Applications of GRNN Based on Particle swarm algorithm Forecasting Stock Prices

Jinna Lu [1] Yanping Bai [2]

[1] Jinna Lu, graduate students of NUC, research direction: math problems of the computer science. TEL: 13653609632, E-mail: jinna4813@gmail.com, Address: Mailbox 722, 3 xueyuan RD, North University of China, Taiyuan, Shanxi Prov, China, 030051;
2 Yanping Bai, professor, PhD supervisor, E-mail：baiyp666@163.com

Abstract

Generalized regression neural network (GRNN) has very good effect on making nonlinear forecasting model with large number of stock data. Particle swarm optimization (PSO) has simple operation analysis and is easy to implement. We use PSO algorithm to optimize the GRNN in order for optimal smoothing factor and connection weights. The prediction errors of the two models are both small. The MSE error by GRNN model reaches 0.0486, while the error by PSO-GRNN model is 0.0104. The analysis shows that PSO-GRNN model is more accurate, more stabilized and more generic than GRNN model.

Keywords: generalized regression, Particle Swarm Optimization, neural network model, Stock Price Prediction

1. Introduction

Stock prediction is an indispensable part of financial industry and becomes one of the important branches of economic forecasting. There are some stock prediction analysis methods such as securities investment analysis, time series analysis and neural network prediction method [1,2]. Generalized regression neural network (GRNN) is a nonlinear regression based on the theory of feedforward neural network model [3]. It is commonly used in stock prediction of the neural network forecast.

This article proposes the particle swarm algorithm to optimize the GRNN neural network, establishes a PSO-GRNN neural network model, then forecasts three branch glamour stocks of our country, Shanghai electric power (code: 600021), CITIC securities (code: 600030) and China petrochemical (code: 600028) .

1.1. Generalized Regression Neural Network

The generalized regression neural network architecture is put forward by Specht [4], and is composed of input layer, hidden layer and output layer. It is a one-pass learning algorithm with a highly parallel structure [5]. In the hidden layer, the output of the weighted input neuron is calculated through the Gaussian function.

$$u = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

$$d = \sqrt{(x_i - u_i)^T (x_i - u_i)} \tag{2}$$

$$h_i = \exp(-d_i^2/(2b^2)) \tag{3}$$

Where, N is sample size, x is the estimator input vector; u is unit weight vector in hidden layer. d_i is the distance between the input vector x_i and the training vector u_i. b is smoothing factor, decides the shape of basis functions in the hidden layer, b>0.

The GRNN output layer is linear, and the output neurons are calculated as formula:

$$y_i = \frac{\sum_{i=1}^{n} h_i w_{ij}}{s}, \quad s = \sum_{i=1}^{n} h_i \tag{4}$$

Where, w_{ij} is the connection weight corresponding to input training vector x_i and output y_j.

GRNN neural network needs only one simple smoothing factor. The bigger smoothing factor is, the smoother approximation process of the network is; the smaller the smoothing parameter is, the stronger approximation performance of the network is. So, it is needed to try to get the best value continually.

1.2. Particle Swarm Optimizing Algorithm

The particle swarm optimizing algorithm (PSO algorithm) is a parallel evolutionary computation technique put forward by Kennedy and Eberhart [6]. At first, we initialize a population array of particles with random locations $L_i=(l_{i1},l_{id},\ldots,l_{iD})$ and velocities $V_i=(v_{i1},v_{i2},\ldots,v_{iD})$ on D dimensions in the search space. Then we get the optimal solution through the iterative search particles. From initial to the current searching iteration times, the optimal solution of particles is the particle's individual extremism p_{best}. Particle population current optimal solution is called global extremism g_{best}. The particle i through the two extreme updates its l_{id} and v_{id}.

$$v_{id}^{(k+1)} = \hat{w} v_{id}^{(k)} + c_1 r_1 (p_{id}^{(k)} - l_{id}^{(k)}) + c_2 r_2 (g_{id}^{(k)} - l_{id}^{(k)}) \tag{5}$$

$$l_{id}^{(k+1)} = l_{id}^{(k)} + v_{id}^{(k+1)} \tag{6}$$

$$\hat{w} = \hat{w}_1 + a(\hat{w}_1 - \hat{w}_2) + d\hat{w} \tag{7}$$

$$d\hat{w}(j) = \beta * e_j * h_j \tag{8}$$

Where, c_1 and c_2 are learning factors and positive constants. r_1 and r_2 are random numbers within [0, 1]. $\hat{w}$ is inertia weight and between the distribution [0, 1]. $\hat{w}_1$ is the initial inertia weight, $\hat{w}_2$ is the final inertia weight. a and β are momentum factors and can be arbitrary value between [0, 1].

During continually searching g_{best} and p_{best} of particle, in iteration i, the particle's fitness value is f_i. This article gets fitness values according to the error of particle in every iterative.

2. GRNN Based on Particle swarm algorithm

Using PSO algorithm training GRNN neural network, we definite the particle swarm vector elements were smoothing factor σ and GRNN neural network connection weights W.

Algorithm process is as follows:

(1)Extract target input vector, and do the normalized processing as follows.

$$x_j(i) = x_j(i) / \max(x_j(k)), k = 1,2,\cdots N \tag{9}$$

Among it, $x_j(i)$ is the input value. The number of training samples is N.

(2) Randomly initialize population scale to be 20 and particle's position and velocity in D variables. Each l_i and v_i is

kept within the range [$-L_{max}$, L_{max}] and [$-V_{max}$, V_{max}].

(3) Map each individual component of the particle swarm for GRNN neural network parameters, train input training sample to each GRNN neural network, and calculate the mean square error of each particle in the training set as fitness value. Mean square error's formula is as follows:

$$f_i = e = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m}(y_{ij} - \hat{y}_{ij})^2 \quad (10)$$

Where, N is the training sample capacity, m is output node number, y_{ij} represents the training sample actual value, $\hat{y}_{ij}$ represent the output value of prediction model.

(4) Evaluate each particle search position according to each individual's fitness value, calculate the current each particle individual extreme and population global extreme. If the fitness value is better than that of p_{best}, then set p_{best} to be the current value. And if the fitness value is better than that of g_{best}, then set g_{best} to be the current value [8].

(5)Determine whether the requirements are met. If maximum iterations or mean square error is the initial value, then end particle search. Or, turn to (3).

(6) Output the optimal particle position, recording g_{best}, map each individual's component for smoothing factor and weights of GRNN neural network, and forecast the neural network's input data.

(7) End particle search.

3. MATLAB Training and Prediction

We train the three sets of data with PSO-GRNN model and GRNN model respectively. In PSO-GRNN neural network model momentum factor takes α=0.05, β=0.55. Particle population scale is 20, particle dimension is 12, learning factor c_1=c_2=2, the initial inertia weight value $\hat{w}_0 = 0.9$, the iteration number is 250 times. In order to test the generalization ability of the neural network, we introduce three kinds of evaluation index [9]:

Root mean square error:

$$MSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (11)$$

Mean absolute error:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (12)$$

Mean absolute percentage error:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|} \times 100\% \quad (13)$$

Among them, y_i represents the actual value of the training sample and $\hat{y}_i$ represents the output value of the prediction model. *MAE* and *MAPE* are both index of measuring prediction results deviate from the actual values. The bigger *MAE* (or *MAPE*) value is, the greater the prediction error is; *MSE* can better forecast effect, the smaller *MSE* value is, the better prediction effect is.

After training, the contrast results of two kinds of models see in table 1.

Table 1 prediction errors of two models

evaluation index	error	MSE	MAE	MAPE (100%)
CITIC securities	PSO-G	0.0089	0.0056	0.6624
	GRNN	0.0459	0.0191	2.2696
China petrochemical	PSO-G	0.0058	0.0042	0.4808
	GRNN	0.0813	0.0199	2.1944
Shanghai electric power	PSO-G	0.0166	0.0058	0.7399
	GRNN	0.0175	0.0117	1.4856
Average of 3 indexes	PSO-G	0.0104	0.0052	0.6277
	GRNN	0.0486	0.0169	1.9832

Table 1 shows that, in the circumstances of the same initial factor, the prediction precision to the same sample of PSO-GRNN model is obviously superior and stronger generalization ability than the GRNN model.

Follows are prediction error chart and prediction output value chart with two kinds of models:

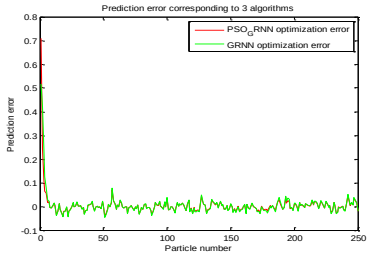

Fig. 1: prediction curve

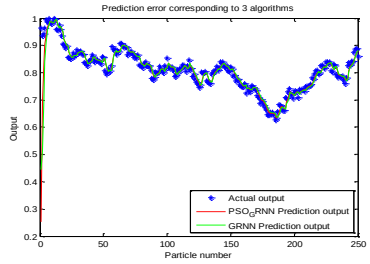

Fig. 2: prediction output value chart

Through the results， we can see that PSO-GRNN model's stability is obviously superior to the GRNN model's stability. Particle swarm optimization (PSO) algorithm improved GRNN neural network, and through the search renewal process for smoothing factor and the right value of the automatic optimization, can achieve better effect. On the basis of the same training samples, PSO-GRNN model has better accuracy, better stability and stronger generalization ability than the GRNN model.

Acknowledgment

The authors are thankful that the research is supported by National Natural Science Foundation of China. (61275120)

References

[1] Fan Qunlin, Li Tao, Wu Huaping (2008). Economic neck measuring model researching based on the generalized regression neural network. *Market Modernization*, 26, 195-196.

[2] Liu Haiyue, Bai Yanping (2011). Analysis of AR model and Neural Network for Forecasting Stock Prices. *Journal of Mathematics in Practice and Theory*, 41(4):14-19.

[3] Zhou Min, Li Shiling (2007). Application of GRNN and uniform design to nonlinear system modeling. *Computer measurement & control*, 15(9), 1189-1191.

[4] Specht, D F. (1991). A general regression neural network. *IEEE Transactions on Neural Network*, 2, 568-576.

[5] Zhang Libiao, Zhou ChunGuang, Ma Ming and Liu Xiaohua (2004). Solutions of Multi- objective optimization Problems based on Particle Swarm optimization. *Journal of computer research and development*, 41(7), 1286-1291.

[6] Kennedy J, Eberhart R C. (1995). Particle swarm optimization. *IEEE International conference on Neural Networks* (pp.1942-1948). Piscataway, NJ, USA: IEEE Press.

[7] R.Karthi, S.Arumugam, K. Rameshkumar (2008). Comparative evaluation of Particle Swarm Optimization Algorithms for Data Clustering using real world data sets. *IJCSNS International Journal of Computer Science and Network Security*, 203-212.

[8] Mark A. Wolters (2012). A particle swarm algorithm with broad applicability in shape-constrained estimation. *Computational Statistics & Data Analysis*, 56(10): 2965-2975.

[9] Cong Shuang. Neural Network Theory and Applications with MATLAB Toolboxes. China science and technology university press, 1998.8.