Neural Process Lett (2016) 44:831–856
DOI 10.1007/s11063-016-9499-9

# Selected an Stacking ELMs for Time Series Prediction

**Zhongchen Ma**[1] · **Qun Dai**[1]

Published online: 12 January 2016


**Abstract** Extreme learning machine (ELM) has several interesting and significant features. In this paper, a novel pruned Stacking ELMs (PS-ELMs) algorithm for time series prediction (TSP) is proposed. It employs ELM as the level-0 algorithm to train several models for Stacking. And our previously proposed reduce-error pruning for TSP (ReTSP)-Trend pruning technique is used to solve the problem that the level-0 learners might make many correlated error predictions. ReTSP-Trend refers to an evaluation measure for reduce-error pruning for TSP (ReTSP), which takes into account the time series trend and the forecasting error direction. What's more, ELM and simple averaging are used to generate the level-1 model. With the development of PS-ELMs, firstly, those essential advantages of ELM will be naturally inherited. Secondly, those specific defects of ELM are ameliorated to some extent, with the help of ensemble pruning paradigm. Thirdly, ensemble pruning is employed to raise the robustness and accuracy of time series forecasting, making up for the shortages of the existing research. Fourthly, our previously proposed pruning measure ReTSP-Trend is employed in PS-ELMs, which indeed guarantees that the remaining predictor which supplements the subensemble the most will be selected. And finally, the development of PS-ELMs will promote our investigation to the popular ensemble technique of Stacked Generalization. The experimental results on four benchmark financial time series datasets verified the validity of the proposed PS-ELMs algorithm.

**Keywords** Extreme learning machine (ELM) · Stacked Generalization (Stacking) · Reduce-error pruning for time series prediction (ReTSP) · ReTSP-Trend · Financial time series forecasting · Pruned Stacking extreme learning machines (PS-ELMs)

✉ Qun Dai
daiqun@nuaa.edu.cn

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

## 1 Introduction

Time series prediction (TSP) is a significant active research topic in machine learning and data engineering, and it has indispensable importance in many practical data mining applications. In general, time series involves a subject of research interest in various areas of knowledge such as: economy (stock prices, unemployment rate, and industrial production), epidemiology (rate of cases of an infectious diseases), medicine (electrocardiogram, and electroencephalogram), and meteorology (temperature, wind velocity and pluviometric precipitation) [1]. Financial time series forecasting is one of the most active areas in TSP. A major challenge confronted with speculators, investors and businesses is how to accurately forecast price movements in financial and commodity markets [2]. While many factors might influence the trend of a stock market, including political events, general economic conditions, and trader's expectations [2], and consequently, it is a challenging task to predict the stock market trend, due to its high volatility and noisy environment.

Statistical linear methods dominated in TSP for years. However, on the grounds that some financial time series contain several specific characteristics: large sample sizes, high noise, non-stationary, non-linearity, and varying associated risk [3], it might be obvious that there will be no single statistical linear approach which could perform the best for all the time series modeling tasks. In the last decades, neural network (NN) methods have attracted significant attention for TSP problems in the reason of their theoretical properties of non-parametric, data driven universal approximation of any linear or nonlinear function [4]. Feed-forward NNs (FNNs) were the first models used to detect regularities in the stock market [5]. Since then, many kinds of artificial NNs have been utilized to predict the movement of the stock market [6].

Traditionally, all the parameters of FNNs need to be tuned, and thus there exists dependency between different layers of parameters (weights and biases). Gradient descent-based methods have mainly been used in various learning algorithms of FNNs. However, these gradient descent-based methods are generally very slow and may easily converge to local minima. In recent years, a new learning algorithm called extreme learning machine (ELM) for single-hidden layer FNNs (SLFNs) which randomly chooses hidden nodes and analytically determines the output weights of SLFNs has been proposed. Some simulation results on artificial and real large applications in [7] have showed that this algorithm tends to provide good generalization performance at extremely fast learning speed. In [8], the authors have implemented many simulations and found that ELM has several interesting and significant features different from traditional popular gradient-based learning algorithms for FNNs:

(1) The learning speed of ELM is extremely fast. In their simulations, the learning phase of ELM can be completed in seconds or less than seconds for many applications;
(2) ELM could often obtain a better generalization performance than gradient-based learning algorithms;
(3) ELM learning algorithm tends to reach the solutions straightforward without facing several issues like local minima, etc. It looks much simpler than most learning algorithms for FNNs.

With such many good features, ELM has been widely used in a great deal of fields, such as intelligent measurement [9], face recognition [10], production prediction [11], biofuel engine performance prediction [12], wind speed distribution estimation [13], human activity recognition [14], and biomedical [15,16], etc.

However, it is still a problem to select the appropriate number of neurons in the hidden layer of ELM. The ELM algorithm tends to have problems when irrelevant or correlated variables

are present [17]. For this reason, the optimally pruned ELM methodology is proposed in [17] to perform a pruning of the irrelevant variables, via pruning of the related neurons of the SLFN built by ELM. What's more, there also exist other optimization methodologies in artificial NNs system which have been explored in [18,19]. As there are multiple choices of hypotheses that produce dissimilar results, these works propose some frameworks for the optimization of different classifiers aiming at finding the optimal models. While we find that the ensemble learning methods could be used to solve this problem of selecting the proper number of hidden neurons for ELM. Besides choosing a best single predictor, ensemble methods have achieved great success due to that it is more robust and more precise than a single predictor. And, the ensemble method of combining several ELMs has shown good classification performance in [16].

Lots of works [20–23] using ensemble methods have achieved great success on financial time series forecasting. While, an important shortcoming of ensemble methods is that, in many problems of practical interest, many constituent predictors are needed for the ensemble to achieve good generalization performance. However, larger ensemble requires more storage spaces and takes longer to make predictions [24]. Ensemble pruning is a competitive approach to alleviate the above problems of traditional ensemble methods. Moreover, there exists another benefit that, the generalization performance of pruned ensembles may be even better than the original ensemble consisting of all the given individual learners. Ensemble pruning approach has shown to be effective in classification and regression problems. It can achieve better performance than, or nearly the same level of performance as, the entire ensemble in these tasks [25–32].

However, we find that there are few works studying the performance of ensemble pruning technique on time series forecasting problems, whereas it could be foreseen that the technique could perform well on this task. First of all, ensemble pruning technique could be used to enhance the robustness and accuracy of time series forecasting model. Second, in a sense, time series forecasting is similar to regression problems, and according to the certification given by Zhou et al. [33], it can be concluded that pruned ensemble could be better than the original entire ensemble in regression and time series forecasting tasks.

In one of our previous works [34], we proposed several novel evaluation measures for rank-based ensemble pruning with applications to TSP. The first evaluation measure is complementarity measure for TSP (ComTSP), which aims to incorporate at each selection step the base learner whose performance is most complementary to that of the current subensemble. The second one is concurrency thinning for TSP (ConTSP), which is designed based on the performance of both the subensemble and the candidate learner with regard to a selection set. With the measure ConTSP, a candidate learner is rewarded for obtaining a correct prediction, and rewarded more for obtaining a correct prediction when the subensemble is incorrect. A candidate learner is penalized in the event where both the subensemble and the candidate learner are incorrect.

ReTSP-Value and ReTSP-Trend for reduce-error pruning for TSP (ReTSP) are the third and fourth evaluation measures proposed by us [34]. ReTSP-Value and ReTSP-Trend are based on the reduce error (RE) pruning technique, which incorporates one learner into the subensemble at each selection step by estimating its prediction error on selection dataset. However, ReTSP-Value has the same weakness as ComTSP that, it could not guarantee the remaining predictor which supplements the subensemble the most will be selected. The reason of this weakness is that the predictive error in TSP is directional. It is not very reasonable for the measures to take reducing error as the only objective, while neglect the error direction. Our proposed measure ReTSP-Trend overcomes this weakness, taking into consideration

the time series trend and the forecasting error direction. It could indeed guarantee that the remaining predictor which supplements the subensemble the most will be selected [34].

Moreover, ensembles can be integrated by combining the base models outputs in some fashion. One mechanism of ensemble integration is to use a meta-learning model to combine the outputs of the base models. One of the best known meta-techniques is that of Stacking or Stacked Generalization. Stacking is one of the earliest hierarchical methods of ensemble learning. However, it is a critical issue of selecting the right learners, their parameters and meta-learners. And it has been found that a necessary condition to create a good ensemble of learners is that base-level learners error predictions are uncorrelated [35].

In this work, a pruned Stacking ELMs (PS-ELMs) algorithm is proposed by us for TSP. It uses ELM as level-0 learning algorithm to train a series of models in reason of that the ELM algorithm has a fast learning speed and good generalization. And it has been showed that ELM can obtain a good performance on TSP [36]. And ensemble pruning methods can be used to get rid of those redundant component ELMs in the original ensemble. Specifically, PS-ELMs applies the Stacking method to combine a pruned committee of ELM networks. And the ReTSP-Trend ensemble pruning technique which has proven to be effective in one of our previous work [34], is used to select appreciate ELMs for making up the level-0 committee in order to achieve good generalization performance and less storage spaces. What's more, ELM is also used as the meta-learning algorithm for Stacking in comparison with simple averaging as the meta-learning algorithm.

Based on the above descriptions about the proposed PS-ELMs algorithm, our motivations behind its development, and simultaneously, the novelty and contributions of this work are summarized as follows.

Firstly, PS-ELMs naturally inherits those salient advantages of ELM, including extremely fast learning speed, better generalization capability and the avoidance of local minima problem.

Secondly, with PS-ELMs, we wish to improve those specific defects of ELM to some extent, by the introduction of ensemble pruning paradigm.

Thirdly, in PS-ELMs, we use the ensemble pruning technique to enhance the robustness and accuracy of time series forecasting model. This is the major difference between our work and the related studies [1,20–23]. In their works, they used the traditional ensemble methods for financial time series forecasting, while we increase an intermediate ensemble pruning procedure so as to improve the forecasting performance. Actually, it is found by us that there exist very few works studying the application of ensemble pruning technique on time series forecasting. Thus, we hope that with the development of PS-ELMs, we can facilitate the research of ensemble pruning technique on its applications to TSP.

Fourthly, in PS-ELMs, our previously proposed pruning measure of ReTSP-Trend [34] is employed, which takes into consideration the time series trend and the forecasting error direction, and could indeed guarantee that the remaining predictor which supplements the subensemble the most will be selected.

Finally, PS-ELMs applies the Stacking method to combine a committee of ELM networks, and ELM is used as the meta-learning algorithm for Stacking in comparison with simple averaging as the meta-learning algorithm. Therefore, the development of PS-ELMs will boost our investigation to the popular Stacked Generalization ensemble technique, and we attempt to solve those difficult problems confronted with Stacked Generalization.

Except for the above summarization about the motivations, novelty and contributions of this work, another essential contribution is that, the proposed model PS-ELMs possesses several good features compared with the traditional ensemble methods, such as: it can improve the forecasting accuracy, according to the experimental results. And it also can reduce the

time complexity when making a prediction for an unknown instance, which has been proven by theoretical analysis, while the property of prediction speed is crucial for time series forecasting.

We implement simulation experiments to evaluate the performance of the proposed PS-ELMs algorithm in comparison with choosing the best single model (BSM) in the original ELMs ensemble, and with the Stacking ensemble method without pruning, on four benchmark financial time series datasets. For comparison purposes, support vector regression (SVR) algorithm is also used as the level-0 learning algorithm in Stacking. From the experimental results, it can be found that the proposed PS-ELMs algorithm outperforms its rivals. It can give a best generalization performance, and make a best prediction for the trend of the stock market. Moreover, it could obtain a highest pruning rate which represents that it could make prediction extremely fast. Meantime, it also can be found that the performance of Stacking ELMs without pruning procedure is barely satisfactory. The phenomenon can be attributed to that the pruning procedure could reduce correlated error predictions made by the level-0 models.

This paper is organized as follows. Section 2 introduces some important concepts of the ELM algorithm. Section 3 gives the novel ideas and details of the proposed PS-ELMs algorithm. Section 4 reports and discusses the experimental results. Finally, Sect. 5 summarizes this paper.

## 2 Preview with ELM

The ELM algorithm was originally proposed by Huang et al. [8]. It makes use of the single layer FNN (SLFN), and it has the same topology as SLFN, as shown in Fig. 1. The main concept behind the ELM lies in the random initialization of the SLFN weights and bias. The input weights and biases do not need to be adjusted. And it is possible to calculate explicitly the hidden layer output matrix and hence the output weights. The network is obtained with very few steps and very low computational costs [37].

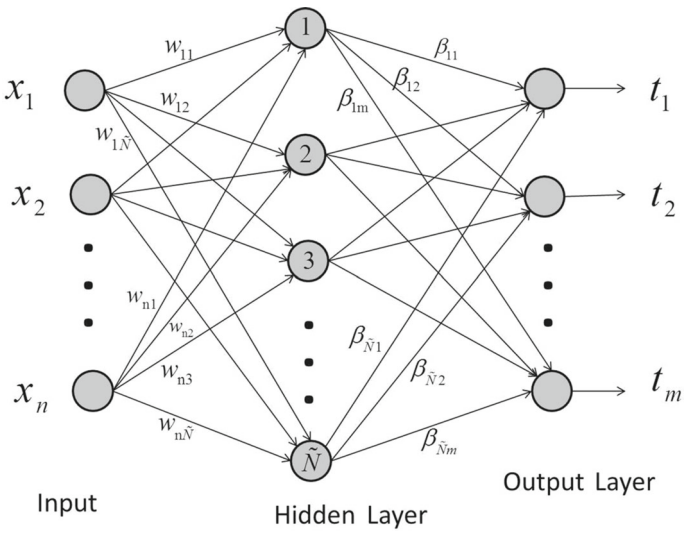


**Fig. 1** ELM network topology

Suppose we have $N$ arbitrary distinct samples $(\mathbf{x}_i,\ \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1},\ x_{i2}, \ldots, x_{in}]^T \in \mathbf{R}^n$ and $\mathbf{t}_i = [t_{i1},\ t_{i2}, \ldots, t_{im}]^T \in \mathbf{R}^m$, standard SLFNs with $\tilde{N}$ hidden nodes with activation function $g(x)$ can be analytically modeled by:

$$\sum_{i=1}^{\tilde{N}} \beta_i g\left(\mathbf{w}_i \cdot \mathbf{x}_j + \mathrm{b}_j\right) = \mathbf{t}_j, \quad j = 1, \ldots, N, \tag{1}$$

where $\mathbf{w}_i = [w_{i1},\ w_{i2}, \ldots, w_{in}]^T$ is the weight vector connecting the $i$th hidden node and the input nodes, $[\beta_{i1},\ \beta_{i2}, \ldots, \beta_{in}]^T$ is the weight vector connecting the $i$th hidden node and the output nodes, and $b_i$ is the bias of the $i$th hidden neuron to the output layer.

Equation (1) can also be written compactly as:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{2}$$

where $\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{N}},\ \mathrm{b}_1, \ldots, \mathrm{b}_{\tilde{N}},\ \mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{bmatrix} g(\mathbf{w}_1 \bullet \mathbf{x}_1 + \mathrm{b}_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \bullet \mathbf{x}_1 + \mathrm{b}_{\tilde{N}}) \\ \vdots & \vdots & \vdots \\ g(\mathbf{w}_1 \bullet \mathbf{x}_N + \mathrm{b}_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \bullet \mathbf{x}_N + \mathrm{b}_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}$,

$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m}$ and $\mathbf{T} = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}$.

Here, $\mathbf{w}_i \bullet \mathbf{x}_j$ denotes the inner product of $\mathbf{w}_i$ and $\mathbf{x}_j$. As named in Huang [38], $\mathbf{H}$ is called the hidden layer output matrix of the NN; the $i$th column of $\mathbf{H}$ is the $i$th hidden node output with respect to inputs $\mathbf{x}_1,\ \mathbf{x}_2, \ldots, \mathbf{x}_N$.

Traditionally, in order to train an SLFN, one may wish to find specific $\hat{\mathbf{w}}_i,\ b_i,\ \hat{\boldsymbol{\beta}}$(i $= 1, \ldots, \tilde{\mathrm{N}}$) to minimizing $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|$. Specifically,

$$\left\| H\left(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_{\tilde{N}},\ \hat{\mathrm{b}}_1, \ldots, \hat{\mathrm{b}}_{\tilde{N}}\right)\hat{\beta} - \mathbf{T}\right\| = \min_{\mathbf{w}_i, \mathrm{b}_i, \beta} \left\| H\left(\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{N}},\ \mathrm{b}_1, \ldots, \mathrm{b}_{\tilde{N}}\right)\beta - \mathbf{T}\right\|, \tag{3}$$

which is equivalent to minimizing the cost function

$$E = \sum_{j=1}^{N} \left( \sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i g\left(\mathbf{w}_i \cdot \mathbf{x}_j + \mathrm{b}_i\right) - \mathbf{t}_j \right)^2. \tag{4}$$

The gradient-decent learning algorithms are generally used to optimize these parameters. It update parameter vector $\mathbf{w}$ iteratively according to:

$$\mathbf{w}_k = \mathbf{w}_{k-1} - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}. \tag{5}$$

Here $\eta$ is learning rate. The popular learning algorithm used in FNNs is the back-propagation (BP) learning algorithm where gradients can be computed efficiently by propagation from the output to the input. However, there are several issues on BP learning algorithms: the way to find an appropriate learning rate $\eta$ is difficult; the BP learning algorithm is the presence of local minima; it may be over-trained and obtain worse generalization performance; gradient-based learning is very time-consuming in most applications.

However, the ELM algorithm resolve the above issues related with gradient-based algorithms. It first chooses input weights $\mathbf{w}_i$ and hidden neuron bias $b_i$ randomly, after which

matrix $\mathbf{H}$ can be calculated immediately. Then the problem of minimizing cost function in Eq. (3) is equivalent to finding a least-squares solution $\hat{\boldsymbol{\beta}}$ of the linear system $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|$:

$$\left\| H\left(\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{N}}, \mathrm{b}_1, \ldots, \mathrm{b}_{\tilde{N}}\right)\hat{\boldsymbol{\beta}} - \mathbf{T} \right\| = \min_{\mathbf{w}_i, \mathrm{b}_i, \beta} \left\| H\left(\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{N}}, \mathrm{b}_1, \ldots, \mathrm{b}_{\tilde{N}}\right)\boldsymbol{\beta} - \mathbf{T} \right\|. \tag{6}$$

If the number $\tilde{N}$ of hidden nodes is equal to the number $N$ of distinct training samples, $\tilde{N} = N$, matrix $\mathbf{H}$ is square and invertible when the input weight vectors $\mathbf{w}_i$ and the hidden biases $b_i$ are randomly chosen, and these training samples can be approximated with zero error. However, in most cases the number of hidden nodes is much less than the number of distinct training samples, $\tilde{N} \ll N$, $\mathbf{H}$ is nonsquare matrix and there may not exist $\mathbf{w}_i$, $b_i$, $\beta_i (i = 1, \ldots, \tilde{N})$ such that $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$. ELM algorithm learns the output weights $\boldsymbol{\beta}$ with the use of a Moore–Penrose generalized inverse of the matrix $\mathbf{H}$, denoted as $\mathbf{H}^{\dagger}$ [38]. The smallest norm least squares solution of the above linear system is

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}. \tag{7}$$

The solution $\hat{\boldsymbol{\beta}}$ defined in Eq. (7) has the norm minimum over all the solutions of the least squares solutions of linear system in Eq. (2). Thus, $\hat{\boldsymbol{\beta}}$ deserves the best generalization performance across all the other least squares solutions [39]. Compared with traditional popular gradient-based learning algorithms for FNNs, the ELM has several interesting and significant features: the learning speed of ELM is extremely fast; the generalization performance of ELM is better than the gradient-based learning algorithms and the ELM learning algorithm tends to reach the solutions straightforward without facing several issues like local minima which the traditional classic gradient-based learning algorithms faced [8].

## 3 The Proposed Pruned Stacking ELMs (PS-ELMs) Algorithm for Time Series Prediction

The ELM algorithm needs much less training time compared to the popular BP algorithm and support vector machine (SVM) or SVR. What's more, the prediction accuracy of basic ELM is usually better than BP and similar to SVM/SVR in many classification and regression applications. However, a single ELM network simply assigns input weights and biases at random, instead of explicit training as in the gradient-decent based network. It also ineluctably brings about uncertainty and susceptibility to errors in the reason of certain stochastic behaviors of ELM. Therefore, we tend to use an ensemble of ELMs instead of selecting a best single ELM for TSP task, as an ensemble is usually superior to a single predictor given the same amount of training information.

Stacking is a widely used ensemble technique for combining learners and improving prediction accuracy. One of the problems of Stacking is how to obtain the right combination of level-0 classifiers and the meta-classifier, specifically in relation to each specific dataset [40]. Typically, Stacking uses an algorithm to learn how to combine the outputs of a set of classifiers obtained with the same learning algorithm [41]. In [42], the authors use probability distributions for the outputs from level-0 models instead of a simple class prediction as level-1 attributes. And the authors proposed to use the multi-response linear regression technique as the level-1 algorithm. In [43], a regression tree as the meta-classifier instead of a linear model was proposed by Ting and Witten. In [40], Agapito Ledezma et al. proposed an evolutionary Stacked Generalization (GA-Stacking) algorithm, an approach to find good Stacking configurations by means of genetic search. GA-Stacking not only determines which level-1, and which (and how many) base classifiers must be present, but also their parameters.

GA-Stacking could find accurate Stacking configurations, and it also provides some automation for the data mining process. However, GA-Stacking requires a longer execution time than the rest of approaches, because several generations of individuals must be evaluated in order to obtain a good individual. Therefore, we are aimed at exploring the performance of the ELM algorithm as the meta-model for Stacking for the many outstanding features that ELM has.

What's more, it has been shown that selecting the right level-0 learners was a critical issue in early researches of Stacking. The ensembles generated by existing techniques are sometimes unnecessarily large, which can lead to extra memory usage, computational costs, and occasional decreases in effectiveness. And a necessary condition to create a good ensemble of learners is that the error predictions made by level-0 learners are uncorrelated [35]. In [44], a variant of Stacking that uses correspondence analysis in order to detect correlations between the level-0 learners was proposed. While in this work, we apply the ensemble pruning technique to search for a good subset of an ELMs ensemble. A well-understood and high-performance pruning strategy for TSP task, i.e., the ReTSP-Trend pruning method, which has been proposed in our previous work [34], is used in this work.

Our thinking behind the construction of PS-ELMs is summarized as below.

First, PS-ELMs will naturally inherit the salient advantages of ELM, including: extremely fast learning speed, better generalization capability and the avoidance of local minima problem.

Second, with PS-ELMs, we hope to ameliorate the specific defects of ELM to some extent, by the incorporation of ensemble pruning paradigm.

Third, we attempt to use the ensemble pruning method to boost the robustness and predictive accuracy of time series forecasting model, whereas it is found by us that there are very few works studying the application of ensemble pruning method on time series forecasting. We wish that with the construction of PS-ELMs, we could advance the application research of ensemble pruning method on TSP.

Fourth, in PS-ELMs, our previously proposed pruning measure ReTSP-Trend is employed [34], which takes into consideration both the time series trend and the forecasting error direction, and could indeed guarantee that the remaining predictor which supplements the subensemble the most will be selected. We hope that, with the help of our proposed ReTSP-Trend, we can further improve the predictive accuracy achieved by our system on time series forecasting problems.

Last, we wish that the construction of PS-ELMs will promote our investigation to the popular ensemble technique of Stacked Generalization. We attempt to solve the difficult problems existed in Stacked Generalization with the help of ELM and our proposed ReTSP-Trend pruning method.

The training procedure of the proposed PS-ELMs algorithm can be summarized with a few steps:

*Step 1* train various of level-0 models using the ELM algorithm;
*Step 2* use the ReTSP-Trend pruning method to select the right learners for Stacking;
*Step 3* use the ELM algorithm as level-1 algorithm to combine the outputs of learners selected in Step 2.

In this section, we first give the basic ideas of Stacked Generalization for time series forecasting. Then, the details of ReTSP-Trend pruning technique are presented. Finally, we introduce the proposed PS-ELMs algorithm for TSP.

### 3.1 Stacked Generalization

Stacking is the abbreviation that refers to Stacked Generalization. Stacking [45] comprises two different stages, namely, level-0 and level-1 stages. Level-0 learners refer to a set of models in the committee, and the original data, called level-0 data, is applied to train each one of the level-0 learners. Once the level-0 learners have been generated, it is required to further combine their predictive results in order to get the final prediction of the ensemble. Stacking uses a meta-learner (or termed level-1 model) to further combine the outputs of the level-0 learners so as to obtain the final prediction. In the following, we use a formalized language to describe it.

Given a data set $L$, it leaves one of the subsets out (e.g., $L_j$) to be used later. The remaining instances $L(-\mathrm{j}) = L - L_j$ are used to generate the level-0 classifiers by applying **K** different learning algorithms, k =1,…,**K**, to obtain **K** learners. After the level-0 models have been generated, the $L_j$ set is used to train the level-1 learner. Level-1 training data is created from the predictions of the level-0 models over the instances in $L_j$, that were left out for this purpose. Level-1 data has **K** attributes, whose values are the predictions of each one of the **K** level-0 learners for every instance in $L_j$. Therefore, a level-1 training example is made of **K** attributes (the **K** predictions) and the target value, which is the right prediction for every instance in $L_j$. Once the level-1 data has been built from all instances in $L_j$, any learning algorithm can be used to generate the level-1 learner. To predict a new instance, the level-0 models produce a vector of predictions that is the input to the level-1 model, which in turn predicts the value of target variable.

### 3.2 ReTSP-Trend Pruning Technique

The objective of investigation about ensemble pruning methods is to design a procedure that can select the subensemble with the lowest generalization error, which has been proven to be a NP-complete problem [46]. In order to simplify the search in the space of subensembles, it is assumed that the best subensemble of size $u-1$ is included in the best subensemble of size $u$ for ranking-based pruning methods, and we can construct a sequence of best subensembles of increasing size by including one learner a time [27]. The learner which should be incorporated into the subensemble has to be decided according to an evaluation measure for ranking-based ensemble pruning methods. While it is too simplistic to using the predictive performance of individual models as evaluation measure, which can not achieve satisfying results [47]. In [48], the authors have shown in their works that neither the accuracy of the base learners nor their diversity are by themselves sufficient to identify effective ensembles. A good ensemble evaluation measure needs to take into account both accuracy and diversity. In order to identify subensembles with a good generalization performance, it is necessary to take into account the complementarity among the learners [49].

Before describing the ReTSP-Trend evaluation measure in detail, it is useful to introduce some notations. The input of the learning algorithm consists in a set of instances $Z_{train} = \{(\mathbf{x}_i, y_i),\ i = 1, \ldots, N_{train}\}$. Each instance is characterized by a feature vector $\mathbf{x}_i$, and a target value $y_i$. The objective of the learning algorithm is to induce from the training dataset $Z_{train}$ a hypothesis $h(\mathbf{x})$ that predicts the predicted value of a new example characterized by the vector of attributes $\mathbf{x}$.

The evaluation measures for rank-based pruning technique make use of a selection set composed of $N_{sel}$ labeled examples, i.e., $Z_{sel} = \{(\mathbf{x}_i, y_i),\ i = 1, \ldots, N_{sel}\}$, to guide the order of aggregation. Base learners that are expected to perform best when combined are

aggregated first. From the subensemble $S_{u-1}$ of size $u-1$, the subensemble $S_u$ of size $u$ is constructed by incorporating a single learner selected from the remaining models in the original ensemble which are not included in $S_{u-1}$.

The ReTSP-Trend pruning procedure is simple in theory. An in-depth analysis for this method has been presented in our previous work [34], along with which the detail of ReTSP-Trend pruning technique is given as below.

Given a selection set $Z_{sel}$ of size $N_{sel}$, the signature vector $\mathbf{c}^{(t)}$ of predictor $t$ is defined as the $N_{sel}$ dimensional vector whose $i$th component is

$$\mathbf{c}_i^{(t)} = \left(h_t\left(\mathbf{x}_i\right) - y_i\right), \quad \left(\mathbf{x}_i,\ y_i\right) \in Z_{sel}. \tag{8}$$

The $i$th component $\mathbf{c}_i^{(t)}$ is equal to 0 if the forecasting value of the $t$th predictor exactly matches to the $i$th example in $Z_{sel}$, and it denotes the squared error made by the $t$th predictor for the $i$th example in $Z_{sel}$. The ensemble signature vector $\mathbf{c}_{ens}$ is defined as the sum of the signature vectors of all the predictors in the ensemble. And the average ensemble signature vector is defined as:

$$\langle \mathbf{c} \rangle = T^{-1} \sum_{t=1}^{T} \mathbf{c}^{(\mathrm{t})}. \tag{9}$$

In TSP, the $i$th component of $\langle \mathbf{c} \rangle$ is the margin of the $i$th example, defined as the accuracy of the ensemble prediction for this example. The $i$th example is fitted better by the ensemble if the $i$th component of $\langle \mathbf{c} \rangle$ is smaller. In consequence, the objective is to select a subensemble whose average signature vector is as close as possible to the origin $\mathbf{o}$ of coordinates.

This method starts with selecting into the empty subensemble the individual learner whose predictive performance on the selection set is the best. Then, the remaining individual learners are sequentially put into the subensemble, such that the Euclidean distance from vector $\langle \mathbf{c} \rangle$ to $\mathbf{o}$ is as small as possible in each selection round. More specifically, the first predictor that is incorporated into the subensemble is the one that reduces the distance from vector $\langle \mathbf{c} \rangle$ to $\mathbf{o}$ the most. In particular, the predictor selected in the $u$th selection step is indexed with:

$$s_u = \arg\min_k d\left(\mathbf{o},\ T^{-1}\left(\mathbf{c}^{(\mathrm{k})} + \sum_{t=1}^{u-1} \mathbf{c}^{(\mathrm{t})}\right)\right), \tag{10}$$

where $h_k \in E_T \backslash S_{u-1}$, $d(u,\ v)$ denotes the usual Euclidean distance between points $u$ and $v$, and $s_u$ represents the serial number of the currently selected base learner in the original ensemble.

We use Algorithm 1 to show the procedure of ReTSP-Trend pruning method. And Fig. 2 is used to illustrate the diagram of this selection procedure. Assume $T$ models are trained in the ensemble, the time complexity of training the ensemble without pruning procedure and of making a prediction for an unknown instance respectively are $O(T \cdot t_{train})$ and $O(T \cdot t_{test})$, where $t_{train}$ depends on the specific training algorithm and the size of training set, and $t_{test}$ depends on the specific base models utilized in the ensemble. When the ReTSP-Trend pruning procedure is applied, the time complexity of training and pruning the ensemble equals to $O(T \cdot t_{train} + u \cdot T)$, while the time complexity of making a prediction for an unknown instance is $O(u \cdot t_{test})$, where $u$ is the size of ensemble obtained after pruning and $u \ll T$. From the theoretical analysis, we can see that the prediction complexity is reduced largely and the prediction complexity means a lot for a time series forecasting system.

**Algorithm 1** The pseudo code of the ReTSP-Trend pruning procedure

**Input**: $Z_{sel} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N_{sel}\}$ : pruning dataset;

$u$ : size of feasible subensemble;

$E_T$ : the original ensemble $E_T \equiv \{h_i(\mathrm{x})\}_{i=1}^{T}$;

**Output**: Best subensemble for prediction $S_u$.

1: Select the learner $h$ which has the best accuracy on the pruning dataset.

2: $S_u = \{h\}$

3: **while** $\|S\| \le u$ //Comments: $\|S\|$ denotes the size of the subensemble $S_u$

4: Select the learner $h_k \in E_T \setminus S_u$ which satisfies Eq.(10).

5: $S_u = \{S_u, h_k\}$

6: **End while**

7: **Return** Best subensemble $S_u$

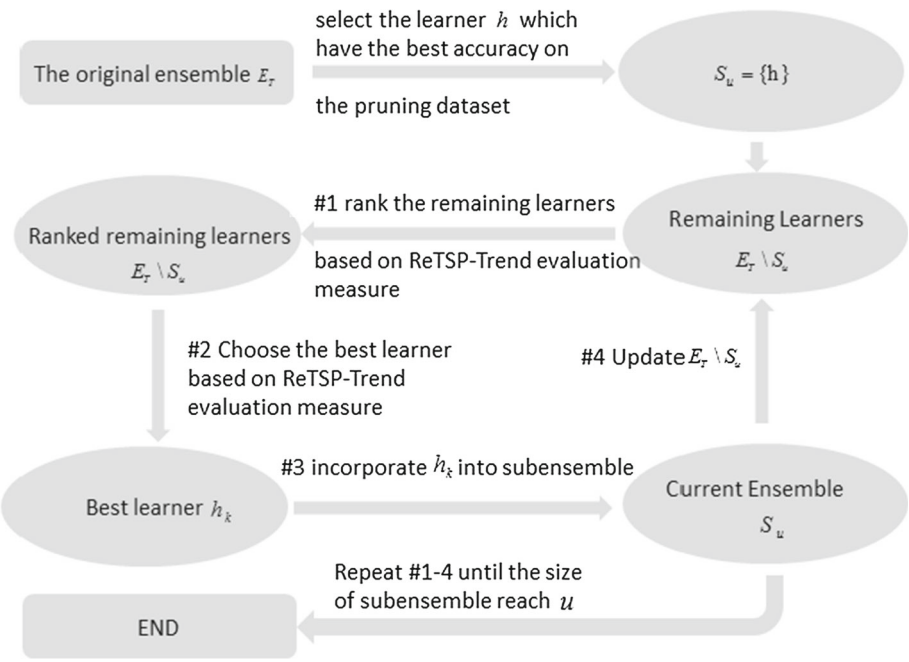


**Fig. 2** The diagram of ReTSP-Trend pruning procedure

### 3.3 The Proposed Pruned Stacking ELMs (PS-ELMs) Algorithm for Time Series Prediction

If we consider an equidistant sampled time series $\{\alpha_i\}_{i=1,\ldots,\mathrm{Q}}$, we can construct a $m$-dimensional state space vector $\mathbf{x}_i$ in the form:

$$\mathbf{x}_i = \left(\alpha_{(i-(m-1))}, \alpha_{(i-(m-2))}, \ldots, \alpha_i\right), \tag{11}$$

$$\hat{\alpha}_{(i+s)} = f\left(\mathbf{x}_i\right), \tag{12}$$

where $s$ is called the horizon of prediction, $m$ denotes the time window size (TWS), and the function $f: R^m \rightarrow R$ is called the fitting function. In this work, we just concentrate on one-step-ahead prediction, i.e., $s = 1$. In this way, we can get a dataset $L = \{(\mathbf{x}_i, \alpha_{(i+1)}), i = m, \ldots, \mathrm{Q}-1\}$, where $\mathbf{x}_i \in R^m$ and $\alpha_{i+1} \in R$ represent input vectors and output variable, respectively. However, to form the input data, we have to determine how large the TWS should be, which is important for forecasting performance. Using a time window of fixed size has proven to be limited in many applications, for the reason that a narrow time window could lead to omission of important information, while a uselessly wide window may cause interfering noise. Ideally, for a given problem, the size of the time window should be adapted to the context. This can be done by using recurrent NNs, which could be learned by a gradient-based learning algorithm, such as BP through time algorithm [50].

In this work, we use a smart way to solve this problem based on ensemble learning paradigm which is proposed in our previous work [34]. First, numbers of models are trained by using training datasets obtained with different TWSs. In other words, we did not use a fixed number of past values to feed into a single model. Instead, we use time windows with different sizes to form different training datasets, based on which numbers of different models could be trained. The detail of the PS-ELMs algorithm is described in the following.

Given a time series $L$, split it into $j$ equal parts $L_1, L_2, \ldots, L_j$ along a chronological order, where $j$ is an integer. Define $L(-j) = L - L_j$ as the level-0 training data, while $L_j$ is kept as the pruning data and level-1 data. For a TWS $\pi \in \{1, 2, \ldots, \mathrm{m}\}$, the dataset $\mathrm{Z}^{(\pi)}$ could be obtained from $L(-j)$ with the TWS $\pi$. Define $K$ members of the committee networks, $M_k$, for $k = 1, \ldots, K$. Separately invoke the $K$ estimators in the committee using $\mathrm{Z}^{(\pi)}$ dataset to produce a set of trained level-0 models, $M_k^{\pi}$, for $\pi = 1, 2, \ldots, m$, $k = 1, \ldots, K$. Obviously, $\Phi$ ($\Phi = K * m$) level-0 estimators could be obtained.

Then, apply our ReTSP-Trend ensemble pruning technique to select a subensemble of level-0 models, $\{M_{\delta_\lambda}, \delta_\lambda \in \{1, \ldots, \Phi\}, \lambda \in \{1, \ldots, P\}\}$, where $P \ll \Phi$ and $P$ denotes the size of final pruned subensemble. Here, dataset $L_j$ is used as the pruning dataset.

After the pruning procedure, apply dataset $L_j$ to each $M_{\delta_\lambda}$ model, $\delta_\lambda \in \{1, \ldots, \Phi\}$, $\lambda \in \{1, \ldots, P\}$, $P \ll \Phi$, and obtain their corresponding outputs. Consequently, we have predictions $\{\mathrm{z}_{i\delta_1}, \mathrm{z}_{i\delta_2}, \ldots, \mathrm{z}_{i\delta_P}\}$ for each input $\mathbf{x}_i$ in $L_j$, where $z_{i\delta_\lambda}$ denotes the $\delta_\lambda$-th model prediction on $\mathbf{x}_i$. Finally, we obtain a new dataset $L_{meta} = \{(\mathrm{z}_{i\delta_1}, \mathrm{z}_{i\delta_2}, \ldots, \mathrm{z}_{i\delta_P}, \alpha_{i+1})\}$, which is the so called level-1 data.

At level-1 learning stage, we derive a level-1 model $\tilde{M}$ from $L_{meta}$. We adopt the ELM algorithm as the level-1 generalizer $\tilde{M}$ to conduct the combination of level-0 estimator predictions in the reason of its generalization ability. The number of hidden neurons and the kernel function of the level-1 ELM generalizer are selected by using the cross-validation method. What's more, we also simply implement averaging the predictions of all level-0 models as the level-1 generalizer.

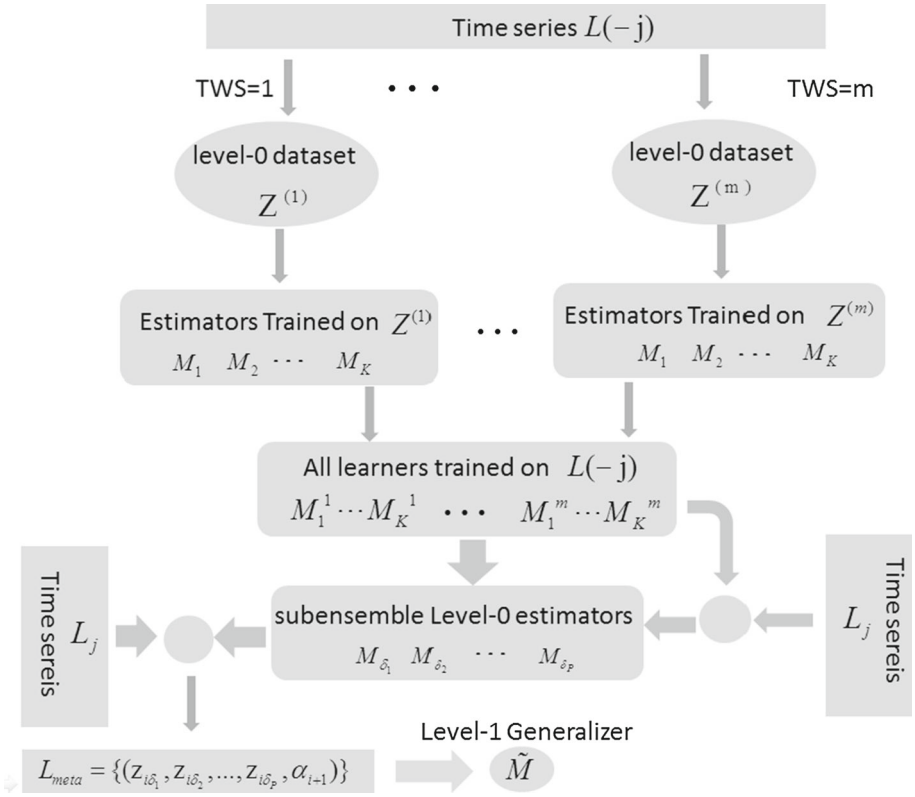


**Fig. 3** The diagram of the pruned Stacking ELMs (PS-ELMs) algorithm

We select suitable models by employing ensemble pruning method, rather than select suitable TWS for a single model in advance. Our method makes the TWS selection procedure no longer necessary. We use Algorithm 2 to show the procedure of training, pruning and Stacking ELM networks. And Fig. 3 is used to illustrate the procedure of our PS-ELMs algorithm.

With regard to the computational complexity of the proposed PS-ELMs algorithm, the analysis to its computational complexity is identical with Algorithm 1, except that the base models in Algorithm 2 have been determined to be ELMs. Therefore, the time complexity of training the original ensemble and then implementing pruning to the ensemble equals to $O(T \cdot t_{trainELM} + u \cdot T)$, where $T$ denotes the size of the original ELMs ensemble, $u$ represents the size of subensemble obtained from pruning and $u \ll T$, and $t_{trainELM}$ depends on the specific ELM training algorithm and the size of training set, while the time complexity of making a prediction for an unknown instance is $O(u \cdot t_{testELM})$, and $t_{testELM}$ denotes the time required for an ELM to make a testing decision.

Algorithm 1 focuses on exhibiting the detailed process of the ReTSP-Trend pruning procedure, while Algorithm 2 shows the process of the proposed PS-ELMs algorithm on the whole. Their computational complexities are consistent with each other. Based on these theoretical analyses, it can be concluded that the prediction complexity is reduced greatly, which is of great concern for the time series forecasting tasks.

**Algorithm 2** Pruned Stacking ELMs algorithm

**Input**: Time series $L = \{\alpha_i\}_{i=1,..,Q}$;

Estimators for training $M_1, M_2, \cdots, M_K$;

$P$ : a constant parameter represent the size of subensemble after pruning procedure;

$\{1,2,..,m\}$ : time window sizes set;

**Output**: Level-0 learners $M_{\delta_1}, M_{\delta_2}, \ldots, M_{\delta_P}$ and level-1 generalizer $\tilde{M}$.

1: Split $L$ into $j$ equal parts $L_1, L_2, ..., L_j$ along a chronological order in order to obtain the level-0 training data $L(-j) = L - L_j$ and pruning data $L_j$.

2: For a time window size $\pi \in \{1,2,..,m\}$, form the datasets $Z^{(\pi)}$, using a time window sizes set $\{1,2,..,m\}$ obtained from $L(-j)$ with the time window size $\pi \in \{1,2,..,m\}$.

3: Separately invoke the $K$ estimators in the committee using $Z^{(\pi)}$ $(\pi \in \{1,2,..,m\})$ dataset. $\Phi$ level-0 models $M_k^{\pi}$, for $\pi = 1,2,...,m$, $k = 1,...,K$ could be produced.

4: Apply the ReTSP-Trend ensemble pruning technique to select a subensemble of level-0 models, $M_{\delta_\lambda}$, where $\delta_\lambda \in \{1,...,\Phi\}, \lambda \in \{1,...,P\}, P \ll \Phi$ and $P$ denotes the size of final pruned subensemble. Here $L_j$ is used as the pruning dataset.

5: Apply the pruning dataset $L_j$ to each $M_{\delta_\lambda}$ model and obtain their corresponding outputs $\{z_{i\delta_1}, z_{i\delta_2}, ..., z_{i\delta_P}\}$ for each input $\mathbf{x}_i$ in $L_j$, where $z_{i\delta_\lambda}$ denotes the $\delta_\lambda$-th model prediction on $\mathbf{x}_i$.

6: Derive a level-1 model $\tilde{M}$ from $L_{meta}$, $L_{meta} = \{(z_{i\delta_1}, z_{i\delta_2}, ..., z_{i\delta_P}, \alpha_{i+1})\}$.

**7: Return**: Level-0 learners $M_{\delta_1}, M_{\delta_2}, \ldots, M_{\delta_P}$ and level-1 generalizer $\tilde{M}$.

## 4 Empirical Analysis and Evaluation

### 4.1 Research Data and Data Pre-processing

The research data in this study consists of four typical stock indices: (1) Dow Jones Industrial Average (^DJI), (2) GlaxoSmithKline plc. (GSK), (3) Hang Seng Index (^HSI), (4) Johnson

Outdoors, Inc. (JOUT). The historical data are collected daily and are obtained from Yahoo Finance [51]. The entire dataset covers the period from 1 January 1996 to 31 December 2012.

We verified the performance of rank-based pruning methods with $k$-fold cross-validation approach by computing mean values. Yet, time series could be problematic for cross-validation. In the forecasting domain, recent patterns should have higher importance when compared with older ones. When the data are not independent, cross-validation becomes more difficult as leaving out an observation does not remove all the associated information due to the correlations with other observations. An approach that is sometimes more principled for time series is forward chaining [52]. Therefore, the similar 5-fold cross-validation approach as adopted in [24] is also used here, where the procedure would be something like this.

For each fold of cross-validation, the entire dataset is divided into three periods. The first period, which is assigned to in-sample estimation, is used for network training, i.e., the training set. The second period is reserved for ensemble pruning, i.e., the selection dataset. The third period, which is assigned to out-of-sample evaluation, is used for testing purpose, i.e., the testing set.

- *fold 1*

The first period is from 1 January 1990 to 31 December 2010, the second period is from 1 January 2011 to 31 December 2011, while the third period is from 1 January 2012 to 31 December 2012.

- *fold 2*

The first period is from 1 January 1990 to 31 December 2009, the second period is from 1 January 2010 to 31 December 2010, while the third period is from 1 January 2011 to 31 December 2011.

- *fold 3*

The first period is from 1 January 1990 to 31 December 2008, the second period is from 1 January 2009 to 31 December 2009, while the third period is from 1 January 2010 to 31 December 2010.

- *fold 4*

The first period is from 1 January 1990 to 31 December 2007, the second period is from 1 January 2008 to 31 December 2008, while the third period is from 1 January 2009 to 31 December 2009.

- *fold 5*

The first period is from 1 January 1990 to 31 December 2006, the second period is from 1 January 2007 to 31 December 2007, while the third period is from 1 January 2008 to 31 December 2008.

There are five attributes in the financial time series:

(1) The highest value that the stock was negotiated in a certain day, denoted by $H_t$;
(2) The lowest value that the stock was negotiated during the same day, denoted by $L_t$;
(3) The value of the first negotiation of the day, i.e., opening price, denoted by $O_t$;
(4) The value of the last negotiation of the day, i.e., closing price, denoted by $C_t$;
(5) The business volume of the stock during the same day.

While there is a lot of missing data in the volume series, we just use the first four attributes of the five attributes. Among the four attributes, the closing price is the attribute that is really important, since most of the professional investors and financial institutions take action based on its value [53]. So our forecasting goal is to forecast the daily close price using its past value and the other three time series attributes. Specifically,

$$\hat{C}_t = f\left(C_{t-1},\ C_{t-2}, \ldots, C_{t-p},\ H_t,\ L_t,\ O_t\right), \tag{13}$$

where $\hat{C}_t$ is the $t$th prediction of closing price by learning algorithm $f$. $C_i$ $(i = t-1,\ t-2, \ldots, t-p)$ is the $i$th observation of the closing price, $H_t$ is the $t$th observation of the highest price, $L_t$ is the $t$th observation of the lowest price, $O_t$ is the $t$th observation of the opening price, and $p$ is the TWS.

Since the attributes of sample sets have different value scales, it is necessary to adjust the scale of each attribute into the range of [0, 1]. It ensures that the larger value input attributes do not overwhelm smaller value inputs, and then helps to reduce prediction errors. We use the following two steps to describe normalization method:

*Step 1* each of the attributes $d_t$ was converted to logarithms returned as:

$$r_t = \log\left(d_t\right). \tag{14}$$

*Step 2* each of the series value $r_t$ was normalized by the linear interpolation equation (15)

$$r'_t = \frac{r_t - MIN}{(MAX - MIN)}, \tag{15}$$

where $r'_t$ is the normalized value, $r_t$ is the value to be normalized, *MIN* is the minimum value of the series to be normalized, *MAX* is the maximum value of the series to be normalized.

### 4.2 Performance Measurement

A performance measurement is necessary to appropriately evaluate the predictive performance of pruned ensemble obtained with different ensemble pruning measures. The performance measurements are defined on the basis of prediction error, which is established as the difference between the real value of the series (target or objective of the prediction) and the predicted value (the output of the ensembles). Therefore, they are presented by the following equation:

$$e_t = \left(target_t - output_t\right), \tag{16}$$

where $target_t$ is the desired output of the prediction model at time $t$, and $output_t$ is the output of the NN model at time $t$. Based on the prediction error, three performance measurements used to evaluate the predictive performance of the pruned ensembles are described below.

#### *4.2.1 RMSE: Root Mean Square Error*

RMSE [54] is the most common metric used to analyze ensemble performance and it is defined by the equation:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(target_t - output_t\right)^2}, \tag{17}$$

where N denotes the number of data values of the testing time series. Obviously, the lower the value of RMSE, the better is the result of the prediction. Even though RMSE is quite common

as a performance measurement, it does not provide complete and convincing evidence about the accuracy of the predictive model. Therefore, other two metrics are also used to evaluate the performance of the proposed models.

#### *4.2.2 MAPE: Mean Absolute Percentage Error*

The measure MAPE [55] describes the errors in percentages which is an advantage in relation to the RMSE measure, since it does not depend on the values or the scale of the time series, which simplifies its usage. MAPE is defined as:

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{target_t - output_t}{output_t}\right|. \tag{18}$$

Clearly, the lower the value of MAPE, the closer is the desired results from the predicted ones.

#### *4.2.3 POCID: Prediction on Change in Direction*

The measurement POCID [53] demonstrates the percentage of the number of correct decisions when predicting whether the value of the time series will increases or decreases in the next time interval. POCID is defined as:

$$\text{POCID} = 100\frac{\sum_{t=1}^{N} D_t}{N} \tag{19}$$

having the value of $D_t$ determined by:

$$D_t = \begin{cases} 1 & if\ (target_t - target_{t-1})(output_t - output_{t-1}) > 0, \\ 0 & otherwise. \end{cases} \tag{20}$$

The values that POCID assumes are in between 0 and 100, so that, the closer the values are to 100, the better is the model prediction. This measurement is important when applied to the stock market, because a correct prediction on the direction of the series of the stock quotation affects directly the financial gains and losses on the investment.

### 4.3 Ensemble Construction and Experimental Results

First, 600 ELM networks were used to assemble the proposed Stacking ELM. The numbers of hidden neurons for each TSP problem are chosen among [1:60]. Therefore, 60 ELMs could be constructed. Meantime, with 10 different TWSs processing time series to form 10 different training dataset for learning algorithm, we can obtain 600 ELM networks. These 600 ELM networks are used as the level-0 learners for Stacking algorithm. We name using ELM model as level-1 generalizer to Stacking ELM networks without (or with) the level-0 learners pruning procedure as ELMSTOFELM-ALL (or ELMSTOFELM-pruned) and using averaging as level-1 generalizer to Stacking ELM networks without (or with) the level-0 learners pruning procedure as AVSTOFELM-ALL (or AVSTOFELM-pruned).

Then, 200 SVR models are constructed for the comparison purpose. The 200 SVR models are constructed based on different internal model parameters of $\varepsilon$-SVR algorithm and different TWSs. Trade-off and kernels are crucial internal parameters for $\varepsilon$-SVR algorithm. Suppose use $k_1$ TWSs, $k_2$ trade-offs, $k_3$ kernels, we can construct $(k_1 * k_2 * k_3)$ SVR models.

**Table 1** The SVRs models with training parameters

| Ensemble | Parameters | | | |
|---|---|---|---|---|
| | Time window size | Trade-off (C) | Polynomial kernels (d) | RBF kernels ($\gamma$) |
| 200 Models (10*2*(2+8)) | [1:10] | [10, 100] | Degree = [2, 3] | [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2] |

**Table 2** RMSE on the test set of Dow Jones Industrial Average (^DJI), GlaxoSmithKline plc. (GSK), Hang Seng Index (^HSI), Johnson Outdoors, Inc. (JOUT) time series

| RMSE | Time series | | | |
|---|---|---|---|---|
| | ^DJI | GSK | ^HSI | JOUT |
| Ensemble | | | | |
| BSM–SVR | 0.0257 | 0.0161 | 0.0202 | 0.0451 |
| BSM–ELM | 0.0011 | **0.0092** | **0.0032** | **0.0120** |
| AVSTOFSVR-ALL | 0.0462 | 0.0190 | 0.0723 | 0.0385 |
| AVSTOFELM-ALL | 0.0023 | 0.0115 | 0.0146 | 0.0157 |
| ELMSTOFSVR-ALL | 0.0390 | 0.2597 | 0.0287 | 0.1341 |
| ELMSTOFELM-ALL | 0.0023 | 0.2083 | 5.7989 | 6899.9512 |
| Method | | | | |
| AVSTOFSVR-pruned | 0.0125 | 0.0115 | 0.0140 | 0.0278 |
| AVSTOFELM-pruned | **0.0010** | **0.0092** | 0.0033 | 0.0121 |
| ELMSTOFSVR-pruned | 0.0013 | 0.0127 | 0.0088 | 0.0854 |
| ELMSTOFELM-pruned | 0.0011 | 0.0133 | 0.0035 | 0.0355 |

The boldface indicates the algorithm which performed best on each time series

In this work, we set $\varepsilon = 0.1$, and two kernels (polynomial kernels and RBF kernels) are used. Table 1 describes the values of parameters that we use to train SVR models.

After the level-0 models have been generated, the ELM and simple averaging is also used as the level-1 generalizer. We use 10-fold cross-validation to determine the optimal number of hidden neurons for ELM from a set [1:30].

Correspondingly, we name using ELM model as level-1 generalizer to Stacking SVR networks without (or with) the level-0 learners pruning procedure as ELMSTOFSVR-ALL (or ELMSTOFSVR-pruned) and using averaging as level-1 generalizer to Stacking ELM networks without (or with) the level-0 learners pruning procedure as AVSTOFSVR-ALL (or AVSTOFSVR-pruned).

Table 2 gives the specifications of RMSE performance on four financial time series. From the results, it shows that BSM–ELM provides a slightly better RMSE performance on average than our proposed Stacking ELM algorithm. A smaller standard error indicates the underlying network has more consistent performance through all the runs. In order to estimate whether there are great difference between the BSM–ELM algorithm and our proposed Stacking ELM algorithm (AVSTOFELM and ELMSTOFELM) on the RMSE performance measurement, we implement the paired t-test on each time series at the 5 % significance level. The results

**Table 3** T-test on RMSE for BSM–ELM and AVSTOFELM algorithms

| Time series | ^DJI | GSK | ^HSI | JOUT |
|---|---|---|---|---|
| T-test (RMSE) | h = 0 | h = 0 | h = 0 | h = 0 |
| | p = 0.7691 | p = 0.9826 | p = 0.9858 | p = 0.9294 |

**Table 4** T-test on RMSE for BSM–ELM and ELMSTOFELM algorithms

| Time series | ^DJI | GSK | ^HSI | JOUT |
|---|---|---|---|---|
| T-test (RMSE) | h = 0 | h = 0 | h = 0 | h = 0 |
| | p = 0.8402 | p = 0.3941 | p = 0.8202 | p = 0.1651 |

**Table 5** The prediction error (MAPE) on the test set of Dow Jones Industrial Average (^DJI), GlaxoSmithK-line plc. (GSK), Hang Seng Index (^HSI), Johnson Outdoors, Inc. (JOUT) time series

| MAPE | Time series | | | |
|---|---|---|---|---|
| | ^DJI | GSK | ^HSI | JOUT |
| Ensemble | | | | |
| BSM–SVR | 0.1812 | 0.0326 | 0.0288 | 0.1073 |
| BSM–ELM | 0.0143 | 0.0221 | 0.0185 | 0.0483 |
| AVSTOFSVR-ALL | 0.2488 | 0.0375 | 0.1072 | 0.0797 |
| AVSTOFELM-ALL | 0.0143 | 0.0221 | 0.0185 | 0.0483 |
| ELMSTOFSVR-ALL | 0.0666 | 0.0574 | 0.1754 | 1.8440 |
| ELMSTOFELM-ALL | 0.0206 | 0.0424 | 0.2472 | 0.1557 |
| Method | | | | |
| AVSTOFSVR-pruned | 0.0854 | 0.0217 | 0.0187 | 0.0668 |
| AVSTOFELM-pruned | **0.0056** | **0.0176** | **0.0037** | **0.0439** |
| ELMSTOFSVR-pruned | 0.0103 | 0.0282 | 0.0117 | 0.1400 |
| ELMSTOFELM-pruned | 0.0066 | 0.0197 | 0.0039 | 0.0644 |

The boldface indicates the algorithm which performed best on each time series

are showed in Tables 3 and 4. And it has clearly shown from Tables 3 and 4 that our proposed PS-ELMs algorithm performs basically the same with BSM–ELM algorithm.

Table 5 gives the specifications of MAPE performance on four financial time series. From the results, it is clear that the AVSTOFELM-pruned algorithm performs the best on these four financial time series. Basically AVSTOFELM obtains a smaller MAPE error. A smaller standard error indicates the underlying network has more consistent performance through all the runs. Additionally, the AVSTOFELM has a slight improvement over ELMSTOFELM network. This implies that constructing PS-ELMs with far cheaper averaging operator other than the ELM is efficient for TSP problem for MAPE performance measure measurement.

Besides the above two evaluation measures, we also test the performance of the PS-ELMs algorithm on POCID evaluation measure which reflects the tendency of time series. Table 4 gives the specifications of POCID performance on these four financial time series. Results showed in Table 6 that PS-ELMs has obvious advantage over the comparable models for

**Table 6** The prediction error (POCID) on the test set of Dow Jones Industrial Average (^DJI), GlaxoSmithKline plc. (GSK), Hang Seng Index (^HSI), Johnson Outdoors, Inc. (JOUT) time series

| POCID | Time series | | | |
|---|---|---|---|---|
| | ^DJI | GSK | ^HSI | JOUT |
| Ensemble | | | | |
| BSM–SVR | 64.3679 | 53.2065 | 65.9985 | 59.0952 |
| BSM–ELM | 78.6567 | 53.3462 | 82.4434 | 66.3310 |
| AVSTOFSVR-ALL | 54.7666 | 54.1302 | 66.3899 | 59.0973 |
| AVSTOFELM-ALL | 78.6567 | 53.3462 | 82.4434 | 66.3310 |
| ELMSTOFSVR-ALL | 78.4214 | 62.1940 | 74.8875 | 56.1811 |
| ELMSTOFELM-ALL | 80.2392 | 59.4070 | 81.0192 | 61.5881 |
| Method | | | | |
| AVSTOFSVR-pruned | 58.7349 | 54.8178 | 69.0207 | 59.3380 |
| AVSTOFELM-pruned | **80.8779** | 60.6056 | 83.0808 | **67.6079** |
| ELMSTOFSVR-pruned | 78.7385 | 54.5364 | 68.3801 | 58.7935 |
| ELMSTOFELM-pruned | 80.3205 | **70.7558** | **83.4754** | 64.4373 |

The boldface indicates the algorithm which performed best on each time series

**Table 7** Pruning rates for AVSTOFELM-pruned algorithm and AVSTOFSVM-pruned (SVM) algorithm

| Algorithms | AVSTOFELM-pruned | | | | AVSTOFSVM-pruned | | | |
|---|---|---|---|---|---|---|---|---|
| | ^DJI | GSK | ^HSI | JOUT | ^DJI | GSK | ^HSI | JOUT |
| Total models | 600 | | | | 200 | | | |
| Ensemble size | 5 | 5 | 5 | 10 | 20 | 30 | 20 | 40 |
| Pruning rates (%) | 99.17 | 99.17 | 99.17 | 98.33 | 90 | 85 | 90 | 80 |

**Table 8** Pruning rates for ELMSTOFELM-pruned algorithm and ELMSTOFSVR-pruned algorithm

| Algorithms | ELMSTOFELM-pruned | | | | ELMSTOFSVR-pruned | | | |
|---|---|---|---|---|---|---|---|---|
| | ^DJI | GSK | ^HSI | JOUT | ^DJI | GSK | ^HSI | JOUT |
| Total models | 600 | | | | 200 | | | |
| Ensemble size | 5 | 10 | 5 | 20 | 30 | 5 | 5 | 10 |
| Pruning rate (%) | 99.17 | 98.33 | 99.17 | 96.67 | 85 | 97.5 | 97.5 | 98.33 |

POCID performance measurement. This implies that constructing PS-ELMs is efficient for financial TSP when we pay much attention on predicting tendency of time series.

Besides the prediction performance, the pruning rates indicate that how much models should be pruned to obtain a good prediction performance for the ensemble. Tables 7 and 8 show the detail of the pruning rate for different prediction algorithms and we can find PS-ELMs is more robust than using the SVM as level-0 learners Stacking algorithms.

In order to estimate whether there are great difference between AVSTOFELM-pruned network and ELMSTOFELM-pruned level-1 generalizer for performance, we implement

**Table 9** T-test on RMSE for averaging and ELM level-1 generalizer

| T-test (RMSE) | Time series | | | |
|---|---|---|---|---|
| | ^DJI | GSK | ^HSI | JOUT |
| AVSTOFELM–ELMSTOFELM-pruned | p = 0.6244<br>h = 0 | p = 0.3972<br>h = 0 | p = 0.9275<br>h = 0 | p = 0.1686<br>h = 0 |

**Table 10** T-test on MAPE for averaging and ELM level-1 generalizer

| T-test (RMSE) | Time series | | | |
|---|---|---|---|---|
| | ^DJI | GSK | ^HSI | JOUT |
| AVSTOFELM–ELMSTOFELM-pruned | p = 0.5955<br>h = 0 | p = 0.7535<br>h = 0 | p = 0.8865<br>h = 0 | p = 0.5901<br>h = 0 |

**Table 11** T-test on POCID for averaging and ELM level-1 generalizer

| T-test (RMSE) | Time series | | | |
|---|---|---|---|---|
| | ^DJI | GSK | ^HSI | JOUT |
| AVSTOFELM–ELMSTOFELM-pruned | p = 0.5936<br>h = 0 | p = 0.1002<br>h = 0 | p = 0.5505<br>h = 0 | p = 0.4614<br>h = 0 |

the paired t-test on the two methods at the 5 % significance level. The results are showed in Tables 7, 8 and 9. From the results, we can find that these two networks are in the same performance level (Tables 10, 11).

Next, we use figures to visualize the predictive performance of the PS-ELMs algorithm. As has been mentioned in Sect. 4.1, 5-fold cross-validation approach has been adopted in our experiments. Therefore, the forecasting results based on the test dataset obtained by the fifth fold cross-validation procedure are reported in detail in the following figures (Figs. 4, 5, 6, 7).

From the above comparisons, we can conclude that PS-ELMs has better generalization performance than its comparison methods on financial TSP problems. In addition, we found that the prediction performance of averaging level-1 generalizer was not remarkable difference from ELM Stacking level-1 generalizer. Therefore, it is not useful to apply ELM algorithm to combine members of level-0 learners in a PS-ELMs for financial time series problems.

## 5 Conclusions

The ELM algorithm needs much less training time compared to the popular BP algorithm and SVM/SVR. What's more, the prediction accuracy of the basic ELM algorithm is usually better than BP and similar to SVM/SVR in many classification and regression applications. However, a single ELM network simply assigns input weights and biases at random, instead of explicit training as in the gradient-decent based network. It also ineluctably brings about

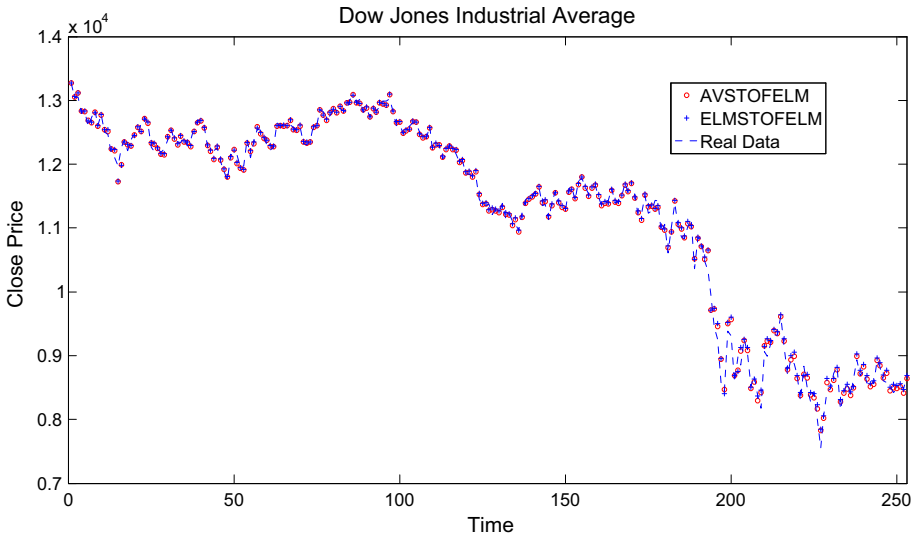


**Fig. 4** Dow Jones Industrial Average (test set)

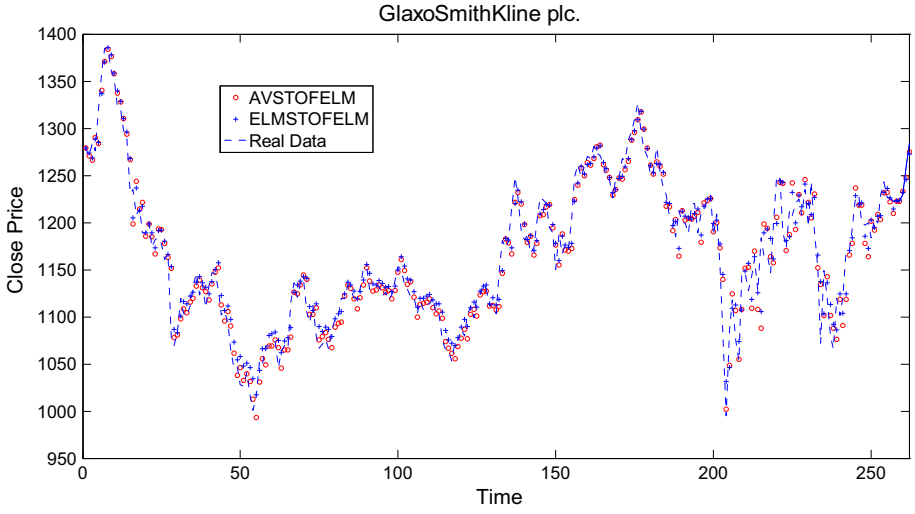


**Fig. 5** GlaxoSmithKline plc. (GSK)

uncertainty and susceptibility to errors in the reason of certain stochastic behaviors of ELM. In order to overcome the above-mentioned shortcomings of ELM, we use an ensemble of ELM models instead of selecting a best single ELM for TSP task in this work, since an ensemble is usually superior to a single predictor given the same amount of training information.

Stacking is a widely used ensemble technique for combining learners and improving prediction accuracy. One major problem of Stacking is how to obtain the right combination of level-0 classifiers and the meta-classifier, specifically in relation to each specific dataset. While in this work, we are aimed at exploring the performance of the ELM algorithm as the meta-model for Stacking due to the many outstanding features that ELM has.

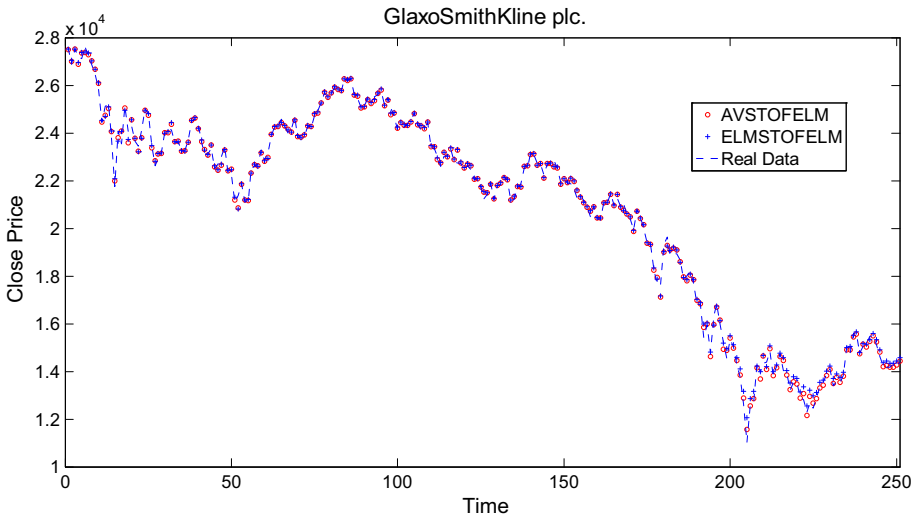


**Fig. 6** Hang Seng Index (^HSI)

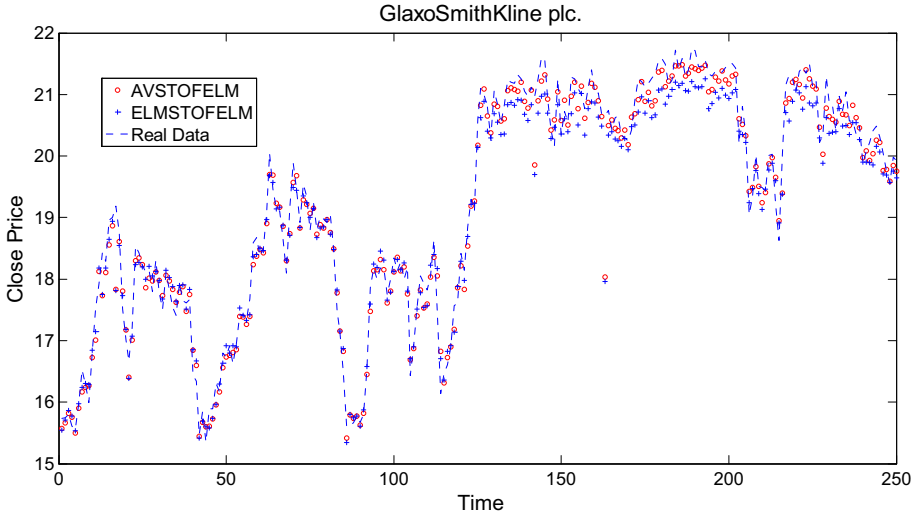


**Fig. 7** Johnson Outdoors, Inc. (JOUT)

What's more, the chosen of suitable level-0 learners is another critical issue of Stacking. The ensembles generated by existing techniques are sometimes unnecessarily large, which can lead to extra memory usage, computational costs, and occasional decreases in effectiveness. And a necessary condition to create a good ensemble of learners is that the error predictions made by level-0 learners are uncorrelated. In this work, we apply ensemble pruning technique to search for a good subset of ensemble members. And a well-understood and high-performance pruning strategy for TSP task, i.e., the ReTSP-Trend pruning method, which has been proposed in our previous work, is used.

To sum up, in this work, a PS-ELMs algorithm is proposed by us for TSP. It uses ELM as the level-0 learning algorithm to train a series of models, uses the ReTSP-Trend pruning method to prune the level-0 ELMs, and applies the Stacking method to combine a pruned committee of ELM networks. Finally, ELM is also used as the meta-learning algorithm for Stacking in comparison with simple averaging as the meta-learning algorithm.

Our motivations behind the development of PS-ELMs, and meanwhile, our contribution of this work are summarized as below.

Firstly, PS-ELMs naturally inherits those salient advantages of ELM, including: extremely fast learning speed, better generalization capability and the avoidance of local minima problem.

Secondly, with PS-ELMs, we hope to ameliorate those specific defects of ELM to some extent, with the help of ensemble pruning paradigm.

Thirdly, we attempt to use the ensemble pruning technique to enhance the robustness and accuracy of the time series forecasting model, while it is found by us that there are very few works studying the application of ensemble pruning technique on time series forecasting. We wish that with the development of PS-ELMs, we could promote the application research of ensemble pruning technique on TSP.

Fourthly, in PS-ELMs, our previously proposed pruning measure ReTSP-Trend is employed, which taking into consideration the time series trend and the forecasting error direction, and could indeed guarantee that the remaining predictor which supplements the subensemble the most will be selected. We hope that, with the help of our proposed ReTSP-Trend pruning method, we can further improve the predictive accuracy achieved by our system on time series forecasting.

Finally, we wish that the development of PS-ELMs will promote our investigation to the popular ensemble technique of Stacked Generalization. We attempt to solve the difficult problems existed in Stacked Generalization with the help of ELM and our proposed ReTSP-Trend pruning method.

The simulation experiments conducted by us show that the proposed PS-ELMs algorithm possesses a high predictive performance on financial TSP. It could not only improve the generalization of the single ELM model but also remove the needless members from the level-0 models. This can be attributed to that the ReTSP-Trend pruning method is effective in pruning the correlated error predictions of level-0 models. Moreover, the prediction time of the ensemble can be reduced sharply in comparison with that of the traditional Stacking without pruning procedure for the level-0 models.

What's more, different choices of level-1 generalizer are implemented and compared. We can find that the averaging and ELM level-1 generalizer are not remarkably different for the prediction performance on the four financial datasets. It indicates that the averaging would be the first choice when using the PS-ELMs algorithm to make predictions on financial time series.

**Acknowledgments** This work is supported by the National Natural Science Foundation of China under the Grant No. 61473150.

## References

1. Neto A, Calvalcanti GD, Ren TI (2009) Financial time series prediction using exogenous series and combined neural networks. In: International joint conference on neural networks, 2009. IJCNN 2009, pp 149–156
2. Abu-Mostafa YS, Atiya AF (1996) Introduction to financial forecasting. Appl Intell 6:205–213

3. Jiang H, He W (2012) Grey relational grade in local support vector regression for financial time series prediction. Expert Syst Appl 39:2256–2262
4. Crone SF, Hibon M, Nikolopoulos K (2011) Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. Int J Forecast 27:635–660
5. White H (1988) Economic prediction using neural networks: the case of IBM daily stock returns. In: IEEE international conference on neural networks, 1988, pp 451–458
6. Zhiqiang G, Huaiqing W, Quan L (2013) Financial time series forecasting using LPP and SVM optimized by PSO. Soft Comput 17:805–818
7. Huang G-B, Zhu Q-Y, Siew C-K (2006) Real-time learning capability of neural networks. IEEE Trans Neural Netw 17:863–878
8. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
9. He Y-L, Geng Z-Q, Xu Y, Zhu Q-X (2015) A robust hybrid model integrating enhanced inputs based extreme learning machine with PLSR (PLSR-EIELM) and its application to intelligent measurement. ISA Trans 58:533–542
10. Peng Y, Wang S, Long X, Lu B-L (2015) Discriminative graph regularized extreme learning machine and its application to face recognition. Neurocomputing 149:340–353
11. Na W, Zhu Q, Su Z, Jiang Q (2015) Research on well production prediction based on improved extreme learning machine. Int J Model Identif Control 23:238–247
12. Wong KI, Vong CM, Wong PK, Luo J (2015) Sparse Bayesian extreme learning machine and its application to biofuel engine performance prediction. Neurocomputing 149:397–404
13. Shamshirband S, Mohammadi K, Tong CW, Petković D, Porcu E, Mostafaeipour A et al (2015) Application of extreme learning machine for estimation of wind speed distribution. Clim Dyn 1–15. doi:10.1007/s00382-015-2682-2
14. Wang W, Yu L, Liu H, Sun F (2015) Extreme learning machine for linear dynamical systems classification: application to human activity recognition. In: Proceedings of ELM-2014, vol 2. Springer, Switzerland, pp 11–20
15. Daliri MR (2012) A hybrid automatic system for the diagnosis of lung cancer based on genetic algorithm and fuzzy extreme learning machines. J Med Syst 36:1001–1005
16. Daliri MR (2015) Combining extreme learning machines using support vector machines for breast tissue classification. Comput Methods Biomech Biomed Eng 18:185–191
17. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) OP-ELM: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21:158–162
18. Gonzalez-Carrasco I, Garcia-Crespo A, Ruiz-Mezcua B, Lopez-Cuadrado JL (2012) An optimization methodology for machine learning strategies and regression problems in ballistic impact scenarios. Appl Intell 36:424–441
19. Gonzalez-Carrasco I, Garcia-Crespo A, Ruiz-Mezcua B, Lopez-Cuadrado JL, Colomo-Palacios R (2014) Towards a framework for multiple artificial neural network topologies validation by means of statistics. Expert Syst 31:20–36
20. Lai KK, Yu L, Wang S, Wei H (2006) A novel nonlinear neural network ensemble model for financial time series forecasting. In: Computational science—ICCS 2006. Springer, Berlin, pp 790–793
21. Kim D, Kim C (1997) Forecasting time series with genetic fuzzy predictor ensemble. IEEE Trans Fuzzy Syst 5:523–535
22. Qian B, Rasheed K (2010) Foreign exchange market prediction with multiple classifiers. J Forecast 29:271–284
23. Khashei M, Bijari M (2012) A new class of hybrid models for time series forecasting. Expert Syst Appl 39:4344–4357
24. Hernández-Lobato D, Martínez-Muñoz G, Suárez A (2011) Empirical analysis and evaluation of approximate techniques for pruning regression bagging ensembles. Neurocomputing 74:2250–2264
25. Margineantu DD, Dietterich TG (1997) Pruning adaptive boosting. In: ICML, 1997, pp 211–218
26. Prodromidis AL, Stolfo SJ (2001) Cost complexity-based pruning of ensemble classifiers. Knowl Inf Syst 3:449–469
27. Martınez-Munoz G, Suárez A (2004) Aggregation ordering in bagging. In: Proceedings of the IASTED international conference on artificial intelligence and applications, 2004, pp 258–263
28. Zhou Z-H, Tang W (2003) Selective ensemble of decision trees. In: Rough sets, fuzzy sets, data mining, and granular computing. Springer, Berlin, pp 476–483
29. Caruana R, Niculescu-Mizil A, Crew G, Ksikes A (2004) Ensemble selection from libraries of models. In: Proceedings of the twenty-first international conference on machine learning, 2004, p 18
30. Banfield RE, Hall LO, Bowyer KW, Kegelmeyer WP (2005) Ensemble diversity measures and their application to thinning. Inf Fusion 6:49–62

31. Martínez-Muñoz G, Suárez A (2006) Pruning in ordered bagging ensembles. In: Proceedings of the 23rd international conference on machine learning, 2006, pp 609–616
32. Martínez-Muñoz G, Suárez A (2007) Using boosting to prune bagging ensembles. Pattern Recognit Lett 28:156–165
33. Zhou Z-H, Wu J, Tang W (2002) Ensembling neural networks: many could be better than all. Artif Intell 137:239–263
34. Ma ZC, Dai Q, Liu NZ (2015) Several novel evaluation measures for rank-based ensemble pruning with applications to time series prediction. Expert Syst Appl 42:280–292
35. Hansen LK, Salamon P (1990) Neural network ensembles. IEEE Trans Pattern Anal Mach Intell 12:993–1001
36. Grigorievskiy A, Miche Y, Ventelä A-M, Séverin E, Lendasse A (2014) Long-term time series prediction using OP-ELM. Neural Netw 51:50–56
37. Zhao G, Shen Z, Miao C, Gay RK (2008) Enhanced extreme learning machine with stacked generalization. In: IJCNN, 2008, pp 1191–1198
38. Huang G-B (2003) Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans Neural Netw 14:274–281
39. Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: 2004 IEEE international joint conference on neural networks, 2004. Proceedings, pp 985–990
40. Ledezma A, Aler R, Sanchis A, Borrajo D (2010) GA-stacking: evolutionary stacked generalization. Intell Data Anal 14:89–119
41. Dietterich TG (2000) Ensemble methods in machine learning. In: Multiple classifier systems. Springer, Heidelberg, pp 1–15
42. Ting KM, Witten IH (1999) Issues in stacked generalization. J Art Intel Res 10:271–289
43. Dzeroski S, Zenko B (2002) Is combining classifiers better than selecting the best one? In: ICML, 2002, pp 123–130
44. Merz CJ (1999) Using correspondence analysis to combine classifiers. Mach Learn 36:33–58
45. Wolpert DH (1992) Stacked generalization. Neural Netw 5:241–259
46. Tamon C, Xiang J (2000) On the boosting pruning problem. In: Machine learning: ECML 2000. Springer, Berlin, pp 404–412
47. Partalas I, Tsoumakas G, Vlahavas I (2009) Pruning an ensemble of classifiers via reinforcement learning. Neurocomputing 72:1900–1909
48. Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach Learn 51:181–207
49. Martinez-Munoz G, Hernández-Lobato D, Suárez A (2009) An analysis of ensemble pruning techniques based on ordered aggregation. IEEE Trans Pattern Anal Mach Intell 31:245–259
50. Assaad M, Boné R, Cardot H (2008) A new boosting algorithm for improved time-series forecasting with recurrent neural networks. Inf Fusion 9:41–55
51. Yahoo Finance. http://finance.yahoo.com/. Accessed 7 July 2015
52. CrossValidated. http://stats.stackexchange.com/questions/14099/using-k-fold-cross-validation-for-time-series-model-selection. Accessed 7 July 2015
53. Neto A, Calvalcanti G, Ren TI (2009) Financial time series prediction using exogenous series and combined neural networks. In: International joint conference on neural networks, 2009. IJCNN 2009, pp 149–156
54. Root-mean-square deviation. http://en.wikipedia.org/wiki/Root-mean-square_deviation. Accessed 7 July 2015
55. Mean absolute percentage error. http://en.wikipedia.org/wiki/Mean_absolute_percentage_error. Accessed 7 July 2015