Applied Mathematics and Nonlinear Sciences

<https://www.sciendo.com>

Back propagation mathematical model for stock price prediction

Yanran Ma^{1†}, Nan Chen², Han Lv¹

¹ School of Economics and Finance, Xi'an International Studies University, Xi'an, Shaanxi 710128, P.R. China

² School of Management, Northwestern Polytechnical University, Xi'an, Shaanxi 710114, P.R. China

Submission Info

Communicated by Juan Luis García Guirao

Received June 17th 2021

Accepted September 24th 2021

Available online December 30th 2021

Abstract

Due to the extremely volatile nature of financial markets, it is commonly accepted that stock price prediction is a task filled with challenges. However, in order to make profits or understand the essence of equity market, numerous market participants or researchers try to forecast stock prices using various statistical, econometric or even neural network models. In this work, we survey and compare the predictive power of five neural network models, namely, back propagation (BP) neural network, radial basis function neural network, general regression neural network, support vector machine regression (SVMR) and least squares support vector machine regression. We apply the five models to make price predictions for three individual stocks, namely, Bank of China, Vanke A and Guizhou Maotai. Adopting mean square error and average absolute percentage error as criteria, we find that BP neural network consistently and robustly outperforms the other four models. Then some theoretical and practical implications have been discussed.

Keywords: back propagation, stock price prediction, radial basis function, support vector machine regression, least squares support vector machine regression

AMS 2010 codes: 92B20

1 Introduction

Price prediction in equity markets is of great practical and theoretical interest. On the one hand, relatively accurate prediction brings maximum profit to investors. Many market participants, especially institutional ones, spend a lot of time and money to collect and analyse relevant information before making investment decisions. On the other hand, researchers often use the fact of whether or not the price can be forecast as a tool to check market efficiency. Also, they invent, apply or adjust different models to improve the predictive power. Finding a good method to forecast stock price more accurately will be a topic forever in both the academic field and the financial industry. Equity price prediction is regarded as a challenging task in the financial time series prediction

[†]Corresponding author.

Email address: ma_yanran@126.com

process since the stock market is essentially dynamic, nonlinear, complicated, nonparametric and chaotic in nature [1]. Besides, many macro-economic environments, such as political events, company policies, general economic conditions, commodity price index, interest rates, investor expectations, institutional investors choices and psychological factors of investors, are also the influencing factors [2]. In this paper, we apply five artificial intelligence (AI) models in the predicting research. Among the AI models, the back propagation neural networks (BPNN), radial basis neural networks (RBFNN), general regression neural network (GRNN), support vector machine regression (SVMR) and least squares support vector machine regression (LS-SVMR) are the most widely used and mature methods. The BPNN is successfully used in many fields, such as engineering [3], power forecasting [4], time series forecasting [5], stock index forecasting [6] and stock price variation prediction [7]. BPNN is also useful in the economic field. Lu and Bai [8] proposed a hybrid forecasting model [Wavelet Denoising-based Back Propagation (BP)], which firstly decomposed the original data into multiple layers by wavelet transform, and then established BPNN model using the low-frequency signal of each layer for predicting the Shanghai Composite Index (SCI) closing price. The radial basis function neural network (RBFNN) is a feed forward neural network with a simple structure, which has a single hidden layer. Miller et al. [9] applied RBFNN as a tool for nonlinear pattern recognition to correct the estimation error in the prediction of linear models in predicting two stock series in Shanghai and Shenzhen stock exchanges. In Osuna's study [10], the author demonstrated RBFNN's effectiveness in financial time series forecasting. RBFNN is proposed to overcome the main drawback of BPNN – namely, that of easily falling into local minima in the training process. RBFNN have also been used in various forecasting areas and achieve good forecasting performance, with demonstrated advantages over BPNN in some applications [11]. The GRNN, which is put forward by Specht [12], shows its effectiveness in pattern recognition [13], stock price prediction [14] and groundwater level prediction [15]. Tan et al. [16] showed the forecasting ability of GRNN in the prediction of closing stock price. However, their research is characterised by a lack of comparison with other data mining models, which is also the limitation of other references cited in this paper. Support Vector Machine (SVM), first developed by Vapnik [17], is based on statistical learning theory. Owing to its successful performance in classification tasks [18] and regression tasks, especially in time series prediction and finance-related applications, SVM has drawn significant attention and thus earned intensive study. By using the structural risk minimisation principle to turn the solving process into a convex quadratic programming problem, the SVM obtains better generation performance; and moreover, the solution is unique and globally optimal. The LS-SVMR, based on structural risk minimisation principle, is able to approximate any nonlinear system. As a reformulation of the SVM algorithm, LS-SVMR overcomes the drawbacks of local optima and overfitting in the traditional machine learning algorithm. To the best of our knowledge, there is a dearth in the literature of studies that are focused on comparing the effectiveness of the above-mentioned five algorithms reviewed in this paper. In this study, we present this comparative view by using data to compare the performance of these five neural networks, namely BPNN, RBFNN, GRNN, SVM and LS-SVMR, in predicting stock price. This paper is organised as follows. Section 2 XXX. Section 3 XXX.

2 Methodology

2.1 BP neural networks

A neutral network generally contains one input layer, one or many hidden layers and one output layer. Supposing that the total number of layers is L , we use l to indicate a single layer. $l = 1$ corresponds to the input layer, $l = 2, \dots, L-1$ corresponds to the hidden layers and $l = L$ corresponds to the output layer. For example, Figure 1 shows a neutral network containing only one hidden layer, which means $L = 3$. Each layer contains one or many neurons; in Figure 1, the input layer contains three neurons, the hidden layer contains four neurons and the output layer contains only one neuron.

We use w_{jk}^l to denote the weight for the connection from the k -th neuron in the $(l-1)$ -th layer to the j -th neuron in the l -th layer. For illustration, we list some weights on the arrows in Figure 1. According to our

Fig. 1 Structure of a three-layer neural network

notation, $w_{11}^2 = 2$, $w_{43}^2 = 4$, $w_{11}^3 = 2$, $w_{12}^3 = 3$, $w_{13}^3 = 4$, $w_{14}^3 = 5$. Explicitly, we use b_j^l for the bias of the j -th neuron in the l -th layer. And we use a_j^l for the activation of the j -th neuron in the l -th layer. With these notations, the activation a_j^l of the j -th neuron in the l -th layer is related to the activations in the $(l-1)$ th layer by the equation

$$a_j^l = \sigma \left(\sum_k \omega_{jk}^l a_k^{l-1} + b_j^l \right) \quad (1)$$

where the sigmoid function is defined as $\sigma(z) = \frac{1}{1+\exp(-z)}$.

To rewrite this expression in a matrix form, we define a weight matrix w^l for each layer l . The entries of the weight matrix w^l are just the weights connecting to the l -th layer of neurons, that is, the entry in the j -th row and k -th column is w_{jk}^l . Similarly, for each layer l we define a bias vector b^l . The components of the bias vector are just the values b_j^l , one component for each neuron in the l -th layer. And finally, we define an activation vector a^l whose components are the activations a_j^l .

With these notations in mind, (2.1) can be rewritten in the elegant and compact vectorised form

$$a^l = \sigma(w^l a^{l-1} + b^l) \quad (2)$$

Let z^l be the weighted input to the neurons in layer l , that is

$$z^l = w^l a^{l-1} + b^l \quad (3)$$

The cost function is defined by the following quadratic form

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2 \stackrel{\text{def}}{=} \frac{1}{n} \sum_x C_x \quad (4)$$

We recall that the Hadamard products $\odot t$ between two vectors s, t with the same length is defined by $(s \odot t)_j = s_j t_j$. The intermediate error function is computed as

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad \delta^l = \left((w^{l+1})^T \delta^{l+1} \right) \odot \sigma'(z^l), \quad l \geq 2$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l, \quad \frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l, \quad \frac{\partial C}{\partial b^l} = \delta^l, \quad \frac{\partial C}{\partial w^l} = a^{l-1} \times \delta^l.$$

With learning rate, the weights are learned by

$$w_{jk}^l \longrightarrow w_{jk}^l - \eta \frac{\partial C}{\partial w_{jk}^l}, \quad b_j^l \longrightarrow b_j^l - \eta \frac{\partial C}{\partial b_j^l}.$$

Owing to the additivity of cost over sample, we can adopt the idea of stochastic gradient descent to speed up the learning. To make these ideas more precise, stochastic gradient descent works by randomly picking out a small number m of randomly chosen training inputs. We label those random training inputs $X_1, X_2, \dots, X_m$, and refer to them as a mini-batch. Provided the sample size m is large enough, we expect that the average value of the ∇C_{X_j} will be roughly equal to the average over all ∇C_x , that is

$$\frac{\sum_{j=1}^m \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C \quad (5)$$

Supposing that w_k and b_l denote the weights and biases, respectively, in our neural network, then stochastic gradient descent works by picking out a randomly chosen mini-batch of training inputs, and training with those, we obtain

$$w_k \longrightarrow w_k' = w_k - \frac{\eta}{m} \sum_{j=1}^m \frac{\partial C_{X_j}}{\partial w_k} \quad (6)$$

$$b_l \longrightarrow b_l' = b_l - \frac{\eta}{m} \sum_{j=1}^m \frac{\partial C_{X_j}}{\partial b_l}. \quad (7)$$

Here the sums are over all the training examples X_j in the current mini-batch.

2.2 Radial basis function (RBF) networks

RBF networks typically have three layers: an input layer, a hidden layer with a nonlinear RBF activation function and a linear output layer. Let us suppose that the hidden layer has I neurons, and the i -th neuron centres at c_i with its preferred value w_i . The input can be modelled as a vector of real numbers $x \in \mathbb{R}^n$, and the output of the network is then a scalar function of the input vector $\varphi: \mathbb{R}^n \rightarrow \mathbb{R}$, given by

$$\varphi(x) = \frac{\sum_{i=1}^I w_i \rho(\|x - c_i\|^2)}{\sum_{i=1}^I \rho(\|x - c_i\|^2)} \quad (8)$$

where $\rho(\|x - c_i\|^2) = \exp(-\beta_i \|x - c_i\|^2)$. It is emphasised that the output $\varphi(x)$ derived based on the given input x is the weighted average of w_i with weight $\rho(\|x - c_i\|^2)$. The cost function is defined by the following quadratic form $C = \frac{1}{2n} \sum_x \|y(x) - \varphi(x)\|^2 = \frac{1}{n} \sum_x C_x$.

The parameters w_i, c_i, β_i are selected by decreasing C .

2.3 General regression neural networks

GRNN essentially belongs to radial basis neural networks. GRNN was suggested by D.F. Specht in 1991. Recalling the framework of RBF network, we recollect that the number of neurons in the hidden layer is the same as the sample size n of training data. Moreover, the centre of the i -th neuron is just the i -th sample x_i , and the preferred value w_i is set to be the desired output $y_i = y(x_i)$. Then the output of a new input x is

$$\varphi(x) = \frac{\sum_{i=1}^I y_i \rho(\|x - x_i\|^2)}{\sum_{i=1}^I \rho(\|x - x_i\|^2)} \quad (9)$$

where $\rho(\|x - c_i\|^2) = \exp(-\beta\|x - c_i\|^2)$. It may be emphasised that the output derived based on the input x is just the weighted average of y_i with weight $\rho(\|x - x_i\|^2)$. We remark that GRNN directly produces a predicted value without training process. One only needs to select a suitable smoothing parameter to implement GRNN.

If the number of neurons in the hidden layer stayed at the sample size n of training data on each prediction, we call it a static GRNN. Instead, as new observations come, we may increase the number of neurons. We call such a pattern as a dynamic GRNN. We choose a dynamic GRNN in the current paper since it has stronger prediction power. A dynamic GRNN has a long memory and updates in a timely manner.

2.4 Support vector regression

A version of SVM for regression (SVR) was proposed in 1996 by Vladimir N. The Vapnik et al. model aims to find a linear function of x to predict y , namely,

$$f(x) = w \cdot x + b \quad (10)$$

Let ε be error acceptance. Then one wants to find optimal w and b such that

$$\sum_i^n |f(x_i) - y_i| \leq \varepsilon \quad (11)$$

A modified problem is to solve

$$\min \frac{1}{2} \|w\|^2, s.t. \|Xw + bI_n - Y\| \leq \varepsilon \quad (12)$$

Here $Y = (y_1, \dots, y_n)^T$ and X is a matrix with its transpose $X^T = (x_1, x_2, \dots, x_n)^T$, the i -th column of which is just x_i .

The linear predictor (2.5) cannot reveal a possible nonlinear relation between x and y , and thus manifests its limitation. For a general (thus possibly nonlinear) function $\phi: \mathbb{R}^m \rightarrow \mathbb{R}^m$, with m being the dimension of x , one may adopt the following predictor

$$f(x) = w^T \phi(x) + b \quad (13)$$

Different functions correspond to different kernel functions in application. The subsequent subsection provides information on more kernel functions.

2.5 Least squares SVM

For some function $\phi: \mathbb{R}^m \rightarrow \mathbb{R}^m$, let us suppose that we instead use $f(x) = w^T \phi(x) + b$ to predict $y(x)$. Replacing the inequality constraint by an equality constraint, the least squares SVMR reads

$$\min \frac{1}{2} \|w\|^2 + \frac{\gamma}{2} \|\xi\|^2, s.t. \phi(X)w + bI_n - Y = \xi \quad (14)$$

Here $\phi(X) = (\phi(x_1), \dots, \phi(x_n))^T$. Compared to SVM, LS-SVM requires less computation cost.

The solution of LS-SVM regression will be obtained after we construct the Lagrangian function:

$$L(w, b, \xi, \lambda) = \frac{1}{2} \|w\|^2 + \frac{\gamma}{2} \|\xi\|^2 + \lambda^T (\phi(X)w + bI_n - Y - \xi) \quad (15)$$

$$\begin{cases} \frac{\partial L}{\partial w} = 0 \Rightarrow w = \phi(X)^T \lambda = \sum_i^n \lambda_i \phi(x_i); \\ \frac{\partial L}{\partial b} = 0 \Rightarrow I_n^T \lambda = \sum_i^n \lambda_i = 0; \\ \frac{\partial L}{\partial \xi} = 0 \Rightarrow \lambda = \gamma \xi; \\ \frac{\partial L}{\partial \lambda} = 0 \Rightarrow \phi(X)w + bI_n - Y = \xi. \end{cases} \quad (16)$$

$$\begin{pmatrix} 0 & I_n^T \\ I_n & \Omega + \gamma^{-1} I_{nn} \end{pmatrix} \begin{pmatrix} b \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ Y \end{pmatrix} \quad (17)$$

where I_{nn} is the $n \times n$ identity matrix and Ω is a $n \times n$ matrix defined by $\Omega_{ij} = \phi(x_i)^T \phi(x_j) = K(x_i, x_j)$. Once the previous equation is solved, the predicted value $f(x)$ for input x is given by $f(x) = \lambda^T \phi(X) \phi(x) + b = \sum_i^n \lambda_i K(x_i, x) + b$.

The kernel function $K(x_i, x_j)$ has many forms:

Linear kernel: $K(x_i, x_j) = x_i^T x_j$,

Polynomial kernel of degree d : $K(x_i, x_j) = (1 + x_i^T x_j / c)^d$,

RBF kernel: $K(x_i, x_j) = \exp(-||x_i - x_j||^2 / \sigma^2)$,

MLP kernel: $K(x_i, x_j) = \tanh(kx_i^T x_j + \theta)$,

where d, c, σ, k, θ are constants. We remark that the linear kernel corresponds to the linear function $\phi(x) = x$. The most commonly used kernel is the RBF kernel.

3 Data and analysis

3.1 Data description

In this work, we study the weekly adjusted closing price of three individual stocks: Bank of China (601988), Vanke A (000002) and Guizhou Maotai (600519). Each price data has a sample size of 427, ranging from 3 January 2006 to 11 March 2018. As usual, we split the whole data set into a training set (80%) and a test set (20%).

We intentionally select the three stocks based on such an observation: they are totally different in price scale. As shown in Table 1, the price of Bank of China is about within 2–5 RMB, Vanke A (000002) is approximately in the range 5–40 RMB and Guizhou Maotai has a wide range of 80–800 RMB. Actually, Guizhou Maotai ranks first in terms of price per share among all stocks listed in the only two stock exchanges of Mainland of China: Shanghai and Shenzhen.

Table 1 Price range

Name	Bank of China	Vanke A	Guizhou Maotai
Lowest price	2.00	5.65	81.13
Highest Price	5.01	40.04	788.42

Let us use $\{S_i\}_{1 \leq i \leq 427}$ to denote the time series of price. We use three previous periods to predict the price of the next period. More precisely, we set $x_i = (S_i, S_{i+1}, S_{i+2})$ and $y_i = S_{i+3}$ for $1 \leq i \leq 424$. Then we regard (x_i, y_i) as one sample. That is, for an input x_i , its desired output is y_i . It may be emphasised that we use weekly data, and thus the information contained in the price is assumed to be effective within 1 month.

3.2 Hyper-parameters

We adopt a neural network with three layers, which contains only one hidden layer. The input layer has three neurons, and the output layer has only one neuron which represents the predicted value. To determine the number of neurons in the hidden layer, by rule of thumb, we apply the following formula

$$m = \sqrt{0.43 \ln + 0.12 l^2 + 2.54 n + 0.77 l + 0.35} + 0.51,$$

where l is the number of neurons in the output layer and n is the number of neurons in the input layer. With $l = 1, n = 3$, we get $m = 3$ after rounding to integer. The learning rate $\eta = 0.01$ is chosen after amounts of tests.

For the implementation of RBF, SVMR and LS-SVMR, we use standard R packages. When applying GRNN, we choose $\beta = 20, 0.5, 0.0005$ for Bank of China, Vanke A and Guizhou Maotai, respectively, which are representative of the different price scales of the three stocks.

3.3 Results

Table 2 shows the performance of the five neural network models. From these results, we can see all the five models have some predictive power. Even the worst one, GRNN, has MAPE not exceeding 5%, which is very satisfactory considering we are forecasting stock price rather than volatility.

Across all the three stocks, in terms of both MSE and MAPE, BP neural network outperforms the other four models. One may refer to Figure 2 in the next subsection for a more intuitive view of the accuracy of prediction for Bank of China using the BP method. SVMR ranks second consistently across the three stocks. However, in terms of both MSE and MAPE, results from SVMR are greater than that of BP by at least 10%. Moreover, on the prediction of Bank of China and Vanke A, BP surpasses SVMR by at least 20% under both criteria.

Fig. 2 Forecast of Bank of China

Table 2 Results of the five methods

Method		BP	RBF	GRNN	SVMR	LS-SVMR
Bank of China	MSE	0.009	0.014	0.02	0.012	0.018
	MAPE	0.019	0.025	0.024	0.023	0.028
Vanke A	MSE	2.976	4.686	6.036	3.422	5.472
	MAPE	0.049	0.065	0.067	0.059	0.072
Guizhou Maotai	MSE	395.1	740.1	1103.6	407.4	405.5
	MAPE	0.026	0.036	0.048	0.029	0.027

BP, back propagation; GRNN, general regression neural network;

LS-SVMR, least squares support vector machine regression;

RBF, radial basis function; SVMR, support vector machine regression.

We cannot tell which one among RBF and LS-SVMR is better. As shown in Table 2, on the prediction of Bank of China and Vanke A, RBF is more accurate than LS-SVMR, while on the prediction of Guizhou Maotai, LS-SVMR has a better performance. Overall, they share a similar accuracy level of prediction. Finally, GRNN behaves the worst consistently across the three stocks.

One reason we could guess for the superior performance of BP over the other methods is that the latter four models all involve the mostly used kernel function: $\exp(-|x|^2)$. To check whether this guess is correct, we use

other kernels to apply the second-best model: SVMR. Table 3 gives the results for the above-mentioned four kernels.

Table 3 Comparison of four kernels in SVMR

Kernel		Linear	Polynomial	Sigmoid	RBF
Bank of China	MSE	0.01	0.01	0.011	0.012
	MAPE	0.019	0.02	0.021	0.023
Vanke A	MSE	2.993	3.292	3.515	3.422
	MAPE	0.05	0.053	0.055	0.059
Guizhou Maotai	MSE	395.6	403.5	405.7	407.4
	MAPE	0.027	0.028	0.028	0.029

RBF, radial basis function; SVMR, support vector machine regression.

We have two remarks on Table 3. On the one hand, linear kernel is the best in this prediction task, and consistently outperforms the other three kernels. Although RBF kernel is the default kernel in many packages due its flexibility to various data resources, it is not good enough here. Thus, we should try other kernels to make comparison when doing similar predicting projects. On the other hand, BP stills surpasses SVMR with linear kernel, even if the advantage is not obvious now. They share a similar prediction error, possibly resultant to the fact they both involve weighted average, which captures some linear relation in the network.

3.4 Two more discussions: stability of BP and market inefficiency

When implementing BP algorithm, it needs to initialise the weights randomly, which causes instability of the result. To show that the result of BP is stable, we train the neural network for 100 times, and compute its mean and standard deviation. Table 4 helps us eradicate this concern since the standard deviations are extremely small compared to the scale of their corresponding mean values. In other words, the result of every experiment is reliable.

Table 4 100 times experiment

Statistics		Mean	Std.
Bank of China	MSE	0.009	4.8×10^{-5}
	MAPE	0.019	0.0001
Vanke A	MSE	2.976	0.0067
	MAPE	0.049	0.0001
Guizhou Maotai	MSE	395.1	1.3728
	MAPE	0.026	7.7×10^{-5}

Figure 2 plots the observed and predicted prices of Bank of China. It can be seen clearly that the predicted values fit the observed ones well. Also, the turning points are forecasted quite timely. When there is a trend in the actual price, the predicted value follows accordingly and closely.

At a first glance, the network needs at least one period to react or assimilate new information. Actually, it is a false appearance that the predicted values lag one period of the observed values. More precisely, supposing that y_t is the observed price, $\hat{y}_t$ the predicted price, then it seems $y_t \approx \hat{y}_t$ from Figure 2, which means the best predicted value is just the price of the previous period. Alternatively, the stock price process is Markovian. If such a phenomenon is true, then the market is efficient and thus unpredictable.

To prove that the market is actually inefficient, we take difference $e_t = y_t - \hat{y}_{t+1}$ and plot the difference series e in Figure 3. It may be emphasised that in Figure 3, the error is not cantered at 0. Actually, it has a bias towards negative values. In other words, on average, $y_t < \hat{y}_{t+1}$, which contradicts market efficiency.

Fig. 3 Lag one error

4 Conclusion

In this work, we have successfully demonstrated that the five neural network models are all able to effectively extract meaningful information from past price. With evidence from the forecast accuracy of three unrelated stocks, we find that BP surpasses the other four models consistently and robustly. Also, by implementing the algorithm many times and checking the standard deviation, the stability of BP is observed and confirmed. Based on our trial on different kernels, we advise readers of the current paper not take the default kernel for granted and ‘descrisized’ other kernels. For our own interest, we test the error series and destroy the market efficiency hypothesis. In our future research, we will investigate other more involved neural networks to complete the current tentative work.

References

- [1] Abu-Mostafa, Y. S., & Atiya, A. F. Introduction to financial forecasting. *Applied intelligence*., 1996 6(3):205-213.
- [2] Han, R., & Cao, Q. Fuzzy chance constrained least squares twin support vector machine for uncertain classification. *Journal of Intelligent & Fuzzy Systems*., 2017 33(5):3041-3049.
- [3] Hsieh, L. F., Hsieh, S. C., & Tai, P. H. Enhanced stock price variation prediction via DOE and BPNN-based optimization. *Expert Systems with Applications*., 2011 38(11):14178-14184.
- [4] S. Ismail and A. Shabri and R. Samsudin, A hybrid model of self-organizing maps (som) and least square support vector machine (lssvm) for time-series forecasting, *Expert Systems with Applications*., 2011 38(8): 10574-10578.
- [5] Abrahart, R. J., & See, L. Comparing neural network and autoregressive moving average techniques for the provision of continuous river flow forecasts in two contrasting catchments. *Hydrological processes*., 2000 14(11-12):2157-2172.
- [6] Li, G., & Shi, J. On comparing three artificial neural networks for wind speed forecasting. *Applied Energy*., 2010 87(7):2313-2320.
- [7] C.J. Lu and J.Y. Wu, An efficient cmac neural network for stock index forecasting, *Expert Systems with Applications*., 2011 38(12):15194-15201.
- [8] J. Lu and Y. Bai, Applications of grnn based on particle swarm algorithm forecasting stock prices, *Perception & Psychophysics*., 2013 30(3):303-307.
- [9] K.R. Mller and A.J. Smola and G. R?tsch and B. Sch?lkopf and J. Kohlmorgen and V. Vapnik, Predicting time series with support vector machines, *Advances in Kernel Methods-Support Vector Learning*., 1999 20(2): 999-1004.
- [10] E.E. Osuna, Support vector machines: training and applications, A. I. Memo no. 1602, C. B. C. L. Paper., 1997 144(9):1308-16.
- [11] W. Pan, Performing stock price prediction use of hybrid model, *Chinese Management Studies*., 2010 4(1):77-86.
- [12] R. Shan and H. Dai and J. Zhao and W. Liu, Forecasting study of Shanghai’s and Shenzhen’s stock markets using a hybrid forecast method, *Communications in Statistics – Simulation and Computation*., 2015 44(4):1066-1077.
- [13] B. Shirmohammadi and M. Vafakhah and V. Moosavi and A. Moghaddamnia, Application of several data-driven techniques for predicting groundwater level, *Water Resources Management*., 2013 27(2):419-432.
- [14] D.F. Specht, A general regression neural network, *IEEE Trans Neural Netw.*, 1991 2(6):568-576.
- [15] F. Sun and C. Liang and L. Zhang and P. Lee and Z. Lin and J. Cao, Optimal partition algorithm of the rbf neural

- network and its application to nancial time series forecasting, *Neural Computing & Applications.*, 2005 14(1): 36-44.
- [16] T.Z. Tan and Q. Chai and G.S. Ng, Biological brain-inspired genetic complementary learning for stock market and bank failure prediction1, *Computational Intelligence.*, 2007 23(2):236C261.
- [17] Wang, J. Z., Wang, J. J., Zhang, Z. G., & Guo, S. P. Forecasting stock indices with back propagation neural network. *Expert Systems with Applications.*, 2011 38(11):14346-14355.
- [18] L. Wang and Y. Zeng and T. Chen, Back propagation neural network with adaptive differential evolution algorithm for time series forecasting, *Expert Systems with Applications.*, 2015 42(2):855-863.