International Journal of Computational Intelligence Systems, Vol. 6, No. 1 (January, 2013), 96-114

Ensemble ANNs-PSO-GA Approach for Day-ahead Stock E-exchange Prices Forecasting

Yi Xiao[1,3], Jin Xiao[2,3,*], Fengbin Lu[3], Shouyang Wang[3]
[1]*School of Information Management, Central China Normal University, No 152 Luoyu Road*
Wuhan, 430079, China
E-mail: yxiao@mail.ccnu.edu.cn, xybill@amss.ac.cn

[2]*Business School, Sichuan University, No 29 Wangjiang Road*
Chengdu, 610064, China
E-mail: xjxiaojin@126.com

[3]*Academy of Mathematics and Systems Science, Chinese Academy of Sciences, No 80 Zhongguancun East Road*
Beijing, 100190, China
E-mail: fblu@amss.ac.cn, sywang@amss.ac.cn

Received 23 November 2011
Accepted 22 September 2012

Abstract

Stock e-exchange prices forecasting is an important financial problem that is receiving increasing attention. This study proposes a novel three-stage nonlinear ensemble model. In the proposed model, three different types of neural-network based models, i.e. Elman network, generalized regression neural network (GRNN) and wavelet neural network (WNN) are constructed by three non-overlapping training sets and are further optimized by improved particle swarm optimization (IPSO). Finally, a neural-network-based nonlinear meta-model is generated by learning three neural-network based models through support vector machines (SVM) neural network. The superiority of the proposed approach lies in its flexibility to account for potentially complex nonlinear relationships. Three daily stock indices time series are used for validating the forecasting model. Empirical results suggest the ensemble ANNs-PSO-GA approach can significantly improve the prediction performance over other individual models and linear combination models listed in this study.

Keywords: artificial neural networks; ensemble forecasting; particle swarm optimization; genetic operator; stock e-exchange prices

1. Introduction

Stock market is a complex financial market with high volatility, noise non-stationary, unstructured nature, high degree of uncertainty, and hidden relationships. Due to its irregularity, stock e-exchange prices forecasting is regarded as a rather challenging task. The main purpose of forecasting is to reduce the risk in decision making that is important for financial organizations, firm and private investors. The methods of forecasting stock could be classified into two broad classes: fundamental analysis and technical analysis. The fundamental analysis depends upon exact knowledge of the various factors that influence the stock market such as micro-economics, macro-economics, political and even psychological factors. But the knowledge is usually not readily available. The technical analysis attempts to make predictions based on past patterns. However, these patterns are not always evident because of the noise. For traditional statistical methods, it is extremely difficult to capture the irregularity [1]. In these traditional models, we need to

*Corresponding author: Jin Xiao, Business School, Sichuan University, Chengdu, 610064, China. E-mail: xjxiaojin@126.com.

assume a functional relationship between input and output and try to fit the data as per that relationship. This particularly hampers our efforts, since the predictors of stock form multidimensional input space and the relationship between input and output is essentially non-linear [2]. This has encouraged academic researchers and business practitioners to develop more predictable forecasting models [3]. As a result models using artificial intelligence such as artificial neural network (ANN) techniques have been recognized as more useful than conventional statistical forecasting models [4]. The neural networks can simultaneously handle the non-linear data of multidimensional input space. Furthermore, neural networks do not require an explicitly well defined relationship between input and output as they determine their own relationships based on input and output values [5]. With its proven generalization ability, the ANN is able to infer from historical patterns the characteristics of performing stocks. During the last few years, a number of neural network models and hybrid models have been proposed for obtaining accurate prediction results.

Some researches are presented. Brownstone [6] predicts the daily Market close 5 days ahead, and 25 days ahead of the Footsie by neural network. The results indicate that predictions can be produced to a high level of accuracy, in a readily understandable format. Quah [7] and Srinivasan uncover the intricate relationships between the performance of stocks and the related financial and technical variables by neural network. Experimental results obtained this far have been very encouraging. Kuo et al. [8] develop a genetic algorithm based fuzzy neural network (GFNN) to formulate the knowledge base of fuzzy inference rules which can measure the qualitative effect on the stock market. Plikynas et al. [9] use ANN for compound (technical and fundamental) analysis and prognosis' of LNSE, LITIN-A and LITIN-VVP. They employ initial pre-processing (analysis for entropy and correlation) for filtering out model input variables. A wide spectrum of different results has shown a high sensitivity to ANN parameters. Ao [10] designs a simplified automated system to study the correlation between the US market and the Asian markets by employing the evolutionary computation to simulate the markets interactive dynamics. Slim [11] proposes a stochastic neural network (SNN) to the modelling and forecasting the time varying conditional volatility of the TUNINDEX returns. The empirical analysis shows that out-of-simple volatility forecasts of the SNN are superior to forecasts of traditional linear methods (GARCH) and also better than merely assuming a conditional Gaussian distribution. O'Connor and Madden [12] evaluate the effectiveness of using external indicators, such as commodity prices and currency exchange rates. In the experiments presented, basing trading decisions on a neural network trained on a range of external indicators result in a return on investment of 23.5% per annum, during a period when the DJIA index grew by 13.03% per annum. Thomaidis et al. [13] experiment with a "top-down" pruning technique as well as two "bottom-up" strategies that start with simple models and gradually complicate the architecture if data indicate so. Liao et al. [14] propose an improved neural network - the stochastic time effective neural network model and test the forecasting performance of the model by using different volatility parameters. Guresen et al. [15] analyze multi-layer perceptron (MLP), dynamic artificial neural network (DAN2) and the hybrid neural networks which use generalized autoregressive conditional heteroscedasticity (GARCH) to extract new input variables. Jasemi et al. [16] propose the network that is not going to learn the candlestick lines alone or in combination, but is to present a kind of regression model whose independent variables are important clues and factors of the technical analysis patterns; and its dependent variable is the market trend in near future.

The broad spectrum of applications to which neural networks have been applied, however, has revealed some of its drawbacks making researchers in particular suspicious as to their suitability in financial forecasting. As we all know, neural networks are a kind of unstable learning methods. Even for some simple problems, different architectures of neural networks (e.g., different number of hidden layers, different hidden nodes and different initial conditions) result in different patterns of network generalization. Researchers have devoted a great deal of effort during the last decade in order to find the optimal parameters of neural network (e.g., number of units, number of hidden layers, type of neurons, learning rates for supervised and unsupervised training and initial weights, etc.) which can solve an actual problem based on performance evaluation criteria.

Evolutionary computation algorithms which are comprised of four major paradigms, genetic algorithms, evolutionary programming, evolution strategies, and genetic programming, have demonstrated to be suitable for optimization of different ANNs. As a popular evolutionary computation paradigm, namely, the particle swarm optimization (PSO), utilizes a "population" of candidate solutions to evolve toward an optimal or near optimal solution of an actual problem. Because of its simplicity, easy implementation, and quick convergence, PSO has attracted more and more attention and has been applied extensively in various fields. Despite of its success and popularity, Grimaldi et al. [17] have indicated that, although PSO may find solutions of reasonable quality much faster than other evolutionary computation algorithms, it can not improve the quality of the solutions as the number of iterations increases. Hence, a premature phenomenon may occur for the original PSO, especially when optimizing complex multi-objective functions. Therefore, many improved PSO algorithms have been proposed. For example, Alfi et al. [18] have presented a methodology for finding optimal system parameters and optimal control parameters by a novel adaptive particle swarm optimization (APSO) algorithm. Wang et al. [19] have presented a poly-hybrid PSO optimization method with intelligent parameter adjustment.

Unfortunately, more and more researchers have realized that only selecting a single neural-network model with the best performance may lead to loss of potentially valuable information contained by other neural-network models that may have slightly weaker performances. Therefore, some different learning strategies such as combined/ensemble learning and meta-learning have been presented. For example, Abraham and AuYeung [20] present two ensemble approaches: based on a direct error measure and based on an evolutionary algorithm to search the optimal linear combination. Experimental results reveal that the ensemble techniques perform better than the individual methods and the direct ensemble approach seems to work well for the problem considered. Kwon and Moon [21] propose a genetic ensemble of recurrent neural networks for stock prediction model. It shows notable improvement on the average over not only the buy-and-hold strategy but also other traditional ensemble approaches. Chun and Park [22] propose a new learning technique which extracts new case vectors using Dynamic Adaptive Ensemble CBR (DAE CBR). The main idea of DAE CBR originates from finding combinations of parameter and updating and applying an optimal CBR model to application or domain area. Chen et al. [23] propose a flexible neural tree (FNT) ensemble technique. The structure and parameters of FNT are optimized using genetic programming (GP) like tree structure-based evolutionary algorithm and particle swarm optimization (PSO) algorithms, respectively. Experimental results show that the model considered could represent the stock indices behavior very accurately. Aladag et al. [24] propose a new forecast combination approach based on artificial neural networks. The forecasts obtain from different fuzzy time series models are combined by utilizing artificial neural networks. The proposed method is applied to index of Istanbul stock exchange (IMKB) time series. It is seen that the proposed combination approach produces better forecasts than those produced by other combination methods available in the literature. Hung [25] presents the results of using a fuzzy system method to analyze clustering in generalized autoregressive conditional heteroskedasticity (GARCH) models. Although there are many studies on ensemble forecasting, we find that the ensemble model from different base models is often combined in a linear way in the existing studies. However, a linear weighted approach is not necessarily appropriate for the financial time series. Moreover, how to optimize the performance of the base models is a key problem.

To solve the above two main problems, an ensemble model from three different base models, Elman network, generalized regression neural network (GRNN) and wavelet neural network (WNN) combined by support vector machines (SVM) neural network in a nonlinear way is presented. The optimal architectures of the three base models are obtained by the improved swarm particle optimization algorithm (IPSO). The main contributions of this study are summarized as follows:

(a) A three-stage neural-network-based nonlinear weighted ensemble model for forecasting stock indices time series is proposed. In the first stage, a certain data sampling technique is used to generate three different training sets for three base models, a validation set and a testing set. Based on the different training sets, three neural-network base models are optimized by IPSO

algorithm in the second stage. Because the three neural-network base models' training processes do not affect the overall efficiency of time series forecasting system, they can be optimized and formulated in parallel. In the final stage, a neural-network-based nonlinear weighted meta-model is produced by learning the three neural-network base models through SVM neural network.

(b) All parameters in the three base models are adaptively adjusted by the improved particle swarm optimization (IPSO) algorithm. To ameliorate the performance of standard PSO, IPSO employs adaptive nonlinear inertia weight updating with fitness values. At the same time, acceleration parameters are controlled by a declining arccosine function and an increasing arccosine function. Further, the crossover operation and mutation operation are introduced to improve the performance of the candidate particles. We adopt two-point crossover and design a crossover rate only depending on generation and an adaptive mutation rate depending on individual fitness. Finally, the optimal structure and parameters of base models are adjusted by a 2-level algorithm in the training process, i.e., binary particle swarm optimization (BiPSO) and decimal particle swarm optimization (DePSO). With IPSO the deadly drawbacks of the base models, e.g., difficult to select the parameters and frequent confinement to local minima, have been significantly improved.

The rest of this study is organized as follows. Section 2 describes the building process of the proposed model in detail. For further illustration, three stock time series are used for testing in Section 3. Then, Section 4 presents and discusses the results of forecasting and compares the forecasting performance with other methods in terms of all kinds of evaluation criteria. Finally, some concluding remarks are drawn in Section 5.

2. The architecture of the nonlinear ensemble model

The nonlinear ensemble model in this study adopts the concept of metal-earning, which use some individual learning algorithms to extract knowledge from several different data subsets and then to construct a unified body of knowledge, metamodel, that adequately represents the entire dataset.

2.1. *Meta-learning*

As above mentioned, the performance of the ensemble models from several different base models is superior to a single neural-network model with the best performance. The ensemble model is a kind of learning from learned knowledge recently developed as an emerging machine learning technique in order to forecast financial time series using multiple training datasets. Generally, learning involves extracting a pattern $f = f_a$ from a training set, T, while ensemble model does these from several training sets, (T_1, T_2, …, T_n), each of which was used to train an associated base model $f = f_a(i)$, (i = 1, 2, . . ., n). The symbol n represents the number of the base models. The mode of metamodel which learns form learned knowledge by the base models may different from some or all of the base models. In addition, the metamodel will be trained by a new training set, T_M, which is distinct from other training sets, T_n, used by the base models, $f_a(i)$. At present, many different meta-learning approaches have developed such as stacked generalization, boosting, dynamic bias selection, mining meta-knowledge, and inductive transfer, etc.

Metamodel has usually two modeling methods: based on different training sets or different learning algorithms. The former is described as follows:

$$y = f(T_M) \tag{1}$$

$$f = \sigma(f_1(T_1), f_2(T_2), \cdots f_n(T_n)) \tag{2}$$

$$|T_1| = |T_2| = \cdots |T_n| \tag{3}$$

$$f_1 = f_2 = \cdots f_n \tag{4}$$

where y is the output of meta-model, f is the learning algorithm of the meta-model, T_M is the meta-training set, n is the number of base models (i.e. the number of training subset), T_i(i=1,2, …, n) is the ith training subset used by the ith base model, σ is the learning operator of f, $|T_i|$(i=1,2, …, n) is the dimension of the ith training subset and f_i(i=1,2, …, n) is the learning algorithm of the ith base model. Note that the learning algorithm of each base model based on different training sets is identical.

The latter is described as follows:

$$y = f(T_M) \tag{5}$$

$$f = \sigma(f_1(T), f_2(T), \cdots f_n(T)) \tag{6}$$

Note that the learning algorithm of each base model based on identical training sets is different.

This study proposes a novel ensemble model integrating the advantages of two different meta-learning methods. The proposed model uses a hybrid meta-learning strategy in terms of different learning algorithms and different training subsets which include the following three stages:

(a) An original data set D is divided into three parts: training set T, validation set V and testing set S. Then the different training subsets T_1, T_2, …, T_n are sampled from T for the correspond base models.

(b) The optimal architecture of n base models f_i (i=1,2, …, n) are obtained by training subsets T_i (i=1,2, …, n) and validation set V.

(c) Applying the trained m different base models to validation set V, validation results from the m base models can formulate a meta-training set (T_M). Based on the meta-training set, the metamodel is constructed.

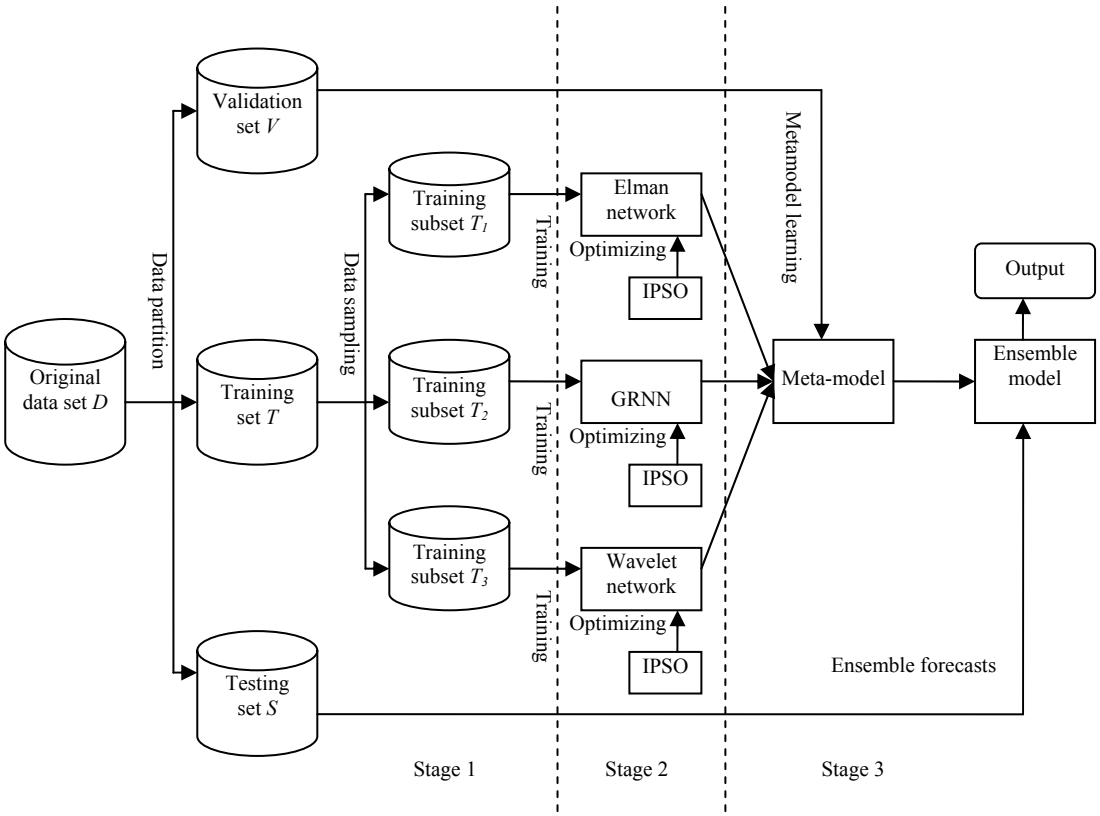

Fig. 1. A flow diagram of the nonlinear ensemble forecast model

In this study, NN is used as both base learner and metalearner. That is, the improved meta-learning process utilizes the nonlinear weighted form to create a novel metamodel different from the existing linear weighted metamodeling. The improved three-stage nonlinear meta-learning process is illustrated in Fig. 1. There are three main problems to be solved: (a) create n different training subsets from the original training set T for n base models, (b) determine the optimal architecture of each neural-network base model, and (c) create a metamodel with different metadata produced by the NN base models.

2.2. *Data partition and sampling*

In forecasting financial time series by neural network, data partitions can have a significant impact on the final results. Generally, the initial data set is only split into training set and testing set. The former set is used for

model construction and the latter one is used for model testing. However, a third set from the original data set, validation set, can effectively improve the robustness of neural networks. Therefore, this study divides the original data set into three different parts non-overlapping: training set, validation set and testing set. The partition ratios usually are no consensus and selected subjectively. However, the appropriate ratios will generate better forecasting accuracy such as 7:2:1 [26]. The partition ratios are not absolute according to different problems.

How to create n training subsets from the original training set T is a problem after data partition? There are three common sampling techniques to create training subsets: direct replication, bagging and noise injection. In direct replication method, training subsets is a sample duplicate of original training set T.

Because of the feature of its random sampling with replacement, Bagging [27] has become a widely used data sampling method in machine learning. But there are maybe many duplicates in some training subsets. Noise injection can increase the independence between different training subsets and further effectively reduce variance between models by inserting noise to the training dataset [28]. Unfortunately, the injected noise may distort the characteristic of original data. To overcome above three sampling techniques, an interval sampling method is proposed as

$$T_i(j) = T(i + n(j-1)), (i = 1,2,\cdots,n), \\ (j = 1,2,\cdots, i + n(j-1) \le N) \quad (7)$$

where $T_I(j)$ is jth element of ith training subset, T the original training set, n the number of training subsets, and N is the size of T.

2.3. *Neural-network base models*

The performance of base model is the basis to construct an effective ensemble model. The diversity of the base models is crucial for improving the performance of the ensemble model. The neural-network base models usually can be formulated parallel by different ways such as different initial weights, different architecture (e.g., different numbers of layers or numbers of nodes in each layer), different training algorithms (e.g., the gradient descent or Levenberg–Marquardt algorithm, etc). The neural-network base models utilize usually identical technique with different parameters, however it is proven that neural-network base models based on different technique are much helpful to improve the generalization ability and adaptability to actual problems of an ensemble model. Therefore, three different neural networks, Elman network, generalized regression neural network (GRNN) and wavelet neural network (WNN) are adopted because of their respective advantages in financial time series forecasting in this study.

2.3.1 *Neural network*

Neural networks (NNs), first introduced in 1943 [29], are a set of systems derived through neuropsychology models. The basic idea of NN is to emulate the biological system of the human brain to learn and identify patterns. The NN is widely used for time series forecasting because its flexible nonlinear modeling capability can capture the nonlinear characteristics of time series well. When applying NN to time series forecasting, the final output of the ANN-based forecasting model can be represented as

$$y_{t+1} = \varphi(y_t, y_{t-1}, y_{t-2}, \cdots, y_{t-p+1}, v) \quad (8)$$

where v is a vector of all parameters, p nonlinear time dependency of size (lag), and φ is a function determined by the network structure and connection weights. Thus, in some senses, the NN model is equivalent to a nonlinear autoregressive model.

2.3.2 *Elman network*

Elman networks belong to the class of recurrent neural networks (RNN) architecture. Elman network is a three layers feedforward neural network with the addition of a recurrent connection from the output of the hidden layer to its input. The network is augmented at the input level by additional units, called context units. The number of context units is equal to the number of hidden units. The augmented input units, including both the input units and the context units activate the hidden units [30]. The inputs to the context units are the outputs of the hidden neurons forming the second or hidden layer. The outputs of the context units and the external input neurons are fed to the hidden neurons. The context unit values at time step t+1 are exactly the same as the hidden unit values at time step t. There current units which transfer the previous state of the hidden units to the input layer are recognized as a one-step time delay.

Context units are also known as memory units as they store the previous output of the hidden neurons.

During Elman networks operation, the activation values of the input units are set to a desired input pattern. The activation value of every hidden unit is computed by multiplying each input and context activation value by the value of the weight from the unit to the hidden unit. These values are then summed, the bias of the hidden unit is added, and the sum is passed through a squashing function *f*. The resulting value then constitutes the output value of the hidden unit. In the Elman network, the squashing function used is the logistic *f*. Then, the activations of output units are calculated based on the hidden units in an analogous manner. This represents one time step. Next, the activation of each hidden unit is copied into a corresponding context unit on a one-for-one basis with fixed weights of 1, and then the next time step is performed. This is equivalent to a recurrent connection from every hidden unit to itself and is more restrictive than the arbitrary recurrent connections allowed by Minsky's claim [31].

Suppose that *n*, *l*, *m* are number of the input units, hidden units and output units. The primary input is $xi(i=1,2\ldots,n)$, and the network output is y_k(k=1,2,…,m). $w_{ji}(i=1,2,\ldots,n;\ j=1,2,\ldots l)$, $w_{jr}(r=1,2,\ldots,l;\ j=1,2,\ldots l)$, $w_{kj}(j=1,2,\ldots,l;\ k=1,2,\ldots m)$ are the weights of the connections between the input and hidden units, between the recurrent and the hidden units, and between the hidden and the output units, respectively. b_j and b_k are biases of hidden units and output units, and $f(\cdot)$ and $g(\cdot)$ are hidden and output functions, respectively. The architecture of Elman network can be written mathematically by

The output of the hidden unit:

$$y_j^{'}(t)=f\left(\sum_{i=1}^{n}w_{ji}x_i+\sum_{r=1}^{l}w_{jr}y_r^{'}(t-1)+b_j\right) \quad (9)$$

The output of the output unit:

$$y_k(t)=g\left(\sum_{j=1}^{l}w_{kj}y_j^{'}(t)+b_k\right) \quad (10)$$

2.3.3 *Generalized regression neural network*

The generalized regression neural network (GRNN) is memory based feedforward network which was introduced [32] as a generalization of both the radial basis function network (RBFN) and probabilistic neural network (PNN). The GRNN is a powerful tool for linear or nonlinear regression based on kernel estimation theory, which builds the sought function surface in a nonparametric fashion through the available data set. The GRNN can exhibit high accuracy and robustness to sparse and noisy data and its estimate converges to the conditional mean surface while introducing more data samples. The flexible structure of GRNN makes it amenable to adaptation for different environments.

At its standard form, it allows for a simple implementation and a very fast training procedure due to the single window bandwidth parameter σ (sigma) that regulates the smoothness of the regression surface. Moreover, unlike other networks, such as MLFNN or Elman, it does not require an exact topology definition.

The GRNN architecture includes four layers, namely, the input, hidden, summation and output layers. Unlike the most popular backpropagation (BP) algorithm that trains multilayer feedforward networks iteratively, the GRNN training is a single pass procedure. In addition, the GRNN formulation comprises only one free parameter that can be optimized fast. Consequently, the GRNN trains itself in a significantly shorter time, as compared with the BP algorithm.

The GRNN utilizes the Parzen Probability Density Estimator [33] between the independent vector random variable *X* with dimension *m*, and dependent scalar random variable *Y*. Assume that *x* and *y* are the measured values for *X* and *Y* variables, respectively. The clustering version of the GRNN with multiple hyper-spherical kernels is proposed as shown below:

$$\hat{y}(x)=\frac{\sum_{j=1}^{n}y_i\exp\left[-\sum_{j=1}^{m}(x_j-x_{ij})^2/\sigma_{ij}^2\right]}{\sum_{j=1}^{n}\exp\left[-\sum_{j=1}^{m}(x_j-x_{ij})^2/\sigma_{ij}^2\right]} \quad (11)$$

where x_i and y_i are the *i*th training set data, x_i the vector form of variable *x*, $y(x)$ the predicted output, m the dimension of input domain, and n is the number of kernels. x_{ij} and σ_{ij} denote the center and sigma of *j*th variable for the *i*th pattern node, respectively.

2.3.4 *Wavelet neural network*

Wavelet is a type of transformation that retains both time and frequency information of the signal [34]. The transformation process from time domain to time scale domain is a WT, technically known as signal decomposition because a given signal is decomposed into several other signals with different levels of resolution. From these decomposed signals, it is possible to recover the original time domain signal without losing any information. This reverse process is called the inverse WT or signal reconstruction. In Fourier transform, only the sine and cosine functions can be chosen as the basis functions. However, wavelet transformation (WT) has versatile basis functions to be selected based on the type of the signal analyzed. Wavelet transforms can be divided in two categories: continuous wavelet transform (CWT) and discrete wavelet transform (DWT).

The continuous wavelet transform $f_x(a,b)$ of function $x(t)$ with respect to a mother wavelet is shown by

$$f_x(a,b) = \int_0^{+\infty} x(t)\psi_{a,b}^*(t)dt \quad (12)$$

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), a,b \in R, a > 0 \quad (13)$$

where the mother wavelet, $\psi(x)$, is a single fixed function such as Morlet function from which all basis functions $\psi_{a,b}(x)$ can be derived through Eq.13. The dilation (scale) parameter a ($a \in R$ and $a>0$) controls the spread of the wavelet and translation (time-shift) parameter b ($b \in R$) determines its central position and the superscript * represents the complex conjugate (R denotes real number).

The WT transforms the function from original time domain in to wavelet (a, b) domain. A function $x(t)$ having both smooth global variations and sharp local variations can be effectively represented in wavelet domain by corresponding wavelet function $f_x(a,b)$.

The discrete wavelet transform is defined by

$$f_x(a,b) = \frac{1}{2^{a/2}}\int_{-\infty}^{+\infty} x(t)\psi_{a,b}^*\left(\frac{t-b2^a}{2^a}\right)dt \quad (14)$$

where the asterisk denotes the complex conjugate, a and b are scale and time-shift parameters, respectively, and $\psi(x)$ is a selected basis function (mother wavelet).

Both continuous and discrete wavelet transforms have been used to implement wavelet neural networks. Normally, the former can provide much excellent performance in nonstationary signal analysis and nonlinear function modeling.

The neural networks provide flexible mapping between inputs and outputs. Hornik et al. [35] have theoretically proved that a three-layer feedforward neural network (MLFN) can approximate any continuous function arbitrarily well given a sufficient number of middle-layer nodes. However, MLFN with using sigmoid function has some limitations such as settle in local minima of the error surface and convergence too slowly. Except the radial basis function neural networks (RBF), WNNs are also an improvement approach to MLFN. Wavelet function is the same as radial basis function is a local function and influence the networks output only in some local range. Although RBF is also local function, but it does not have the spatial-spectral zooming property of the wavelet function, and therefore it cannot represent the local spatial spectral characteristics of the function. WNN shows surprising effectiveness in solving the conventional problems of poor convergence or even divergence encountered in other kinds of neural networks [36].

In this study, a Morlet mother function (shown Eq.15.) is used as node activation function for the hidden layer of a three-layer MLFN. The dilation and translation parameters, a_t and b_t, of the Morlet function for each node in the hidden layer are different and they need to be optimized. In the WNN, the gradient descend algorithm is employed and the error is minimized by adjusting weight vector of the connections between input units and hidden units and between hidden units and output units, and a_t and b_t.

$$y = \cos(1.75x)e^{-x^2/2} \quad (15)$$

2.4. *Optimizing base models*

2.4.1 *Particle swarm optimization algorithm*

Particle swarm optimization is a population-based stochastic optimization algorithm which has been proposed by Eberhart and Kennedy in 1995. The concept is mainly from the natural flocking and swarming behavior of birds and insects. It is considered

to be able to optimize the performance of the ANN by improve its disadvantages such as difficult to select the parameters and easy to get stuck in a local minimum, etc., because it does not require gradient and differentiable information.

Suppose that the search space is h dimensional, the particles of the swarm can be represented by an n dimensional vector $X_i = (x_{i1}, x_{i2}, ..., x_{ih})^T$. The fitness of each particle can be evaluated according to the objective function of the actual optimization problem. The velocity of each particle can be represented by n dimensional vector $V_i = (v_{i1}, v_{i2}, ..., v_{ih})^T$. Let $P_b = (p_{b1}, p_{b2}, ..., p_{bh})^T$ be the last best position of the i-th particle, which is noted as its individual best position. Further, $G_b = (g_{b1}, g_{b2}, ..., g_{bh})^T$ is the global best position. The new velocity of particle will be assigned according to the following equations:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1[p_{bj} - x_{ij}(t)] + c_2 r_2[g_{bj} - x_{ij}(t)] \quad (16)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), j = 1,2,\cdots,h \quad (17)$$

$$v_{ij} \in [v_{\min}, v_{\max}], x_{ij} \in [x_{\min}, x_{\max}] \quad (18)$$

where c_1 and c_2 represent the acceleration parameters, w represents the inertia weight, and r_1 and r_2 are random numbers ranging from 0 to 1. The velocities of the particles on each dimension are clamped to a maximum velocity: v_{max}. The new position of each particle is calculated by Eq. (16).

2.4.2 *The improved particle swarm optimization algorithm*

Although the traditional PSO can usually find good solutions rapidly, it may be trapped in local minimum and fail to converge to the best position. In order to reduce the opportunity of trapping in a local optimum, expand the search scope of the algorithm and enhance the algorithm's climbing ability, it is certainly critical to always maintain the diversity of particles. The existing algorithms such as chaos mechanism optimization [37], hybrid simplex search PSO [38], comprehensive learning PSO, dynamic random search technique [39] are difficult to solve the two problems (global optimization and premature convergence) simultaneously. Therefore, we design an improved particle swarm optimization (IPSO) with adaptive nonlinear inertia weight and dynamic arccosine function acceleration parameters. At the same time, the crossover operation and mutation operation of GA are introduced in IPSO in order to improve the performance of the candidate particles.

2.4.2.1 Improved acceleration coefficients

In the particle swarm optimization algorithm, acceleration coefficients c_1 and c_2 control the "cognitive" part and the "social" part of the particle velocity, respectively. In general, a large "cognitive" c_1 and a small "social" c_2 in the initial stages and a small "cognitive" c_1 and a large "social" c_2 in the last stages can balance the performance in the entire optimization process [40]. Based on this idea, many methods have been proposed such as linear adjustment strategy, fuzzy control strategy, and random change strategy. However, these methods are unstable. Therefore, we propose a dynamic acceleration parameters adjustment strategy based on arccosine function. c_1 and c_2 are controlled by a declining arccosine function and an increasing arccosine function. This strategy attempts to promote particles to be placed in an unexplored area so that they can contribute to the process of finding better solutions in the early stages of optimization. The method is more conducive to getting rid of the interference of local minimum, obtaining the global optimal solution to avoid premature convergence, and improving the convergence speed and accuracy in the latter stages of optimization. The strategy can be represented as

$$c_1 = c_{1_{start}} + (c_{1_{end}} - c_{1_{start}}) \times [1 - \arccos(\frac{-2Iter}{Iter_{\max}} + 1)/\pi] \quad (19)$$

$$c_2 = c_{1_{start}} - c_1 \quad (20)$$

where c_{start} represents the iteration initial value of acceleration parameters, c_{end} represents the iteration final value of acceleration parameters, $Iter$ is the current iteration number, $Iter_{max}$ is the maximum iteration number.

2.4.2.2 Improved inertia weight

The inertia weight w represents the contribution of past velocity values to the current velocity of the particle. A large inertia weight biases the search towards global exploration, while a smaller inertia weight directs towards fine-tuning the current solutions. Suitable

selection of the inertia weight and acceleration coefficients can provide a balance between the global and the local search [41]. Based on this idea, many methods have been proposed such as linear decreasing inertia weight strategy, random inertia weight strategy [42], inertia weight strategy based on concave function and convex function [43], and fuzzy control strategy. However, these methods are not adaptive. In this study, we employ an adaptive nonlinear adjustment inertia weight strategy depending on particle's fitness value, which will help balance the exploring and exploiting capabilities at different stages during its search process. The strategy can be represented as

$$w = \begin{cases} w_{\min} + \dfrac{(w_{\max} - w_{\min})(fitness - fitness_{\min})}{fitness_{avg} - fitness_{\min}}, \\ (fitness \le fitness_{avg}) \\ w_{\max}, (fitness > fitness_{avg}) \end{cases} \tag{21}$$

where w_{min}~w_{max} represent the range of inertia weight, *fitness* represents the current fitness value of some particle, $fitness_{min}$ and $fitness_{avg}$ represent the minimum fitness value and the average fitness value of all particles respectively. It can be seen from Eq. (21) that, the inertia weight will increase when the fitness values of particles are consistent (become local optimum) and will decrease when the fitness values of all particles are scattered. Therefore, the inertia weights of the superior particles whose fitness values are larger than the average fitness value are smaller to protect their properties. In contrast, the inertia weights of the poor particles whose fitness values are smaller than the average fitness value are larger so that they can search better space.

2.4.2.3 Adaptive genetic operators

In PSO algorithm, when the individual optimum solution P_{best} has not been updated for a long time in the latter part of the training, the particles will be close to the global optimum solution G_{best}. At this point the particle update velocity mainly depends on wv_{ij} of the first part of Eq. (16) because the inertia weight w <1, the particle velocity will become increasingly smaller. The particle swarm will "fly" toward a direction, which will lead to its falling into local minimum position. In this study, the adaptive genetic operators (crossover operation and mutation operation) are introduced in order to improve the performance of the candidate particles. The particles can execute crossover operation and mutation operation according to a certain probability.

Crossover is the main search operator in GAs, creating offsprings by randomly mixing sections of the parental genome. In this study, we use two-point crossover and design a crossover rate only depending on iteration number and not associating with the individual fitness. The crossover rate can be represented as

$$p_{ct} = p_{c,\max} \times q^{(-t/T)}, q \in (2,10) \tag{22}$$

$$p_c(t) = \begin{cases} p_{ct}, p_{ct} > p_{c,\min} \\ p_{c,\min}, p_{ct} \le p_{c,\min} \end{cases} \tag{23}$$

$$[p_{c,\min} < p_{c,\max}] \in (0,1) \tag{24}$$

where p_{ct} is calculated variable, T is maximum iteration number, t is current iteration number, q is the decreasing coefficient of crossover probability, $p_{c,\min}$ is minimum crossover probability, $p_{c,\max}$ is maximum crossover probability, $p_c(t)$ is crossover probability of t-th iteration.

A small fraction of the offsprings is randomly selected to undergo genetic mutation. The mutation operator randomly picks a location from a bit-string and flips its contents. To avoid premature convergence, we design an adaptive mutation rate depending on individual fitness as follows,

$$p_m = \begin{cases} p_{m,\min} + \dfrac{(p_{m,\max} - p_{m,\min})(f - f_{\min})}{f_{avg} - f_{\min}}, \\ f \le f_{avg} \\ p_{m,\max}, f > f_{avg} \end{cases} \tag{25}$$

$$[p_{m,\min} < p_{m,\max}] \in (0,1) \tag{26}$$

where $f_{\max}$ is maximum fitness value of current population, f_{avg} is average fitness value of each iteration population, f is fitness value of current mutation individual, $p_{m,\min}$ is minimum mutation probability, and $p_{m,\max}$ is maximum mutation probability.

In order to improve search efficiency, take advantages of PSO's training speed and GA's global

search, the genetic operator control function in this study is defined as

$$GP_k = 1 - \frac{1}{1+\ln k}, k = 1,2,\cdots, \quad (27)$$

$$rand(0,1) < GP_k \quad (28)$$

where k represents current iteration number. In the process of each iteration, a random number will be created ranging from 0 to 1. If the number is less than GP_k, the current particle will execute genetic operator. As can be seen from Eq. (27), in the early iterations $GP_k << 1$, genetic operator will be executed at a small probability, in the later iterations GP_k, it will be close to be 1, so the particle will execute genetic operator at greater probability. Genetic operator expands population search space shrinking in the process of iteration, so that particles can escape from the optimal value searched previously to a larger search space. The particles maintain the diversity of the population, thus it increases the possibility of finding better solutions.

2.4.3 *NN optimized by the improved PSO*

The deadly drawbacks of the NN (frequent confinement to local minima and parameters selection) are expected to be improved with IPSO. The basic idea is to optimize weights and bias of NN by decimal particle swarm optimization (DePSO), and optimize the NN structure by binary particle swarm optimization (BiPSO). A particle in DePSO real-coded represents a set of NN weight vector and bias weight vector.

Let the number of input layer nodes be R, the number of hidden layer nodes Q, the output layer nodes S, the weight and bias vector of NN can be represented as

$$\begin{aligned} X = [&w_{11}^1,\cdots w_{1Q}^1, w_{21}^1,\cdots w_{2Q}^1,\cdots w_{R1}^1,\cdots w_{RQ}^1,\\ &b_1^1,\cdots b_Q^1,\\ &w_{11}^2,\cdots w_{1S}^2, w_{21}^2,\cdots w_{2S}^2,\cdots w_{Q1}^2,\cdots w_{QS}^2,\\ &b_1^2,\cdots b_S^2] \end{aligned} \quad (29)$$

$$h = R(Q+1) + Q(S+1) \quad (30)$$

where $w_{ij}^1 (i = 1,\cdots R; j = 1,\cdots Q)$ represents the weight vector from the input layer to the hidden layer, $w_{ij}^2 (i = 1,\cdots Q; j = 1,\cdots S)$ represents the weight vector from the hidden layer to the output layer, $b_i^1 (i = 1,\cdots Q)$ represents the hidden layer bias vector, $b_i^2 (i = 1,\cdots S)$ represents the output layer bias vector, and h is the dimension of vector X.

A particle binary-coded in BiPSO represents the corresponding hidden layer node, that is, 1 represents the corresponding hidden layer node existence and 0 represents inexistence. The particle velocity is updated according to the Eq. (16). The particle position is updated by the state transition probability depending on the particle velocity. When the particle velocity is greater than a certain value, the particle will be 1 at a larger probability.

The number of hidden nodes is not generally less than the number of input layer nodes and more than the twice of the sum of input layer nodes and output layer nodes. The BiPSO can be represented in the binary form as

$$x_{ij}(t+1) = \begin{cases} 0, \rho_{ij}(t+1) > sig(v_{ij}(t+1)) \\ 1, \rho_{ij}(t+1) \le sig(v_{ij}(t+1)) \end{cases}, \quad (31)$$

$$j = 1,2,\cdots,h$$

$$h = 2 \times (R+Q) \quad (32)$$

where $\rho_{ij}(t+1) \in [0,1], j = 1,2,\cdots,h$ is a random number ranging from 0 to 1, $sig(\cdot)$ is a sigmoid function and $sig(x) = 1/(1+\exp(-x))$, and h is the dimension of vector. If $x = 1$, the corresponding hidden layer node exists, and the weight and bias vector of that node in DePSO is valid. Otherwise, if $x = 0$, the corresponding hidden layer node does not exist, and the weight and bias vector of that node in DePSO is invalid.

2.5. *Non-linear neural-network metamodel*

After the three neural-network base models are trained, next work is how to integrate or combine them by the metamodel. The mode of integration or combination can be defined as

$$y = \sum_{i=1}^{n} w_i f_i(T_i) \quad (33)$$

where y is aggregate output combined the outputs of these base models, n the number of base models (i.e., the number of training subset), w_i the assigned weight of f_i, f_i(i=1,2, …, n) the learning algorithm of ith base model, and T_i(i=1,2, …, n) ith training subset used by ith base model.

There are four common strategies to determine the weights of base models w_i: simple averaging, simple, mean squares error (MSE), stacked regression, and error-variance-based weighting [44]. Besides the individual feature of different strategies, the existing integrated technique is built on linear assumption. However, linear strategies are not necessarily sufficient for financial time series. A nonlinear integrated strategy is proposed to construct a metamodel by using SVM neural network, which is different from the base neural networks in this study. In this nonlinear metamodeling approach, outputs of base neural-network models construct a meta-training set (T_M). This metamodel, SVM neural network, in the final stage can be trained by T_M and assessed by testing set S.

The nonlinear ensemble forecasting model can be defined as

$$y = f(y_1, y_2, \cdots y_n) \tag{34}$$

where $f(\cdot)$ is a nonlinear ensemble function realized by SVM neural network and y_i is the forecast of ith base model.

To validate the effect the proposed the nonlinear ensemble model, an individual Elman network, an individual GRNN, an individual WNN, the linear combination models, and the proposed model to predict stock indices so as to compare forecasting performance.

3. Empirical analysis

3.1. *Data preparation*

The stocks data used in this paper are daily observations obtained from Wind database (http://www.wind.com.cn). They consist of the Shanghai composite index, Shenzhen component index and Shanghai-Shenzhen 300 index studied in this paper. The entire data set of each stock index covers the period of five years. Each time series is split into three sets: training set, validation set and testing set. The first set is used to determine the specifications of the model and parameters of the forecasting technique, the second set is used to not only evaluate the good or bad performance of the predictions of the base models based on evaluation measurements but also construct a meta-training set with outputs of base models, and the third set is used for out-of-sample evaluation of the forecasting model. In addition, the training data set is divided into three training subsets by the interval sampling algorithm. Table 1 shows the information about the time series and size of subsets used.

Table 1. Time series: training set, validation set and testing set.

Stock indices	Training Set			validation set			Testing set		
	Start	End	Obser-vations	Start	End	Obser-vations	Start	End	Obser-vations
Shanghai composite index	2004.7.1	2008.6.30	972	2008.7.1	2009.6.30	244	2009.7.1	2010.6.30	244
Shenzhen component index	2005.1.1	2008.12.31	971	2009.1.1	2009.12.31	244	2010.1.1	2010.12.31	242
Shanghai-Shenzhen 300 index	2005.7.1	2009.6.30	973	2009.7.1	2010.6.30	244	2010.7.1	2011.6.30	243

3.2. *Data Preprocessing*

The data must be normalized before training, which can be described as the following formula:

$$x_n(i,j) = \frac{x(i,j) - \min(X)}{\max(X) - \min(X)} \tag{35}$$

where $x_n(i,j) \in [0,1]$ is the normalized data, $x(i,j)$ is the original data, $\min(X)$ and $\max(X)$ represent the maximum and minimum of the original data.

Some common features from past stocks time series of Shanghai composite index, Shenzhen component index and Shanghai-Shenzhen 300 index are extracted for training and testing purposes. Each set of data are normalized by dividing each value by the maximum value of each set such that each normalized value is less than or equal to unity. Normalization of input data is necessary for obtaining correct trigonometric expansion.

3.3. *Performance measures*

To assess the ensemble prediction model, the forecasts are compared with the true realizations. Following performance measures are used.

Mean absolute error, MAE

$$MAE = \frac{1}{N}\sum_{i=1}^{N} | T_i - T_i^{'} | \tag{36}$$

Mean Absolute Percentage Error, MAPE

$$MAPE = \frac{1}{N}\sum_{i=1}^{N} \left| \frac{T_i - T_i^{'}}{T_i} \right| \tag{37}$$

Root Mean Squared Error, RMSE

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (T_i - T_i^{'})^2} \tag{38}$$

The symbol N is the total number of data patterns. T_i and $T_i^{'}$ represent the actual value and prediction at time i. MAE, MAPE and RMSE are the metrics used to estimate the error of prediction. MAE, MAPE and RMSE are widely used statistical metrics that estimate the error of prediction by measuring the deviation between actual and forecasted value returned in this case. Smaller values of these metrics indicate higher accuracy in forecasting.

Of course accuracy is one of the most important indicators for forecasting models—the others being the cost savings and profit earnings generated from better decisions. From the business, the latter is usually more important because for the business practitioners, the aim of forecasting is to support or improve decisions so as to make more profit. Thus profits or returns are more important than conventional fit measurements. In stock indices forecasting, improved decisions often depend on correct forecasting directions or turning points between the actual and predicted values (T_i and $T_i^{'}$). The ability to forecast movement direction or turning points can be measured by a statistic of directional change (DC), which can be expressed in percentage as

$$DC = \left(\sum_{i=1}^{N-1} \frac{(T_{i+1} - T_i)(T_{i+1}^{'} - T_i) \geq 0}{N-1} \right) \times 100\% \tag{39}$$

where T_i is the actual value at time t, $T_{i+1}^{'}$ is the prediction at time $t+1$. $(T_{i+1} - T_i)(T_{i+1}^{'} - T_i) \geq 0$ is a logical expression.

However, the real aim of forecasting is to obtain profits based on prediction results. To provide a more complete evaluation of the models, our comparison is based on not only the performance statistics but also the trading returns. Here the return rate is introduced as an important evaluation indicator, which is calculated according to the simple principle ignoring the friction costs.

$$MR = \left((AR - IR)\frac{P}{N} \right) \times 100\% \tag{40}$$

where MR is the P periods excess return rate relative to the tested stock indices, AR the amount of the return rate obtained on the entire period of testing set, IR the return rate of the tested stock indices on the entire period of testing set, and N is the number of the testing periods. For convenient computing, we assume that stock can only be bought in a given lot size. It is worth noting that computation of MR is based on the trading strategy, as in the following:

If ($T_{i+1}^{'} - T_i$)>0, then " buy", else " sell".

The difference between the predicted value and the actual value will guide trading. Because the MAE, MAPE and RMSE measure predictions only in terms of levels, it is better to choose DC and every period return rate (MR) as the measurements for forecast evaluation. Of course, MAE, MAPE and RMSE are also taken into consideration for comparison of levels.

3.4. *Set parameters for the ensemble model*

The architectures and parameters of base models are optimized by IPSO. The parameters of metamodel, SVM, are determined by k-fold cross-validate (CV). The initial parameters of IPSO-NN model are defined as: the population size of particle swarm is 30, the maximum iteration number is 150, the training times of NN are 1500; in DePSO the initial particle positions are random numbers ranging from -15 to 15 and the initial particle velocity randomly varies between -8 and 8, c_{1start}=2.95, c_{1end}=1.05, w_{min}=0.2, w_{max}=0.8; in BiPSO, the initial particle positions are random numbers ranging from -1 to 1 and the initial particle velocity is randomly chosen from the range between -0.6 and 0.6, c_1=c_2=1.14, w=1; in genetic algorithm $P_{c,min}$=0.5, $P_{c,max}$=0.9, $P_{m,min}$=0.02, and $P_{m,max}$=0.6.

3.5. *Time dependency*

The daily data of the stock indices possess the time dependency, therefore, the lag order of the time series

need be determined before prediction. A non-linear time dependency of size (lag) p is increased from 1 to 36 in step from 1 lag. For each test, the networks are trained on the training sets until a MSE from 2×10^{-5} or less is reached. The best performing IPSO-NN on MSE for the lag order is 3 by testing the different performance of the lag orders between 1 to 36. The process is modeled with lag 3, and the realization at $t+1$ is dependent on the realizations of the last 3 trading days.

4. Simulation and prediction

The daily stock indices data of Shanghai composite index, Shenzhen component index and Shanghai-Shenzhen 300 index are pre-processed between 0 and 1 and passed to the ensemble model as non-stationary data. Fig. 2-4 illustrate respectively the prediction of the daily stocks data of Shanghai composite index, Shenzhen component index and Shanghai-Shenzhen 300 index using three single base models (i.e., Elman network, GRNN and WNN), four linear ensemble methods (i.e., simple averaging, simple MAE, simple MAPE and simple RMSE), and the proposed non-linear ensemble model on out-of-sample data, showing the best result achieved. Tables 2–6 show respectively the simulation results for the average performance of the three single base models (i.e., Elman network, GRNN and WNN), four linear ensemble methods (i.e., simple averaging, simple MAE, simple MAPE and simple RMSE), and the proposed non-linear ensemble model when executed 20 times from different perspectives. From the graphs and tables, we can generally see that the forecasting results are very promising for three stock indices under study either where the measurement of forecasting performance is goodness of fit such as MAE, MAPE and RMSE (refer to Table 2-4) or where the forecasting performance criterion are DC (refer to Table 5) and MR (refer to Table 6).

Subsequently, the forecasting performance comparisons of three single base models (i.e., Elman network, GRNN and WNN), four linear ensemble methods (i.e., simple averaging, simple MAE, simple MAPE and simple RMSE), and the proposed non-linear ensemble model for the three stock indices via MAE, MAPE, RMSE, DC and MR are reported in Tables 2–6, respectively.

Table 2-4 shows that: (a) the prediction performance of single base models is unstable to different data set. For

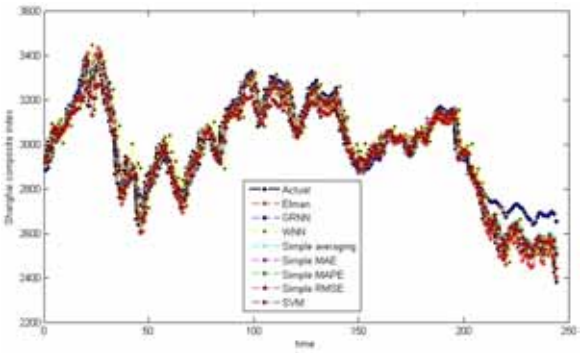

Fig. 2. The prediction of the daily stock index data of Shanghai composite index using three single base models, four linear ensemble methods, and the proposed non-linear ensemble model.

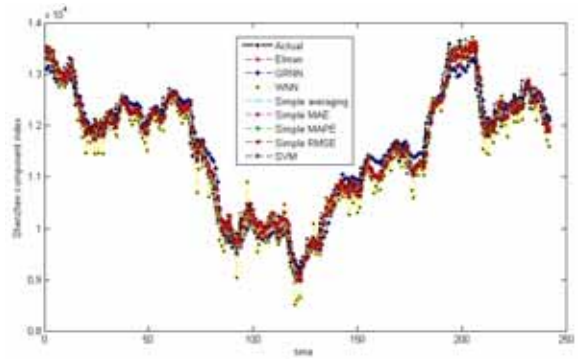

Fig. 3. The prediction of the daily stock index data of Shenzhen component index using three single base models, four linear ensemble methods, and the proposed non-linear ensemble model.

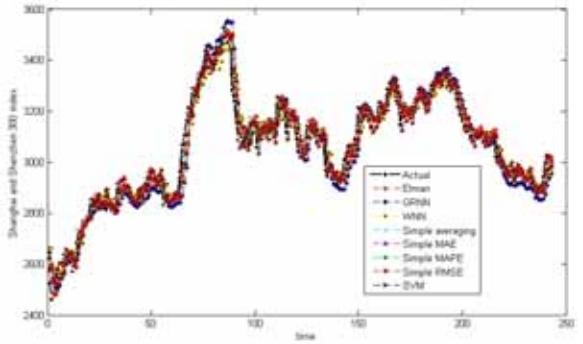

Fig. 4. The prediction of the daily stock index data of Shanghai-Shenzhen 300 index using three single base models, four linear ensemble methods, and the proposed non-linear ensemble model.

example, the best base model in Shanghai composite index, WNN, but is worst in Shenzhen component index; (b) the prediction performance of the linear ensemble models is usually better and more stable than three single base models in all cases, which confirms the ensemble approach to forecasting can effectively reduce errors and provide better performance than single model. Among four linear ensemble models, none can consistently outperform other linear models. For example, although simple MAPE is the best in Shanghai-Shenzhen 300 index, it is worse than simple MAE and simple RMSE in the Shanghai composite index testing case. The main reason is that every linear ensemble model has its own advantages and disadvantages; (c) from the rank of MAE, MAPE and RMSE indicators, the prediction performance of the proposed nonlinear ensemble model is mostly best (except MAPE in Shenzhen component index test case), which indicates the proposed nonlinear ensemble model can improve the performance. The main reason is that the proposed nonlinear ensemble model can capture some nonlinear patterns hidden in financial time series, while linear weighted models cannot.

Table 2. Comparison of performance statistics averaging each model over 20 runs in Shanghai composite index.

Model type	Model	Shanghai composite index					
		MAE	Rank	MAPE(%)	Rank	RMSE	Rank
Single Model	Elman	43.6381	7	1.4786	7	56.9343	7
	GRNN	73.0838	8	2.5516	8	91.6373	8
	WNN	73.8327	4	2.5048	6	93.1843	6
Ensemble Model	Simple averaging	54.6768	6	1.8642	5	68.8027	5
	Simple MAE	50.8187	3	1.7268	3	64.0136	2
	Simple MAPE	50.7134	5	1.723	4	63.8881	4
	Simple RMSE	51.074	2	1.736	2	64.3298	3
	SVM	47.3801	1	1.5853	1	60.7362	1

GRNN: generalized regression neural network; WNN: wavelet neural network; SVM: support vector machines; MAE: mean absolute error; MAPE: mean absolute percentage error; RMSE: root mean squared error.

Table 3. Comparison of performance statistics averaging each model over 20 runs in Shenzhen component index.

Model type	Model	Shenzhen component index					
		MAE	Rank	MAPE(%)	Rank	RMSE	Rank
Single Model	Elman	193.1589	6	1.7119	6	266.1093	6
	GRNN	237.0859	7	2.0761	7	292.9053	7
	WNN	259.0489	8	2.2568	8	325.9413	8
Ensemble Model	Simple averaging	187.9364	5	1.6357	5	239.2687	4
	Simple MAE	184.4873	2	1.6069	2	234.388	2
	Simple MAPE	186.4845	4	1.5928	1	238.3945	3
	Simple RMSE	184.9471	3	1.6159	4	244.7991	5
	SVM	170.8462	1	1.6107	3	225.8467	1

GRNN: generalized regression neural network; WNN: wavelet neural network; SVM: support vector machines; MAE: mean absolute error; MAPE: mean absolute percentage error; RMSE: root mean squared error.

However, the less MAE, MAPE and/or RMSE don't affirmatively represent a high hit rate of forecasting direction for stock indices movement direction prediction. Therefore, the comparison of the directional change statistic (DC) is significantly. From Table 5, we can see the proposed nonlinear ensemble model also performs much better than the other models by the rank. Furthermore, DC is more important than MAE, MAPE and/or RMSE because the former is more useful for the business practitioners' decision. Focusing on Table 5, the

disparities between the eight models are very obvious. For example, in the Shanghai composite index case, the DC for the best single base model, Elman network, is only 58.38%, and for the best linear ensemble model, simple MAPE, DC is 64.54%; while for the proposed nonlinear ensemble model, DC reaches 70.44%. To summarize, the linear ensemble model can model stock indices time series well which contain high noise and nonlinearity, further, the proposed nonlinear ensemble model possesses better performance than the traditional linear ensemble model.

Table 4. Comparison of performance statistics averaging each model over 20 runs in Shanghai-Shenzhen 300 index.

Model type	Model	Shanghai-Shenzhen 300 index					
		MAE	Rank	MAPE(%)	Rank	RMSE	Rank
Single Model	Elman	39.3153	5	1.425	6	51.0497	5
	GRNN	45.1459	8	1.5078	7	56.7412	7
	WNN	42.6522	7	1.5403	8	59.3881	8
Ensemble Model	Simple averaging	39.9869	6	1.3127	5	51.6785	6
	Simple MAE	38.1156	3	1.3042	4	48.8829	3
	Simple MAPE	37.6186	2	1.2843	3	46.887	2
	Simple RMSE	38.7889	4	1.2333	2	49.8461	4
	SVM	33.0141	1	1.0865	1	41.6505	1

GRNN: generalized regression neural network; WNN: wavelet neural network; SVM: support vector machines; MAE: mean absolute error; MAPE: mean absolute percentage error; RMSE: root mean squared error.

Table 5. Comparison of performance of DC averaging each model over 20 runs.

Model type	Model	Stock indices					
		Shanghai composite index		Shenzhen component index		Shanghai-Shenzhen 300 index	
		DC(%)	Rank	DC(%)	Rank	DC(%)	Rank
Single Model	Elman	58.38	6	57.30	8	47.47	7
	GRNN	56.62	7	60.20	6	46.11	8
	WNN	53.50	8	60.28	5	49.79	6
Ensemble Model	Simple averaging	61.67	5	59.37	7	51.35	5
	Simple MAE	63.50	4	62.52	3	52.17	4
	Simple MAPE	64.54	2	60.79	4	52.67	3
	Simple RMSE	63.78	3	62.98	2	54.78	2
	SVM	70.44	1	69.72	1	63.47	1

GRNN: generalized regression neural network; WNN: wavelet neural network; SVM: support vector machines; MAE: mean absolute error; MAPE: mean absolute percentage error; RMSE: root mean squared error.

Table 6. Comparison of performance of MR averaging each model over 20 runs.

Model type	Model	Stock indices					
		Shanghai composite index		Shenzhen component index		Shanghai-Shenzhen 300 index	
		MR(%)	Rank	MR(%)	Rank	MR(%)	Rank
Single Model	Elman	5.55	7	-1.36	8	-1.05	6
	GRNN	4.84	8	2.61	6	-4.09	8
	WNN	5.94	6	1.85	7	-2.63	7
Ensemble Model	Simple averaging	6.35	5	5.23	5	1.67	4
	Simple MAE	7.82	4	6.89	4	2.64	3
	Simple MAPE	9.04	2	7.51	3	3.29	2
	Simple RMSE	8.28	3	7.72	2	1.47	5
	SVM	12.83	1	10.6	1	6.52	1

GRNN: generalized regression neural network; WNN: wavelet neural network; SVM: support vector machines; MAE: mean absolute error; MAPE: mean absolute percentage error; RMSE: root mean squared error.

Considering return rate, the empirical results show that the proposed nonlinear ensemble model could be well forecast future variation of stock indices. Compared with the other models such as three single base models and four linear ensemble models, the proposed nonlinear ensemble model belongs to the best forecasting effect. Interestingly, you can see that the rank of Table 6 is the similar to that of Table 5 because the right forecasting to direction often leads to high return rates. As shown in Table 6, for the Shenzhen component index case, the return rate for the best single base model, GRNN, is 2.61%, and the return rate for the best linear ensemble model, simple RMSE, is also 7.72%; however the return rate for the proposed nonlinear ensemble model reaches 10.6%.

From the experiments presented in this study we can draw the following conclusions: (i) The experimental results show that the proposed nonlinear ensemble forecasting model is superior to four linear ensemble models which are superior to three single base models for the test cases of three stock indices in terms of the measurement of annual return rate (MR), as can be seen from Tables 6. Likewise, the proposed nonlinear ensemble model also outperforms other models in terms of goodness-of-fit or MAE, MAPE and RMSE (refer to Figs. 2-4 and Table 2-4). (ii) MAE, MAPE and RMSE are the metrics used to estimate the error of prediction, however, it don't affirmatively represent a high return rate for stock indices forecasting. For example, in the Shenzhen component index test case, the simple MAPE model is the best in terms of the MAPE (refer to Tables 3), but it is worse than the proposed nonlinear ensemble model in the return rate (refer to Tables 6). Similarly, the indicator DC and MR have a strong positive relationship which isn't absolute. For example, in the Shanghai-Shenzhen 300 index test case, in all linear ensemble models the simple RMSE model is the best, concerning DC (refer to Tables 5), but is the worst in the return rate (refer to Tables 6). (iii) The proposed nonlinear ensemble model can be used as an alternative tool for stock indices forecasting to obtain greater forecasting accuracy and improve the prediction quality further in view of empirical results.

5. Conclusions

In this study we hope to design a model that can provide the most accurate prediction of stock indices. In order to overcome the drawbacks of the traditional NN, a three-stage neural-network-based nonlinear weighted ensemble model is proposed. In this model, three neural-network base models, i.e., Elman, GRNN and WNN are generated by three different training sets, further, they are optimized by improved particle swarm optimization (IPSO) with adaptive nonlinear inertia weight, dynamic arccosine function acceleration parameters and the crossover and mutation operation of GA. Finally, a neural-network-based nonlinear weighted meta-model be produced by learning three neural-network base models through SVM neural network. By applying daily data to these models and comparing the prediction results based on MAE, MAPE, RMSE, DC and MR, we find that in general the annual return rate of the proposed nonlinear ensemble model is better than single base models and the linear ensemble models for forecasting stock indices with high volatility and noise. The result of this paper may be helpful for day-ahead price forecasting of electricity markets. It would be interesting to investigate the optimization of the proposed model e.g., by inclusion of more efficient NNs and effective stochastic search techniques but this will be left for future research.

6. Acknowledgments

This research was supported by the Humanities and Social Sciences Youth Foundation of the Ministry of Education in China under Grant No.11YJC870028, the Special Fund for Basic Scientific Research of Central Colleges under Grant No.CCNU10A01031, the Natural Science Foundation of China under Grant No. 71101100 and 71001096, New Teachers' Fund for Doctor Stations, Ministry of Education under Grant No. 20110181120047, China Postdoctoral Science Foundation under Grant No. 2011M500418 and 2012T50148, Research Start-up Project of Sichuan University under Grant No. 2010SCU11012 and the Center for Forecasting Science of the Chinese Academy of Sciences and the National Center for Mathematics and Interdisciplinary Sciences in China.

References

1. Y. Xiao, J. Xiao and S. Y. Wang, A hybrid forecasting model for non-stationary time series: an application to container throughput prediction, *International Journal of Knowledge and Systems Science*. **3**(2) (2012) 67–82.

2. Y. Xiao, J. Xiao and S. Y. Wang, A hybrid model for time series forecasting, *Human Systems Management*. **31**(2) (2012) 133–143.
3. Y. Xiao, J. Xiao, K. K. Lai and S. Y. Wang, A nonlinear neural networks ensemble model for nonstationary financial market trend mining, *IEEE Transactions on Neural Networks & Learning Systems*. (2012), Forthcoming.
4. Y. Xiao, M. Xiao and F. Z. Zhao, Improving financial returns using neural networks and adaptive particle swarm optimization, in: *Proc.5th Int. Conf. Business Intelligence and Financial Engineering*, (Lanzhou, China, 2012), Forthcoming.
5. Y. Xiao, J. Xiao and S. Y. Wang, A multiscale modeling approach incorporating ARIMA and ANNs for financial market volatility forecasting, in: *Proc. Int. Conf. Forecasting Economic and Financial Systems*, (Beijing, China, 2012), Forthcoming.
6. D. Brownstone, Using percentage accuracy to measure neural network predictions in Stock Market movements, *Neurocomputing*. **10** (1996) 237–250.
7. T. S. Quah and B. Srinivasan, Improving returns on stock investment through neural network selection, *Expert Systems with Applications*. **17** (1999) 295–301.
8. R. J. Kuo, C. H. Chen and Y. C. Hwang, An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network, *Fuzzy Sets Systems*. **118** (2001) 21–45.
9. D. Plikynas, L. Simanauskas and S. Buda, Research of neural network methods for compound stock exchange indices analysis, *Informatica-Lithuan*. **13** (2002) 465–484.
10. S. I. Ao, Automating stock prediction with neural network and evolutionary computation, *Lecture Notes in Computer Science*. **2690** (2003) 203–210.
11. C. Slim, Forecasting the volatility of stock index returns: A Stochastic neural network approach, *Lecture Notes in Computer Science*. **3045** (2004) 935–944.
12. N. O'Connor and M. G. Madden, A neural network approach to predicting stock exchange movements using external factors, *Knowledge-Based Systems*. **19** (2006) 371–378.
13. N. S. Thomaidis, V. S. Tzastoudis and G. D. Dounias, A comparison of neural network model selection strategies for the pricing of S&P 500 stock index options, *Int. J. on Artificial Intelligence Tools*. **16** (2007) 1093–1113.
14. Z. Liao and J. Wang Forecasting model of global stock index by stochastic time effective neural network, *Expert Systems with Applications*. **37** (2010) 834–841.
15. E. Guresen, G. Kayakutlu and T. U. Daim, Using artificial neural network models in stock market index prediction, *Expert Systems with Applications*. **38** (2011) 10389–10397.
16. M. Jasemi, A. M. Kimiagari and A. Memariani, A modern neural network model to do stock market timing on the basis of the ancient investment technique of Japanese Candlestick, *Expert Systems with Applications*. **38** (2011) 3884–3890.
17. E. Grimaldi, A. F. Grimaccia, M. Mussetta and R.E. Zich, PSO as an effective learning algorithm for neural network applications, in: *Proc.3rd Int. Conf. Computational Electromagnetics and Its Applications*, (Beijing, China, 2004), pp. 557–560.
18. A. Alfi and H. Modares, System identification and control using adaptive particle swarm optimization, *Applied Mathematical Modelling.* **35** (2011), 1210–1221.
19. P. C. Wang and T. E. Shoup, A poly-hybrid PSO optimization method with intelligent parameter adjustment, *Advances in Engineering Software.* **42** (2011), 555–565.
20. A. Abraham and A. AuYeung, Integrating ensemble of intelligent systems for modeling stock indices, *Lecture Notes in Computer Science.* **2687** (2003),774-781
21. Y. K. Kwon and B. R. Moon, Evolutionary ensemble for stock prediction, *Lecture Notes in Computer Science.* **3103** (2004),1102-1113
22. S. H. Chun and Y. J. Park, Dynamic adaptive ensemble case-based reasoning: application to stock market prediction, *Expert Systems with Applications.* **28**(3) (2005), 435-443
23. Y. H. Chen, B. Yang and A. Abraham, Flexible neural trees ensemble for stock index modeling, *Neurocomputing.* **70**(4-6) (2007),697-703
24. C. H. Aladag, E. Egrioglu and U. Yolcu, Forecast Combination by Using Artificial Neural Networks, *Neural Processing Letters.* **32**(3) (2010), 269-276
25. J. C. Hung, Applying a combined fuzzy systems and GARCH model to adaptively forecast stock market volatility, *Applied Soft Computing.* **11**(5) (2011), 3938-3945
26. J. T. Yao and C. L. Tan, A case study on using neural networks to perform technical forecasting of forex, *Neurocomputing.* **34** (2000), 79–98.
27. L. Breiman, Bagging predictors, *Machine Learning.* **26** (1996), 123 − 140.
28. Y. Raviv and N. Intrator, Bootstrapping with noise: an effective regularization technique, *Connection Science.* **8** (1996), 355 − 372.
29. W. S. McCulloch and W. Pitts, A logical calculus of the ideas imminent in nervous activity, *Bulletin and Mathematical Biophysics.* **5** (1943), 115–133.
30. J. L. Elman, Finding Structure in Time, *Cognitive Science.* **14** (1990), 179-211.
31. D. T. Pham and X. Liu, Dynamic system identification using partially recurrent neural networks, *Journal of Systems Engineering.* **2**(2) (1992), 90–97.
32. D. F. Specht, A general regression neural network, *IEEE Transactions on Neural Networks.* **2** (1991), 568–576.
33. E. Parzen, On estimation of a probability density function and mode, *Annals of Mathematical Statistics.* **33** (1962), 1065–1076.

34. I. Daubechies, *Ten Lectures on Wavelet.* (SIAM, Philadelphia, 1992).
35. K. Hornik, M. Stinchocombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks.* **2** (1989), 359–366.
36. X. Zhang, J. Qi, R. Zhang, et al. Prediction of programmed-temperature retention values of naphthas by wavelet neural network, *Journal of Computational Chemistry.* **25** (2001), 125.
37. C. W. Jiang and B. Etorre, A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimization, *Mathematics and Computers in Simulation.* **68**(1) (2005), 57–65.
38. S. F. Fan and E. Zahara, Hybrid simplex search and particle swarm optimization for unconstratined optimization problems, *European Journal of Operational Research.* **181**(2) (2007), 527–548.
39. C. Hamzacebi and F. Kutay, Continuous functions minimization by dynamic random search technique, *Applied Mathematical Modelling.* **31**(10) (2007), 189–198.
40. A. Ratnawecra and S. Halgamuge, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, *Evolutionary Computation.* **8**(3) (2004), 240–255.
41. AP. Engelbrecht, *Fundamentals of computational swarm intelligence.* (Wiley, Hoboken, 2005).
42. X. Huang, J. Zhang and Z. H. Zhan, Faster particle swarm optimization with random inertia weight, *Computer Engineering and Design.* **30**(3) (2009), 647–650.
43. G. M.Chen, J. Y. Jia and Q. Han, Study on the strategy of decreasing inertia weight in particle swarm optimization algorithm, *Journal of Xi'an Jiaotong University.* **40**(1) (2006), 53–56.
44. J. A. Benediktsson, J. R. Sveinsson, O.K. Ersoy and P.H. Swain, Parallel consensual neural networks, *IEEE Transactions on Neural Networks.* **8** (1997) 54 – 64.