in-

Please cite this article as: R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.01.012

sample weights to all the individual models. In the next phase, a neural network is designed to recognize and learn the inherent pattern in this in-sample weight set. Finally, the desired weights are predicted through this fitted neural network and are used to obtain the combined forecasts. Thus, our approach identifies the combining weights through recognizing their underlying pattern from the past forecasting records of the component models. The details of the proposed combination method are described below.

4.1. Description of the proposed linear combination scheme

Let, $\mathbf{Y}=[y_1,y_2,\ldots,y_N]^{\mathrm{T}}$ be the time series that is divided into the in-sample training dataset $\mathbf{Y}_{\mathrm{tr}}=[y_1,y_2,\ldots,y_{N_{\mathrm{tr}}}]^{\mathrm{T}}$ and the out-of-sample testing dataset $\mathbf{Y}_{\mathrm{ts}}=[y_{N_{\mathrm{tr}}+1},y_{N_{\mathrm{tr}}+2},\ldots,y_{N_{\mathrm{tr}}+N_{\mathrm{ts}}}]^{\mathrm{T}}$. The dataset $\mathbf{Y}_{\mathrm{tr}}$ is again subdivided into a suitable number of consecutive training and validation pairs $\left(\mathbf{Y}_{\mathrm{tr}}^{(j)},\mathbf{Y}_{\mathrm{vd}}^{(j)}\right)$ as follows:

$$\left.\begin{aligned}\mathbf{Y}_{\mathrm{tr}}^{(j)}&=\left[y_1,y_2,\ldots,y_{base+j-1}\right]^{\mathrm{T}}\\ \mathbf{Y}_{\mathrm{vd}}^{(j)}&=\left[y_{base+j},y_{base+j+1},\ldots,y_{base+j+(N_{\mathrm{vd}}-1)}\right]^{\mathrm{T}}\end{aligned}\right\}$$
$$\forall j=1,2,\ldots,M. \tag{5}$$

Here, N_{vd} is the size of the validation set and $M(M<<N_{\mathrm{tr}})$ is the total number of in-sample training–validation pairs. The term $base=N_{\mathrm{tr}}-N_{\mathrm{vd}}-M+1$ is the length of the first training set $\mathbf{Y}_{\mathrm{tr}}^{(1)}$. The sum of the lengths of the last training and validation subsets is essentially equal to N_{tr}. In (5), a new data point is iteratively added to the end of the previous training subset and the previous validation subset is moved one point forward by deleting the same value from the beginning. So, the size of the training subset increases by one at each step, whereas that of the validation subset remains throughout constant. Now, each model is trained on the training subsets and is used to forecast the respective validation subsets. The forecasting efficiency of each model is evaluated through some total absolute error measures. In this study, we consider the following four error measures: *Mean Absolute Error (MAE)*, *Mean Squared Error (MSE)*, *Root Mean Squared Error (RMSE)*, and *Mean Absolute Percentage Error (MAPE)*. These error measures are defined below:

$$\mathrm{MAE}=\frac{1}{N_f}\sum_{t=1}^{N_f}|y_t-\hat{y}_t| \tag{6}$$

$$\mathrm{MSE}=\frac{1}{N_f}\sum_{t=1}^{N_f}(y_t-\hat{y}_t)^2 \tag{7}$$

$$\mathrm{RMSE}=\frac{1}{N_f}\sqrt{\sum_{t=1}^{N_f}(y_t-\hat{y}_t)^2} \tag{8}$$

$$\mathrm{MAPE}=\frac{1}{N_f}\sum_{t=1}^{N_f}\left|\frac{y_t-\hat{y}_t}{y_t}\right|\times 100 \tag{9}$$

Here, y_t and $\hat{y}_t$ are the actual and forecasted observations, respectively and N_f is the total number of observations. MAE and MSE are quite effective in evaluating the total absolute forecasting error, but they are very sensitive to extreme errors as well as change of scale and data transformations. RMSE has similar properties like MSE, but it is far less sensitive to the extreme errors and is more stable. MAPE is another effective measure that evaluates the percentage of average absolute error and is independent of the scale of measurement. On the basis of these error measures, we assign the weight to the ith model for the jth forecasting trial as follows:

$$w_i^j=\frac{\exp\left(v_i^j\right)}{\sum_{i=1}^{n}\exp\left(v_i^j\right)}$$
$$\forall i=1,2,\ldots,n,\ \ j=1,2,\ldots,M. \tag{10}$$

Here, $v_i^j=\left(\mathrm{MAE}_i^j+\mathrm{RMSE}_i^j+\mathrm{MAPE}_i^j\right)^{-1}$. The three terms in v_i^j represent the respective errors of the ith model for the jth validation set.

The weights to the forecasting models for each training and validation pair are assigned using the so-called SOFTMAX distribution [46] and are inversely proportional to the combined effect of their three respective error measures. The differences in magnitudes of the error measures are regularized through using the exponential function. Each weight in a specific in-sample forecasting trial is nonnegative and unbiased. Now, let, $\mathbf{w}^j=\left[w_1^j,w_2^j,\ldots,w_n^j\right]^{\mathrm{T}}$ $(\forall j=1,2,\ldots,M)$ be the column vector of weights of the n models for the jth forecasting trial. After M in-sample forecasting trials, we obtain the following $n\times M$ weight matrix:

$$\mathbf{W}=\begin{bmatrix}w_1^1 & w_1^2 & \ldots & w_1^M\\ w_2^1 & w_2^2 & \ldots & w_2^M\\ \vdots & \vdots & \ldots & \vdots\\ w_n^1 & w_n^2 & \ldots & w_n^M\end{bmatrix}_{n\times M}=\left[\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^M\right] \tag{11}$$

Each column of $\mathbf{W}$ represents the weights to the n component models in the respective forecasting trial. Accumulating these weight vectors, we form the in-sample weight dataset $\mathbf{w}=\left[\left(\mathbf{w}^1\right)^{\mathrm{T}},\left(\mathbf{w}^2\right)^{\mathrm{T}},\ldots,\left(\mathbf{w}^M\right)^{\mathrm{T}}\right]^{\mathrm{T}}$ of length nM that exhibits a very interesting property. There is a notable recurrence of some recognizable weight pattern in each forecasting trial, i.e. the weight vectors $\mathbf{w}^j(j=1,2,\ldots,M)$ show some regular recurring patterns, as can be visualized from the two plots in Fig. 2. The primary reason of this recurrence is the *sliding window mechanism* that is used to formulate the consecutive pairs of training and validation subsets.

Fig. 2a and b respectively represents the weight sets, corresponding to 40 forecasting trials with the four individual models for the *River flow* and *Vehicles* time series, which are discussed in Section 5. The vertical and horizontal axes respectively represent the normalized SOFTMAX weights $w_i^j(i=1,2,\ldots,n;\ j=1,2,\ldots,M)$ and their indices in the weight set $\mathbf{w}$. In each plot, we can observe that the weights follow a regular pattern that almost recurs after every interval of four terms. It should be noted that this recurrence of weights pattern would be absent, if the training–validation pairs were selected haphazardly and not through the sliding window scheme. We use a neural network model to identify and learn this regular repeating form in the in-sample weight set. From previous works, it has been found that ANNs are very powerful and efficient in recognizing recurring pattern in a dataset. For example, Hamzaçebi [31] developed a novel ANN architecture, known as the *Seasonal ANN (SANN)* to effectively model the regular fluctuation in a seasonal time series that recurs in every season. SANN utilizes the direct forecasting approach of neural networks and has been found to provide considerably improved forecasting results for a variety of seasonal time series. In the present scenario, the in-sample weight set is also quite akin to a seasonal time series and so an analogous ANN structure can be used. We form an ANN with number of input and output nodes to be both equal to the number of component models. So, the designed ANN has an $n\times h\times n$ architecture, where "h" is the number of nodes in the hidden layer. It has $(M-1)$ input and output patterns, constituted by the weight vectors $\mathbf{w}^j=\left[w_1^j,w_2^j,\ldots,w_n^j\right]^{\mathrm{T}}$ and $\mathbf{w}^{j+1}=\left[w_1^{j+1},w_2^{j+1},\ldots,w_n^{j+1}\right]^{\mathrm{T}}$ $(\forall j=1,2,\ldots,M-1)$, respectively. The structure of this ANN model is depicted in Fig. 3.

The designed ANN learns the weights pattern through each pair of input and output vectors. One major benefit with this model is

Please cite this article as: R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.01.012

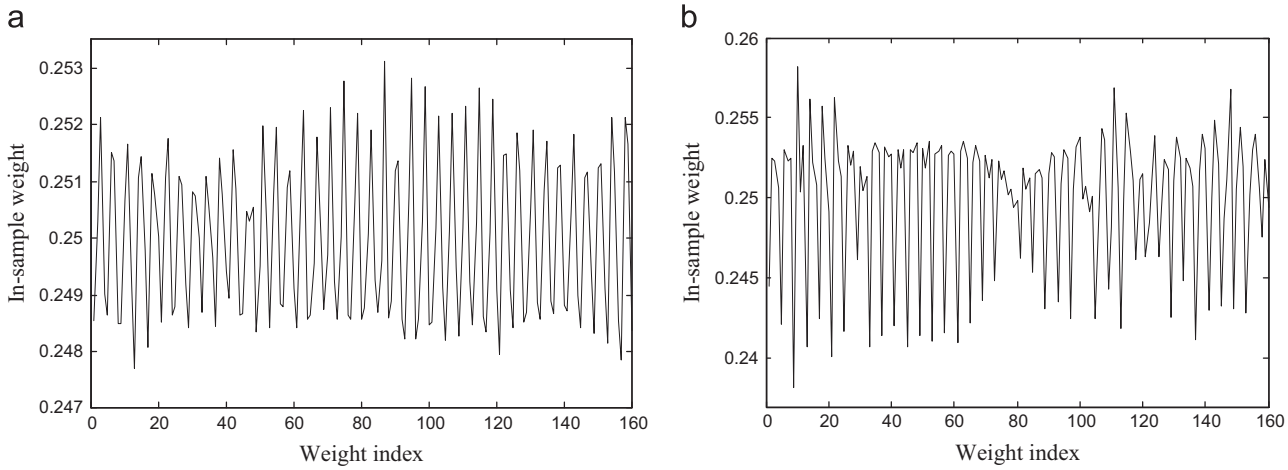

Fig. 2. The recurring in-sample weight patterns for the time series: (a) river flow and (b) vehicles.

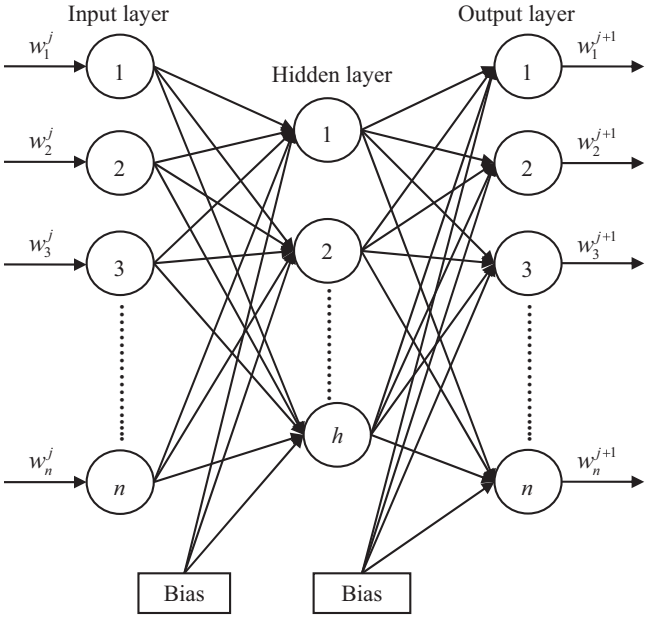

Fig. 3. The ANN architecture for the in-sample weight set.

its highly efficient yet simple architecture that only requires the selection of the number of hidden nodes. The number of input and output nodes is already specified beforehand. After the model fitting phase, the last weight vector $\mathbf{w}^M$ is passed as the input to the fitted ANN in order to obtain the final predicted weights as $\mathbf{w}=[w_1, w_2, \ldots w_n]^{\mathrm{T}}$. The requisite steps of the proposed combination method are presented in Algorithm 1.

Algorithm 1. The proposed ensemble framework.

Inputs: The training set $\mathbf{Y}_{\mathrm{tr}}=[y_1, y_2, \ldots, y_{N_{\mathrm{tr}}}]^{\mathrm{T}}$, the testing set size N_{ts}, the n forecasting models $\mathrm{M}_i (i=1,2,\ldots,n)$, the corresponding n component forecasts $\hat{Y}^{(i)}=\left[\hat{y}^{(i)}_{N_{\mathrm{tr}}+1}, \hat{y}^{(i)}_{N_{\mathrm{tr}}+2}, \ldots, \hat{y}^{(i)}_{N_{\mathrm{tr}}+N_{\mathrm{ts}}}\right]^{\mathrm{T}}$, and the number M of in-sample forecasting trials.

Output:The combined forecast $\hat{Y}=\left[\hat{y}_{N_{\mathrm{tr}}+1}, \hat{y}_{N_{\mathrm{tr}}+2}, \ldots, \hat{y}_{N_{\mathrm{tr}}+N_{\mathrm{ts}}}\right]^{\mathrm{T}}$.

1. $\mathbf{w}=[]$ //*Initialize* $\mathbf{w}$ *as the empty column vector*
2. **for** $i=1$ to n **do**
3. $\mathbf{w}^j=[]$ // *Initialize* $\mathbf{w}^j$ *as the empty column vector*
4. **for** $j=1$ to M **do**
5. Divide $\mathbf{Y}_{\mathrm{tr}}$ into the training–validation pair $\left(\mathbf{Y}^{(j)}_{\mathrm{tr}}, \mathbf{Y}^{(j)}_{\mathrm{vd}}\right)$ through (5).
6. Fit the model M_i to $\mathbf{Y}^{(j)}_{\mathrm{tr}}$ and use it to predict $\mathbf{Y}^{(j)}_{\mathrm{vd}}$.
7. Compute MAE^j_i, RMSE^j_i, MAPE^j_i as the errors of M_i for $\mathbf{Y}^{(j)}_{\mathrm{vd}}$.
8. Compute the SOFTMAX weight w^j_i through (10).
9. $\mathbf{w}^j=\left[(\mathbf{w}^j)^{\mathrm{T}} \middle| w^j_i\right]^{\mathrm{T}}$ //*Append* w^j_i *to* $\mathbf{w}^j$
10. **end for**
11. $\mathbf{w}=\left[\mathbf{w}^{\mathrm{T}} \middle| (\mathbf{w}^j)^{\mathrm{T}}\right]^{\mathrm{T}}$ //*Append* $(\mathbf{w}^j)^{\mathrm{T}}$ *to* $\mathbf{w}$
12. **end for**
13. Fit an appropriate $n \times h \times n$ ANN model to $\mathbf{w}(1 : n(M-1))$.
14. Give $\mathbf{w}(n(M-1)+1 : nM)=\left[w^M_1, w^M_2, \ldots, w^M_n\right]^{\mathrm{T}}$ as input to the fitted ANN.
    //*The last n weights of* $\mathbf{w}$ *are given as the inputs to the ANN*
15. Obtain the n predicted weights as $\mathbf{w}=[w_1, w_2, \ldots w_n]^{\mathrm{T}}$.
16. Obtain the final combined forecast as $\hat{Y}=\sum_{i=1}^{n} w_i \hat{Y}^{(i)}$.

4.2. Computational complexity of our combination method

It is evident that accurately finding the computational complexity of a soft-computing algorithm is often not straightforward. But, an estimation of the asymptotic time complexity of such an algorithm is very helpful in understanding its computational intricacy and identifying scopes for further enhancements. Here, we consider the complexity that is solely contributed by our ensemble only and not that of finding the component forecasts, as the later one is common to any combination scheme.

Let, $C_i (i=1,2,\ldots,n)$ be the asymptotic time complexity for implementing the ith individual model on an in-sample forecasting trial. Although, C_i varies for different trials, but in asymptotic sense, it can be considered to be fixed. Then, the time complexity for M forecasting trials is of order $M\sum_{i=1}^{n} C_i \approx n\mathrm{MC}_{\max}$, where $C_{\max}$ is assumed to be the maximum time complexity of an individual model. Now, assuming that each basic mathematical operation requires a constant time, the asymptotic time complexity of computing the weight $w^j_i (\forall i=1,2,\ldots,n;\ j=1,2,\ldots,M)$ will be of $\mathcal{O}(nN_{\mathrm{vd}})$. Next, the time complexity of fitting an $n \times h \times n$ ANN is essentially a function of the total number of network weights [47], i.e. of $N_{n,h}=h(2n+2)+n$ So, this can be assumed to be of $\mathcal{O}(f(N_{n,h}))$, where f is some function of $N_{n,h}$. Now, accumulating the computational costs of all the components, the

asymptotic time complexity of our linear combination method is obtained as

$$T = \mathcal{O}(nMC_{\max} + nN_{\text{vd}} + f(N_{n,h})) \tag{12}$$

From (12), we notice that the asymptotic time complexity T of the proposed combination algorithm is a function of n, h, N_{vd}, M, and $C_{\max}$. Evidently, the proposed method requires more computational time than some other existing linear combination schemes. But, in time series analysis, the forecasting accuracy is much more important than the computational complexity of the method. Moreover, in the present era of high speed computing systems, the time complexity of a forecasting method is not a severe performance barrier for a time series of moderate length. The increased computational time of the proposed combination method is justifiably traded off with the substantially increased accuracy. We should also note that the proposed formulation enables us to achieve very good accuracy with a reasonably small number of in-sample forecasting trials and as such, the computational time can be effectively controlled.

5. Empirical results and discussions

This section presents the empirical works, performed to study the effectiveness of the proposed linear combination method. Eight discrete time series, representing real-world phenomena, are used in this study. All the raw series are available at the *Time Series Data Library* (*TSDL*) [48], an open repository of a wide collection of time series datasets. These eight time series are (1) *Lynx* contains the annual number of lynx trapped in the Mackenzie River district of Northern Canada from 1821 to 1934, (2) *Sunspots* contain the annual number of observed sunspots from 1700 to 1987, (3) *River flow* contains the monthly flow in cms of the Clearwater river at Kamiah, Idaho, USA from 1911 to 1965, (4) *Vehicles* contain monthly sales of vehicles in USA from January 1971 to December 1991, (5) *RGNP* contains the yearly real GNP values of USA from 1890 to 1974 in billions of dollars, (6) *Wine* contains the monthly Australian sales of red wine in thousands of liters from January 1980 to July 1995, (7) *Airline* contains the monthly number of international airline passengers in thousands from January 1949 to December 1960, (8) *Industry* contains quarterly industrial observations, starting from the first quarter of 1977 and ending at the last quarter of 1992.

Fig. 4 depicts the plots of the eight time series datasets. The first three time series are stationary and exhibit quite regular movements, which is also verified through the augmented *Dickey-Fuller unit root test* [49]. The Lynx series has a periodic pattern with about a 10 year cycle and is extensively studied in the time series literature, especially to analyze Box–Jenkins and neural network models. It is common to use the base 10 logarithm of this dataset for stabilizing its variance [26,35] and this transformation is used in the present study. The Sunspots series that represents the yearly records of observed sunspots has significant practical importance. It has an average cycle of about 11 years. The River flow series also follows a somewhat similar pattern with fluctuating between notably high and low values. The Vehicles and RGNP are nonstationary time series, both having strong upward trends. Modeling of nonstationary data is in general quite challenging. Both the Vehicles and RGNP datasets are at first made stationary, through a single ordinary differencing process. Each of the last three time series exhibits a multiplicative seasonality, i.e. the size of the seasonal fluctuation varies with the means. The Wine and Airline series have monthly seasonal patterns with clear upward trends, whereas the Industry series has quarterly seasonality with a downward trend. These three time series are deseasonalized through applying a seasonal differencing and are also detrended before fitting the forecasting models.

The experiments in this study are carried out on MATLAB. The automatic model selection toolbox ARMASA [50] is used for fitting the ARIMA and SARIMA models. The SVM code blocks of Chapelle [40] and the neural network toolbox [37] are respectively used for fitting the SVM and ANN models. FANNs and EANNs are trained through the functions `trainlm` and `traingdx`, respectively. The numbers of input and hidden nodes of the individual FANN models for nonseasonal data are selected from the set $\{1, 2, \ldots, 12\}$. Thus, for each nonseasonal time series, we have investigated a total 144 FANN models on the in-sample dataset and then selected the one that produced the least BIC value. We have excluded the possibility of zero hidden nodes, as in this case the resulting FANN just turns out to be a linear autoregressive model [25,26]. An $s \times h \times 1$ FANN structure is chosen for seasonal time series, where "s" is the period of seasonality that is 12 for monthly and 4 for quarterly series. It has been observed that choosing the number of input nodes to be equal to the seasonal period helps the neural network to efficiently learn the seasonal effect [31]. The number of hidden nodes, "h" is selected from the set $\{1,\ 2,\ 3,\ 4,\ 5,\ 6\}$. It means that we have experimented with six potential FANN models for each seasonal dataset, before identifying the optimal one. Each EANN has the same structure as the corresponding FANN, except the number of hidden nodes that is fixed to be 15 for all time series. The requisite modeling information for all the eight time series datasets are presented in Table 1.

The performance of our proposed forecast combination algorithm primarily depends on the values of the parameters M, N_{vd}, and the architecture of the suggested ANN model. A very large value of M increases the size of the in-sample weight set and the time complexity of the algorithm. Consequently, the weights from many training–validation pairs are included which are far away from the actual testing dataset. We have found that increasing the number of forecasting trials after a certain level reasonably degrades the precision, instead of improving it. On the other hand, if M is too small, then the combination becomes bias to the most recent in-sample observations and may be inadequate. However, there is no theoretical guideline to find the exact number of forecasting trials. From our experiments with different values of M, ranging from 25 to 100, we have found that the appropriate number of the forecasting trials lies between 40 and 50 for all time series and as such, we fix M to be 50 so that the size of the in-sample weight dataset $\mathbf{w}$ is 200 for all time series. Appropriate size of the validation sets, i.e. N_{vd} is another important parameter. In the present study, we consider that the size of all validation sets to be equal to that of the out-of-sample testing dataset, i.e. $N_{\text{vd}} = N_{\text{ts}}$ for all time series. As there are four individual models in the present study, our suggested neural network for learning the in-sample weight patterns has a $4 \times h \times 4$ architecture. To have a small network size, fast training rate, and effective generalization, we have considered a maximum of four hidden nodes. However, from some prior in-sample experiments, we have found that the proposed ANN provides very good results with 2 or 3 hidden nodes and as such, we fix the value of "h" as 3, i.e. the network structure $4 \times 3 \times 4$ for modeling the in-sample weight datasets of all time series.

Prior to the model fitting phase, the observations of each time series are normalized to lie in the $[0,\ 1]$ interval as follows:

$$y_t^{(\text{new})} = \frac{y_t - y^{(\min)}}{y^{(\max)} - y^{(\min)}} \quad \forall t = 1, 2, \ldots, N_{\text{tr}}. \tag{13}$$

Here, $\mathbf{Y} = [y_1, y_2, \ldots, y_{N_{\text{tr}}}]^{\text{T}}$ is the training dataset and $\mathbf{Y}^{(\text{new})} = \left[y_1^{(\text{new})}, y_2^{(\text{new})}, \ldots, y_{N_{\text{tr}}}^{(\text{new})}\right]^{\text{T}}$ is its corresponding normalized set, $y^{(\min)}$, $y^{(\max)}$ being respectively the minimum and maximum values of $\mathbf{Y}$. The effectiveness of the proposed method is compared with those of the four individual models and five other widely popular linear combination methods, viz. simple average, median, EB, LSR,

Please cite this article as: R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.01.012

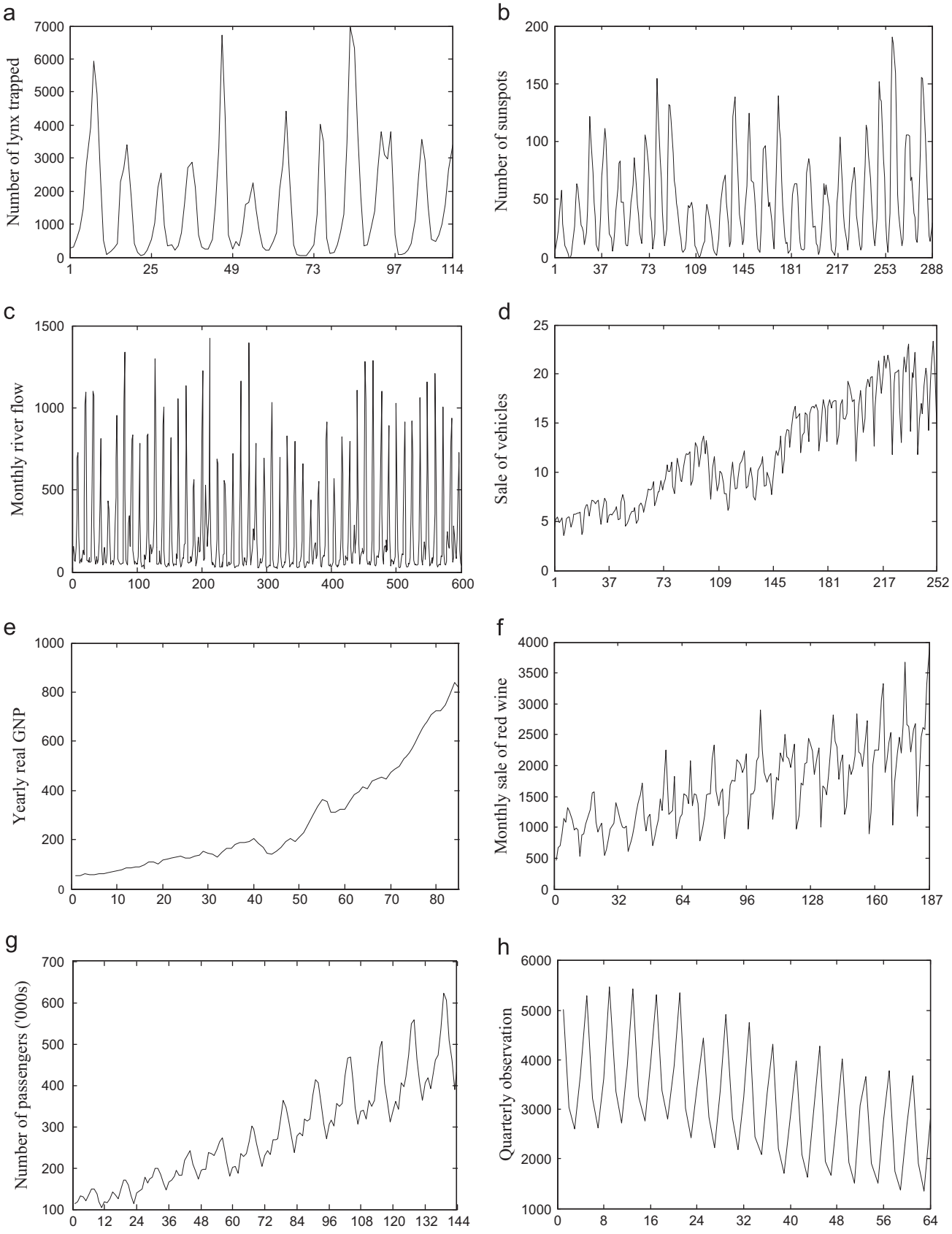

Fig. 4. Time plots of (a) Lynx, (b) Sunspots, (c) River flow, (d) Vehicles, (e) RGNP, (f) Wine, (g) Airline and (h) Industry.

and outperformance. It is in fact a matter of curiosity to check whether ANN really plays a crucial role in recognizing in-sample weight patterns. As such, we also compare the proposed method against the ensemble, termed as the *Average of In-sample Weights (AIW)* scheme that computes the final weights simply as the normalized average of all the in-sample weights. So, in AIW, the

Please cite this article as: R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.01.012

Table 1
Description of the eight time series datasets.

Series	Type	Total size	Testing size	Box–Jenkins	FANN	EANN
Lynx	Stationary, nonseasonal	114	14	ARIMA $(12,0,0)$	$7\times5\times1$	$7\times15\times1$
Sunspots	Stationary, nonseasonal	288	35	ARIMA $(9,0,0)$	$11\times9\times1$	$11\times15\times1$
River flow	Stationary, nonseasonal	600	100	ARIMA $(16,0,0)$	$8\times6\times1$	$8\times15\times1$
Vehicles	Nonstationary, nonseasonal	252	52	ARIMA $(14,1,13)$	$12\times9\times1$	$12\times15\times1$
RGNP	Nonstationary, nonseasonal	85	15	ARIMA $(0,1,0)$	$4\times10\times1$	$4\times15\times1$
Wine	Monthly seasonal	187	55	SARIMA $(0,1,1)\times(0,1,1)^{12}$	$12\times3\times1$	$12\times15\times1$
Airline	Monthly seasonal	144	12	SARIMA $(0,1,1)\times(0,1,1)^{12}$	$12\times2\times1$	$12\times15\times1$
Industry	Quarterly seasonal	64	16	SARIMA $(0,1,1)\times(0,1,1)^{4}$	$4\times3\times1$	$4\times15\times1$

Table 2
The obtained forecasting results of all methods.

Error measures		Individual models				Combination methods						
		Box-Jenkins	SVM	FANN	EANN	Avg.	Median	EB	LSR	Outperf.	AIW	Proposed
Lynx	MAE	0.103	0.173	0.154	0.156	0.112	0.133	0.097	0.133	0.107	0.110	**0.068**
	MSE	0.015	0.053	0.032	0.036	0.018	0.026	0.013	0.027	0.015	0.017	**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**
	MSE	348.5	630.3	526.4	663.6	371.8	428.4	352.3	393.4	375.4	371.8	**311**
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**
	MSE	2.606	1.172	1.217	2.19	1.158	1.138	1.197	1.245	1.068	1.156	**0.978**
Vehicles	MAE	2.174	2.239	2.128	2.173	2.087	2.139	2.06	2.059	2.011	2.071	**2.001**
	MSE	7.142	6.794	7.018	6.989	6.188	6.683	6.011	7.508	6.088	6.175	**5.531**
RGNP	MAE	17.01	20.89	16.2	15	11.74	13.67	10.1	12.25	13.23	11.72	**9.903**
	MSE	470	719.2	359.4	322.2	197.3	265.9	155.2	278.5	260.6	196.6	**139**
Wine	MAE	2.776	2.451	2.923	3.009	2.075	2.173	2.057	2.466	2.008	2.372	**1.923**
	MSE	12.19	10.03	17	18.2	9.233	10.05	9.014	10.07	7.712	9.204	**7.524**
Airline	MAE	12.49	10.85	16.17	15.24	11.63	11.73	10.85	10.68	10.85	10.22	**7.434**
	MSE	291	176.9	378	332.5	181.4	176.5	157.8	158.3	152.5	143.1	**86.63**
Industry	MAE	1.526	1.958	1.388	1.822	1.386	1.37	1.678	1.365	1.452	1.386	**1.272**
	MSE	4.842	5.532	3.162	4.606	2.888	2.869	5.239	2.974	3.764	2.892	**2.576**

combining weights are obtained as $w_i=(1/M)\sum_{j=1}^{M} w_i^j(\forall i=1,2,\ldots,n)$. The forecasting accuracies of all methods are evaluated in terms of MAE and MSE. Two error measures are used to have a fair idea about the performances of different methods. Table 2 presents the forecasting results of all methods, where the results of the proposed ensemble method are shown in bold numbers. In particular, the results of the River flow, Wine, and Industry time series are given in transformed scales, so that original MAE=obtained MAE $\times 10^2$, original MSE=obtained MSE $\times 10^4$.

From Table 2, we can clearly observe that for each time series the proposed combination method achieves the least errors and hence the best accuracies in terms of both the error measures. The results of the individual models vary considerably and no single model alone could achieve the best results for all datasets. It can also be observed that the combination methods have overall better forecasting accuracies than the individual models. Furthermore, Table 2 depicts that the errors of the proposed method are reasonably less than those of the AIW scheme. This important finding empirically justifies the crucial role of ANN in recognizing the in-sample weight patterns. A diagrammatic depiction of the forecasting performance of a model usually consists of a graph of the actual and forecasted observations, plotted against the time indices. We use the terminology *forecast diagram* to refer such a graph. In Fig. 5, we show the eight forecast diagrams of our proposed combination method, where in each plot, the actual and forecasted datasets are respectively represented through the solid and dotted lines. The remarkable closeness between the actual observations and their forecasts is clearly evident in each plot of Fig. 5.

To have an idea about the relative performances of all forecasting methods, we calculate their mean ranks. A mean rank is a popular measure that indicates the average of the ranks of a forecasting method across all time series datasets. Suppose that we have in total k time series and n forecasting methods, considering all individual as well as combination methods. On a particular time series, all methods are consecutively ranked according to their attained forecasting errors, so that the best and worst methods are assigned the ranks 1 and n, respectively. Ties are resolved by assigning the corresponding methods the average of their ranks they would have received if there were no ties. If the jth method has the rank r_{ij} on the ith time series, then its mean rank R_j is the average of these r_{ij} values, over all $i=1,2,\ldots,k$. The mean ranks in terms of MAE and MSE of all methods used in this paper are presented in the second and third columns of Table 3. It can be seen that for both error measures, the top seven ranks are achieved by the seven forecast combination methods, which is as expected. Also, we can see that in both cases, the proposed combination method achieved the best mean ranks.

In the present study, we introduce another measure that is termed as the worth value of a forecasting method. It is calculated as the average percentage reductions in the forecasting errors of

Please cite this article as: R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.01.012

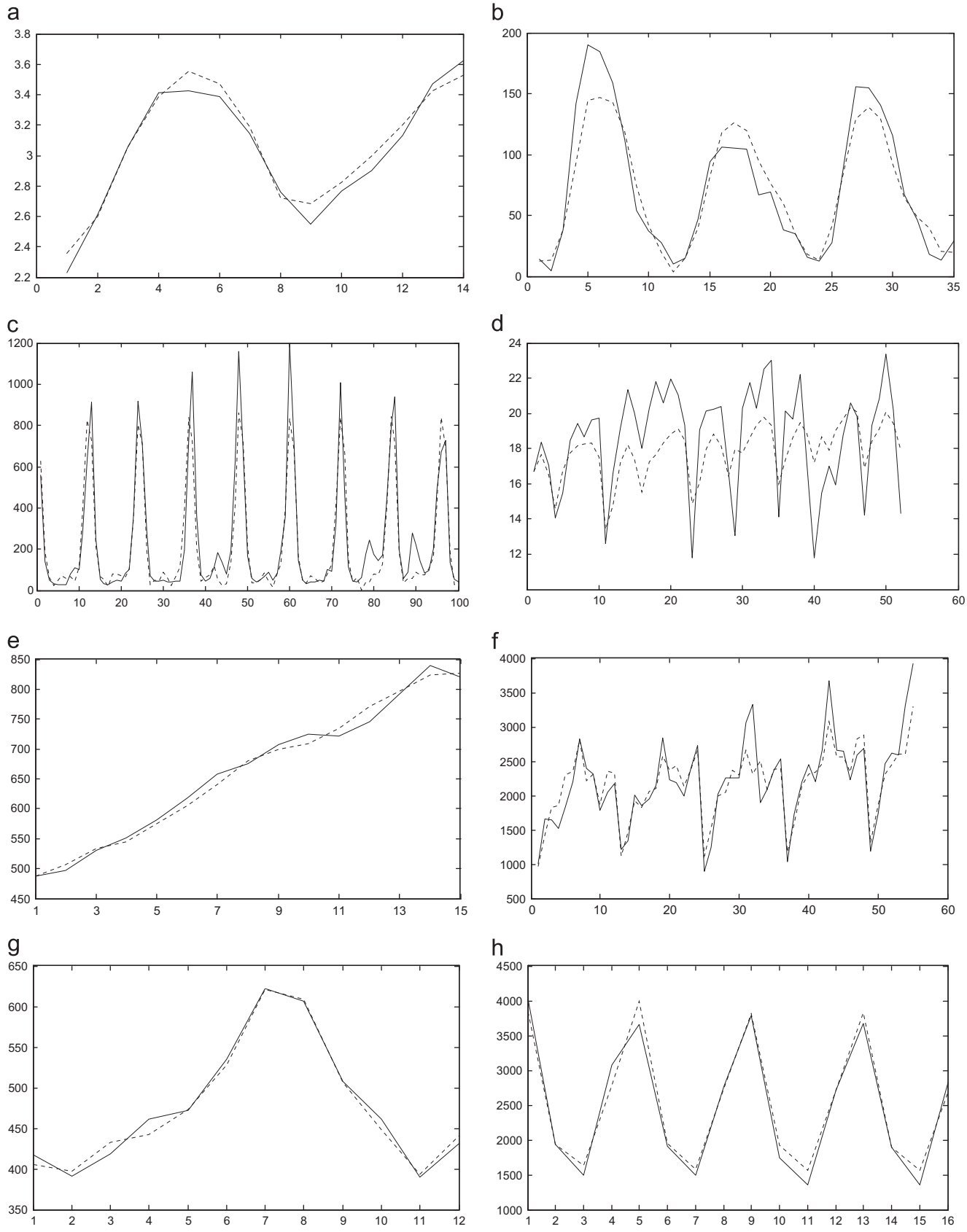

Fig. 5. Forecast diagrams of (a) Lynx, (b) Sunspots, (c) River flow, (d) Vehicles, (e) RGNP, (f) Wine, (g) Airline and (h) Industry.

the worst individual models by a particular method over all datasets. This measure shows the extent to which a forecasting method performs better than the worst individual model and as such represents the overall worthiness of the method. Let, e_{ij} be the forecasting error, attained by the jth method for the ith time series and e_i^w be the maximum error of an individual model for

Please cite this article as: R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.01.012

this time series. Then, the worth value W_j of the jth forecasting method is defined as follows:

$$W_j = \frac{1}{k}\sum_{i=1}^{k}\left[\left(\frac{e_i^w - e_{ij}}{e_i^w}\right)\times 100\right]$$

$$\forall j = 1, 2, \ldots, n. \tag{14}$$

The term inside the summation in (14) measures the percentage error reduction of the jth method, compared to the worst individual model. The worth values of all the methods in terms of the two error statistics are represented in the last two columns of Table 3. From this table, we can see that the proposed combination method has the best worth values, which are around 41% and 62% for MAE and MSE, respectively. This means that in the present scenario with four component models, our combination method can provide about 41% and 62% more accurate results than a worst individual model in terms of MAE and MSE, respectively. A selected individual model has enough possibility to be actually the worst one and as such, a combination method with a relatively high worth value is definitely fruitful to a particular forecasting problem.

We have further carried out the *Friedman's test* to check for the significant differences among the mean ranks of the forecasting methods. It is a non-parametric statistical test that operates on all the mean ranks R_j and evaluates the *null hypothesis* H_0 that the forecasting methods do not differ significantly in terms of their mean ranks [51]. If H_0 is rejected with a certain confidence level α, then we accept the *alternative hypothesis* H_1 that indicates significant differences among the mean ranks. H_0 is tested under the following statistic:

$$\chi_F^2 = \frac{12k}{n(n+1)}\left(\sum_{j=1}^{n} R_j^2 - \frac{n(n+1)^2}{4}\right) \tag{15}$$

The Friedman's test statistic, defined in (15) follows the χ^2 distribution with $(n-1)$ degrees of freedom. In the present study, there are in total 11 forecasting methods and so 10 degrees of freedom. The conducted Friedman's test results at 95% confidence level are as follows:

- For MAE: $\chi_F^2 = 48.79$, $p = 4.45 \times 10^{-7}$.
- For MSE: $\chi_F^2 = 52.41$, $p = 9.57 \times 10^{-8}$.

Here, p represents the probability that the null hypothesis H_0 is true. Due to the very small values of p, we reject the null hypothesis in both cases with $\alpha = 0.95$. So, the conclusion of the test is that the mean ranks of the forecasting methods do significantly differ in terms of both the error measures with a 95% confidence level. The Friedman's test results are diagrammatically depicted in Fig. 6. In each subfigure, a circle represents the mean rank of a method and the horizontal bar across a circle is the critical difference. Two methods have significantly dissimilar performances if their mean ranks differ by at least the critical difference, i.e. if their horizontal bars are non-overlapping. Fig. 6 clearly shows that there is no significant differences among the mean ranks of the individual models, in terms of both the error measures. However, Fig. 6a shows that our combination method has significantly outperformed all the four component models, in term of MAE. Similarly, Fig. 6b shows that the proposed method has outperformed all component models as well as the LSR combination scheme.

Table 3
The mean ranks and worth values of the forecasting methods.

Methods	Mean ranks		Worth values (%)	
	MAE	MSE	MAE	MSE
Box–Jenkins	8.25	7.9375	17.522	27.793
SVM	9	8.625	12.115	20.376
FANN	7.875	8.875	16.935	26.85
EANN	9.75	9.5	9.635	16.77
Simple average	6.0625	5.0625	29.377	50.072
Median	6.1875	5.625	27.387	44.8
EB	4.25	4.125	32.152	48.091
LSR	5.0625	7.5	28.2	43.398
Outperformance	3.875	3.9375	31.961	50.239
AIW	4.6875	3.8125	30.82	51.629
Proposed	1	1	41.399	62.086

6. Conclusions

Time series analysis and forecasting is a dynamic research area, having fundamental importance in numerous practical fields. Improving accuracy of time series forecasts is a challenging task that has been gaining continuous research attentions from past few decades. Extensive works have been performed on combining forecasts from several time series models with the general conclusion that this practice improves the forecasting accuracy to a large extent. Moreover, a forecast combination method often beats all the component models in terms of forecasting precision.

In this paper, we have proposed a forecast combination approach that effectively determines the combining weights through recognizing

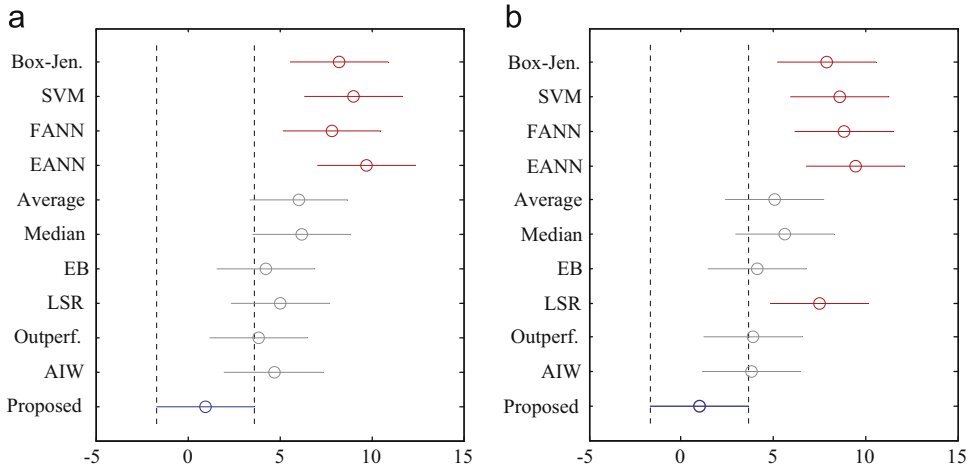

Fig. 6. Friedman's test results of the forecasting methods in terms of: (a) MAE and (b) MSE.

the intrinsic relationship among various groups of in-sample weights. The proposed method at first applies the individual models to a set of consecutive in-sample forecasting trials, where each trial is composed of a pair of training and validation subsets. The component models are assigned weights on each forecasting trial, based on their attained absolute forecasting errors, in an inverse proportionate way. Then, a feedforward neural network structure is designed to identify and learn the pattern in these in-sample weights of different individual models. Finally, the desired combining weights are predicted from this fitted neural network. The empirical study with four models and eight real-world time series clearly shows that the proposed combination method has achieved significantly better accuracies than each component model. Our approach has also consistently outperformed various other widely popular forecast combination schemes. This finding is further justified through two ranking methods and a non-parametric statistical test. The proposed method can also be explored for other large collections of time series and for an extended set of component models. Moreover, other variety of neural networks can be utilized as well, for recognizing the in-sample weight patterns.

Acknowledgements

The author is very much thankful to the anonymous reviewers for their constructive suggestions which significantly facilitated the improvement of this paper. In addition, the author also expresses his profound gratitude to the Council of Scientific and Industrial Research (CSIR), India, for the obtained financial support that provided a great help in performing the present research work.

References

[1] J.M. Bates, C.W.J. Granger, The combination of forecasts, Oper. Res. Q. 20 (4) (1969) 451–468.
[2] R.R. Andrawis, A.F. Atiya, H. El-Shishiny, Forecast combinations of computational intelligence and linear models for the NN5 time series forecasting competition, Int. J. Forecast. 27 (3) (2011) 672–688.
[3] M. Hibon, T. Evgeniou, To combine or not to combine: selecting among forecasts and their combinations, Int. J. Forecast. 21 (1) (2005) 15–24.
[4] I.A. Gheyas, L.S. Smith, A novel neural network ensemble architecture for time series forecasting, Neurocomputing 74 (18) (2011) 3855–3864.
[5] J.S. Armstrong, Principles of Forecasting: A Handbook for Researchers and Practitioners, vol. 30, Kluwer Academic Publishers, Boston, USA, 2001.
[6] S. Makridakis, M. Hibon, The M3-Competition: results, conclusions and implications, Int. J. Forecast. 16 (4) (2000) 451–476.
[7] A. Timmermann, Forecast combinations, Handb. Econ. Forecast. 1 (2006) 135–196.
[8] V.R.R. Jose, R.L. Winkler, Simple robust averages of forecasts: some empirical results, Int. J. Forecast. 24 (1) (2008) 163–169.
[9] J.G. De Gooijer, R.J. Hyndman, 25 years of time series forecasting, Int. J. Forecast. 22 (3) (2006) 443–473.
[10] L.M. De Menezes, D. W Bunn, J.W. Taylor, Review of guidelines for the use of combined forecasts, Eur. J. Oper. Res. 120 (1) (2000) 190–204.
[11] C.W.J. Granger, R. Ramanathan, Improved methods of combining forecasts, J. Forecast. 3 (2) (1984) 197–204.
[12] C. Aksu, S.I. Gunter, An empirical analysis of the accuracy of SA, ERLS and NRLS combination forecasts, Int. J. Forecast. 8 (1) (1992) 27–43.
[13] J. Dickinson, Some comments on the combination of forecasts, Oper. Res. Q. (1975) 205–210.
[14] D.W. Bunn, A Bayesian approach to the linear combination of forecasts, Oper. Res. Q. (1975) 325–329.
[15] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, Adv. Neural Inf. Process. Syst. 20 (1995) 231–238.
[16] Z.-H. Zhou, J. Wu, W. Tang, Ensembling neural networks: many could be better than all, Artif. Intell. 137 (1) (2002) 239–263.
[17] R.T. Clemen, Combining forecasts: a review and annotated bibliography, Int. J. Forecast. 5 (4) (1989) 559–583.
[18] C. Lemke, B. Gabrys, Meta-learning for time series forecasting and forecast combination, Neurocomputing 73 (10) (2010) 2006–2016.
[19] P.S. Freitas, A.J. Rodrigues, Model combination in neural-based forecasting, Eur. J. Oper. Res. 173 (3) (2006) 801–814.
[20] D. Pollock, Recursive estimation in econometrics, Comput. Stat. Data Anal. 44 (1) (2003) 37–75.
[21] S. French, Linear combination of forecasts—a comment, J. Oper. Res. Soc. 32 (1981) 937–938.
[22] S. Makridakis, R.L. Winkler, Averages of forecasts: some empirical results, Manag. Sci. 29 (9) (1983) 987–996.
[23] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, Time Series Analysis: Forecasting and Control, Prentice-Hall, Englewood Cliffs, USA, 1994.
[24] G.E.P. Box, G.M. Jenkins, Time Series Analysis: Forecasting and Control, 3rd edition, Holden-Day, California, 1970.
[25] G. Zhang, B. Eddy Patuwo, M. Y Hu, Forecasting with artificial neural networks: the state of the art, Int. J. Forecast. 14 (1) (1998) 35–62.
[26] G.P. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing 50 (2003) 159–175.
[27] J.L. Elman, Finding structure in time, Cognit. Sci. 14 (2) (1990) 179–211.
[28] C.P. Lim, W.Y. Goh, The application of an ensemble of boosted Elman networks to time series prediction: a benchmark study, Int. J. Comput. Intell. 3 (2) (2005) 119–126.
[29] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.
[30] J.A.K. Suykens, J. Vandewalle, Least squares support vector machines classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[31] C. Hamzaçebi, Improving artificial neural networks' performance in seasonal time series forecasting, Inf. Sci. 178 (23) (2008) 4550–4559.
[32] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[33] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6188) (1986) 533–536.
[34] R. Adhikari, R.K. Agrawal, Forecasting strong seasonal time series with artificial neural networks, J. Sci. Ind. Res. 71 (10) (2012) 657–666.
[35] R. Adhikari, R.K. Agrawal, Performance evaluation of weights selection schemes for linear combination of multiple forecasts, Artif. Intell. Rev. (2012) 1–20. http://dx.doi.org/10.1007/s10462-012-9361-z.
[36] 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.
[37] H. Demuth, M. Beale, M. Hagan, Neural Network Toolbox User's Guide, The MathWorks, Natic, MA, 2010.
[38] S. Kumar, Neural Networks: A Classroom Approach, Tata McGraw-Hill Education, India, 2004.
[39] T. Farooq, A. Guergachi, S. Krishnan, Chaotic time series prediction using knowledge based Greens kernel and least-squares support vector machines, in: IEEE International Conference on Systems, Man and Cybernetics (ISIC), 2007, IEEE, Montreal, Canada, 2007, pp. 373–378.
[40] O. Chapelle, Support Vector Machines: Introduction Principles, Adaptive Tuning and Prior Knowledge (Ph.D. thesis), University of Paris, France, 2002.
[41] G.P. Zhang, A neural network ensemble method with jittered training data for time series forecasting, Inf. Sci. 177 (23) (2007) 5329–5346.
[42] I.S. Markham, T.R. Rakes, The effect of sample size and variability of data on the comparative performance of artificial neural networks and regression, Comput. Oper. Res. 25 (4) (1998) 251–263.
[43] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discov. 2 (2) (1998) 121–167.
[44] J.H. Stock, M.W. Watson, Combination forecasts of output growth in a seven-country data set, J. Forecast. 23 (6) (2004) 405–430.
[45] G.H. Golub, C.F. Van Loan, Matrix computations, 3rd edition, vol. 3, The John Hopkins University Press, Baltimore, USA, 2012.
[46] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, John Wiley & Sons, Hoboken, New Jersey, 2004.
[47] M. Anthony, Probabilistic analysis of learning in artificial neural networks: the PAC model and its variants, Neural Comput. Surv. 1 (1998) 1–60.
[48] R.J. Hyndman, Time Series Data Library (TSDL), 2013, Available online at url: ⟨http://robjhyndman.com/TSDL/⟩(01-01-13).
[49] D.A. Dickey, W.A. Fuller, Distribution of the estimators for autoregressive time series with a unit root, J. Am. stat. Assoc. 74 (366a) (1979) 427–431.
[50] Delft Center for Systems and Control, 2013, MATLAB Toolbox ARMASA, [Online], url: ⟨http://www.dcsc.tudelft.nl/Research/Software⟩(12-07-13).
[51] M. Hollander, D.A. Wolfe, E. Chicken, Nonparametric Statistical Methods, vol. 751, John Wiley & Sons, Hoboken, New Jersey, 2013.

Ratnadip Adhikari received his B.Sc. degree with Mathematics Honors from Assam University, Silchar, India in 2004 and M.Sc. in applied mathematics from Indian Institute of Technology, Roorkee, India in 2006. After that he obtained M. Tech in computer science and technology and Ph.D. in computer science, both from Jawaharlal Nehru University, New Delhi, India in 2009 and 2014, respectively. He was awarded the best graduate from Assam University in 2004 and subsequently received Gold Medal for this achievement. At present, he is working as an Assistant Professor in the Computer Science & Engineering (CSE) Department of the LNM Institute of Information Technology (LNMIIT), Jaipur, Rajasthan, India. His primary research interests include Pattern recognition, time series forecasting, data stream classification, and hybrid modeling. His research works are published in various reputed international journals and conferences. He has attended a number of conferences and workshops throughout his academic career.