Jestr Journal of Engineering Science and Technology Review 8 (2) (2015) 116 - 122
Special Issue on Synchronization and Control of Chaos: Theory, Methods and Applications

JOURNAL OF
**Engineering Science and Technology Review**

www.jestr.org

Research Article

# Prediction Accuracy Optimization of Chaotic Perturbation in the Analysis Model of Network-Oriented Consumption

**Dakai Li[*,1] and Li Yu[2]**

[1]*Shandong University of finance and Economics，School of international economics and trade, Jinan, 250014, Shandong, China.*
[2]*Qilu University Of Technology, School of business, Jinan, 250353, Shandong, China*

Received 2 September 2014; Revised 4 October 2014; Accepted 25 October 2014

***Abstract***

As the slower rate of convergence and lower study ability in the late period of network-oriented consumption prediction model based on neural network algorithm, this paper proposed a network analysis neural model based on chaotic disturbance optimized particle swarm. Firstly, improve the initialization of particle swarm with chaotic disturbance optimization strategy in order to limit the initial position and the initial speed of limited particle. Then have an optimal operation on each individual in particle swarm with chaotic disturbance variables, so that the particles which do not enter into iteration will jump out of the local optima area. And next, optimize the PSO algorithm inertia weight by adopting adaptive adjustment strategy based on individual particle adaptive value. At last, combine the improved PSO algorithm based on chaotic disturbance with neural network algorithm, thus we will construct the network-oriented consumption analysis model. Simulation results show that the proposed network-oriented consumption analysis neural network model based on chaotic disturbance optimized particle swarm has greatly improved in prediction accuracy and computational speed.

*Keywords:* Neural Network Algorithm, Chaotic Disturbance, Improved Particle Swarm, Network Analysis of Consumption, Inertia Weight, Preferential Particle.

## 1. Introduction

With the development of future technological and the acceleration of economic growth, the network coverage rate keeps growing, and network consumption will be more popular and become an important part of consumption [1]. Network consumption shows a rapid growth momentum and now it is becoming a new emerging hot consumer spot [2]. The rapid development of network consumption can expand consumption and it is also an important way to boost economic growth [3]. Against this background, researching on our network consumption status and development trend, finding out existing problems in Chinese online consumption, and research countermeasures have a very important meaning in the further development of our network economy and the promotion of an orderly, healthy and fast-growing network consumption [4].

There are a lot of researches on people's consumption behaviour at home and abroad, and they all have a wide range. Fang Huliu selected retail sales' time-series data, made use of the ADF test to analyze fluctuation rule of our social society consumer goods retail sales, and analyze the influencing factors through Granger causality test and VAR model's application [5]. Guo Xiaojin had a comparison between Chinese Internet consumption development and international network consumption development, in which market access, network consumption scale, the degree of organization, and network consumption competitiveness are mainly considered [6]. Liu Yanan established a multiple linear regression model related to our social network consumption expenditure and he made use of eviews software to have a regression analysis on social network consumption value's annual data [7]. Yuan Aifeng analyzed the changes in our network consumption development through sequence analysis method. He concluded that our network consumption spending increases steadily year after year, which is the inevitable result of the development of Chinese productive force [8]. Li Ruowen took Beijing's network consumption spending as explanatory variable, and had analyzed other correlated variables, concluded that Beijing network spending has a great correlation with gross domestic product, per capita disposable income, and population density takes the second place [9]. On the basis of analyzing our network spending data developing status,

* E-mail address: sundarvtu@gmail.com

Zhang Huachu formulated ARIMA model for our network consumption spending. Based on this, he analyzed and predicted the network spending, and the results show that the total amount of our network spending would continue to increase in the next few years [10]. Wang Naihe studied how housing asset prices effects consumer from the influence that Chinese financial system on real estate wealth, housing sales price's change on freegans, and rental price's change on household consumption [11]. Mu Xiaofang mainly researched the impact that our resident network consumption works on the domestic economic development. The author mainly studied from the relationship between the function that resident network consumption works on economic growth and resident network consumption with economic growth's relationship, with the empirical analysis method of non-stationary sequence's unit root test, co integration test and establishing error correction model, whose results showed that long-term stable equilibrium relationship will continuously exist in our residents' network consumption and domestic economic growth [12]. In order to predict the consumption of network, Tang Gongshuang established a seasonal decomposition model and ARIMA model. After comparing the results that analyzed from several different models, she selected a good fitting model and had an analysis on model's practicality [13].

For the issues exist in neural network algorithm application, this paper designed a network consumption analysis neural network model analysis based on chaos disturbance optimized particle swarm, and adopted chaotic disturbance optimized strategy on particle swarm optimization strategy's initial population, particle merit and also improved inertia weight, combined it with neural networks, so that it will be used in network consumption forecasting model.

## 2. Network Consumption Prediction Model based on Neutral Network

Neural network is a new mathematical modeling method, which is more used in prediction system because of its powerful ability to identify nonlinear systems. This paper built network consumption prediction model based on neural network model. Firstly, we will take consumer age, the culture level of consumer, consumer's income level, and cost performance of network products, network security as variables and then input neural network's value, then normalize the data. Please see Equation (1).

$$\hat{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

$x$ is the original network consumption data, $x_{\min}$ and $x_{\max}$ are the minimum and maximum value in original data, $\hat{x}$ is the data after normalization. After data's training and prediction, adapted corresponding data changing method to obtain the original true value data in output layer, corresponding inverter equation is as followed. Please see Equation (2).

$$x = \hat{x} \cdot (x_{\max} - x_{\min}) + x_{\min} \tag{2}$$

Then weighted input network consumption value. Please see Equation (3).

$$net_q = \sum_q T_{qi} y_i - \theta_q \tag{3}$$

Calculate each layer's error, the input layer node error is:

$$\delta_q = -(t_q - C_q) f(net_q) \tag{4}$$

Then we will have Eq. (5).

$$\delta_q = -(t_q - C_q) f(net_q) \tag{5}$$

Hidden layer node error is:

$$\delta_i = f(net_i) \cdot \sum_q \delta_q T_{qi} \tag{6}$$

See Equation (7).

$$\frac{\partial E}{\partial \omega_{in}} = -\beta \delta_i \alpha_n \tag{7}$$

Therefore, network consumption forecasting modelling based on neural network can be seen from the following steps:

1) Determine neurons' number in neural network. The neural network inputs layer is 5 in this paper, and the hidden layer selected by empirical formula is 10, and the output layer, the number of neurons about to be predicted network consumption value, is 1;
2) Input training samples and testing samples, and normalize all collected samples in accordance with equation (1);
3) Select excitation function, namely, network hidden layer and output layer transferring function, and training function, training iteration numbers, the maximum permissible errors and so on.
4) Use neural network to train the training samples, then do the simulation output through testing samples and calculate the variety errors between predicted value and true value.

Through the experiment, the prediction error of the above network consumption forecasting model's is shown in Fig. 1.

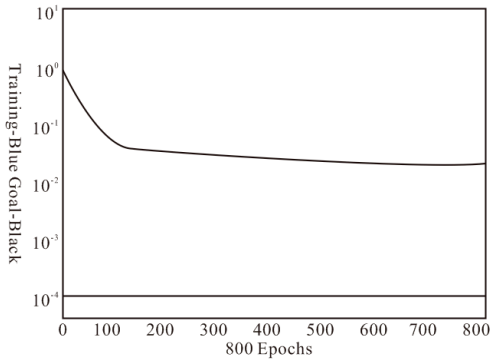


**Fig. 1.** Network consumption based on neural network prediction error.

From the figure of training error, we can see that the traditional neural network algorithm in early learning has a faster convergence, and for the late stage, convergence is getting slower, so that at a certain stage, the learning ability almost drops to zero.

Newton's method is mostly used in the optimization of the neural network. Conventional BP neural network algorithm only uses the first derivative's information in weights correction, namely the gradient of the error function to weight value, so the use of the second derivative to carry out weight adjustment can improve the convergence speed.

Assuming the network weight correction's objective is to minimize error function $E(w)$, the current network value weight is $w(t)$, weight correction amount is $\Delta w(t)$, then the next time the weight is as bellowed. Please see Equation (8).

$$w(t+1) = w(t) + \Delta w(t) \tag{8}$$

To implement the second order Taylor expansion to $E(w(t+1))$, see Equation (9).

$$E(w(t+1)) \approx E(w(t)) + g^T(t)\Delta w(t) + \frac{1}{2}\Delta w^T(t)A^T(t)\Delta w(t) \tag{9}$$

$g(t)$ is $E(w)$ 's gradient vector; square matrix $A(t)$ is Hessian matrix, namely element value is $E(w)$ to each weight second order differential coefficient, $A_{ij}(t) = \frac{\partial^2 E(w)}{\partial w_i \partial w_j}$. After correcting the weight value, error function variation can be seen in Equation (10).

$$\Delta E(t) \approx g^T(t)\Delta w(t) + \frac{1}{2}\Delta w^T(t)A^T(t)\Delta w(t) \tag{10}$$

Through changing $\Delta w(t)$ minimize equation (9), apparently when it meets the equation (11).

$$\Delta w(t) = -A^{-1}(t)g(t) \tag{11}$$

$\Delta E(t)$ obtained minimum, Newton algorithm converges faster, but it is too difficult to calculate the amount. Therefore, we use particle swarm algorithm to optimize neural networks algorithm.

## 3. Neural Network Algorithm Based on Improved Particle Swarm

### 3.1. Chaotic Disturbance Optimized Strategy

Before optimizing neural network through particle swarm algorithm, the first thing is to analysis its defect. Standard particle swarm optimization is in a goal-dimensional search space, we will random initialize a particle swarm and the particle swarm is composed by $m$, the first $i$ 's position $X_i$ (potential solution of the optimization problem) can be expressed as $\{x_{i1}, x_{i2}, ..., x_{iD}\}$, substitutes into optimizing the evaluation function and we can obtain adapted value, which is used to measure the pros and cons of particles; appropriate flight speed $V_i$ can be expressed as $\{v_{i1}, v_{i2}, ..., v_{iD}\}$. In every iteration process, the particles update its velocity and position by tracking the two extremes: one extreme is the the optimal solution researched by particle itself, and individual extreme $P_{ibest}$ expressed as $\{P_{ibest.1}, P_{ibest.2}, ..., P_{ibest.D}\}$; the other extreme value is a group of particles search the optimal solution, and its global minimum $P_{gbest}$ will be expressed as $\{P_{gbest.1}, P_{gbest.2}, ..., P_{gbest.D}\}$.

Specifically, in the calculation of $k+1$ 's first iteration, according to Equation 12 and 13, particle $i$ updates the velocity and position; the speed will be defined as equation. Please see Equations (12- (14).

$$v_{id}^{k+1} = \omega \cdot [v_{id}^k + c_1 r_1 (p_{ibest.d}^k - x_{id}^k) + c_2 r_2 (p_{gbest.d}^k - x_{id}^k)] \tag{12}$$

$$x_{id}^{k+1} = x_{id}^k + v_{id}^k \tag{13}$$

$$\begin{aligned} &if\ v_{id}^{k+1} > v_{\max},\quad v_{id}^{k+1} = v_{\max} \\ &if\ v_{id}^{k+1} < v_{\min},\quad v_{id}^{k+1} = v_{\min} \end{aligned} \tag{14}$$

where $i = 1, 2, ..., m$ ; $d = 1, 2, ..., D$ ; $\omega$ is inertia weight; $c_1$ , $c_2$ are accelerated factors; $r_1$ , $r_2$ are random numbers uniformly distribute in [0, 1] ; $v_{\max}$, $v_{\min}$ are particle maximum speed limits.

This will have certain blindness. Random initialization can basically ensure uniform distribution, but cannot guarantee the quality of each particle, and it may cause some particles away from the optimal solution, for which it will affect the algorithm convergence speed. In addition, the basic PSO algorithm, the inertia weight factor $\omega$ plays a role in maintaining the particle inertia, and it has the trend to expend optimization space. $\omega$ 's value largely determines the optimal performance of the algorithm, the big value is favour to global optimization, however, it is difficult to get an accurate solution; a small value is in favour of local optimization, and it's easy to get a more accurate solution, but it will lead this algorithm into a local solution area. Therefore, this paper presents particle swarm optimization algorithm based on chaotic disturbance.

Chaos is a ubiquitous phenomenon of nonlinear systems in movement. Generally, the random motion equations calculated by certainty algorithm are called Chaos. Chaotic system is the most typical representative derived from logistic equation whose iterative formula is as follows. Please see Equation (15).

$$z^{t+1} = \mu z^t (1 - z^t), t = 1, 2, ... \tag{15}$$

$t$ is iterative calculate times; $\mu$ is system's control parameter.

Optimization strategies based on chaotic disturbance are as followed:

1) First, initialization. Variable $z^t$ in equation (15) is respectively assigned $n$ times initial value in (0, 1) with a slight difference except three fixed points of 0.25, and 0.75. You can get $n$ times completely different movement trajectory's chaotic variables $z^t(t)$, $t = 1$. See Equation (16).

$$f^* = f(x^*) \tag{16}$$

where $x^*$ is the best solution.

2) Using chaotic disturbance to implement the iterative optimization. See Equation (17).

$$x_{ij} = a_j + (b_j - a_j) z_{ij} \tag{17}$$

where $i=1, 2, \ldots N$, $j=1, 2, \ldots, n$, $a_j$, $b_j$ are constants and also are the top and bottom limits of optimization variables, they are ranging from chaotic variables to corresponding optimization variables range. The performance index equation is:

$$f(t)=f(x(t)),x(t) \\ =(x_1(t),x_2(t),\ldots,x_n(k)) \tag{18}$$

3) If $f(t)<f^*$, then $x_i^*=x_i(t)$, $f^*=f(t)$ and when it reaches to the maximum number of iterations, the optimization will be end, otherwise $t=t+1$ will go to step (2) to continue.

### 3.2. PSO Algorithm Based on Chaotic Disturbance

This paper firstly introduced the chaotic disturbance algorithm into the basic optimization PSO algorithm, initialized particle swarm and formed the initial solution group, and operated particle swarm's t "inert" particles to help them come out of the local optimal solution area , so that the algorithm can quickly find the global optimal solution, thus effectively improve the performance of the algorithm.

1) Particle swarms' initialization based on chaos disturbance.

First, we will produce a *n* dimensional in random, and its vector component values are in or between $0\sim1$'s chaos variables. See Equation (19).

$$Z_1=(z_{11},z_{12},\ldots,z_{1n}) \tag{19}$$

where $Z_1$ is initialized value calculated by Logistic completely chaotic iteration equation $z^{t+1}=4z^t(1-z^t)$. Through calculation we can obtain $N$ of $Z_1$, $Z_2$, ..., $Z_N$. Then we will use chaotic disturbance variables to iterative search, and through equation 20 transforms $Z_i$ chaotic variables' each component to allow the solution space. See Equation (20).

$$x_{ij}=a_j+(b_j-a_j)z_{ij} \tag{20}$$

In this equation $a_j$, $b_j$ are variables' top and bottom constraint limits.

Then we will calculate the network input variables' corresponding fitness value, and preferentially select former *m* as the initial position of particle swarm according to the fitness value, while randomly generate *m* initial velocity within the limits of the variable.

2) Particle merit based on chaos disturbance.

Firstly, map each component $x_{pj}(j=1,2,\ldots,n)$ of each control variables $X_p=(x_{p1},x_{p2},\ldots,x_{pn})$ in swarm to the chaotic space by equation $z_{pj}=(x_{pj}-a_j)/(b_j-a_j)$, and then generate chaos variables sequence $z_{pj}^s$ according to iterative equation $z^{t+1}=4z^t(1-z^t)$, then we will search with chaotic disturbance variables.

Back to the original solution space with inverse mapping equation $x_{pj}^s=a_j+(b_j-a_j)z_{pj}^s$. See Equation (21).

$$X_p^s=(x_{p1}^s,x_{p2}^s,\ldots,x_{pn}^s) \tag{21}$$

Calculate fitness value of each feasible solution $X_p^s$ that chaos variables undergo and preferentially select former *M* solutions $X_p^*$.

Finally, we will replace *M* group's control variables by $X_p^*$ which is currently existing in group. If there are adaptive values better than the global optimal solution in $X_p^*$, then replace its global optimum by *gbest* and update the global optimum.

3) Inertia weight adaptive optimization.

In the particle swarm algorithm, inertia weight $\omega$ plays an important role in algorithm's performance, namely global optimization and local optimization of balancing algorithm. When $\omega$ value is bigger than global optimization, it is difficult to get an accurate solution; when $\omega$ value is smaller than local optimization, it is easy to fall into local extreme point. To improve the performance of the algorithm, we use an adaptive $\omega$ strategy based on particle individual value, which means that the $\omega$ of each particle adapts and changes according to its current fitness value. See Equation (22).

$$\omega=\begin{cases}\omega_{\min}+\dfrac{(\omega_{\max}-\omega_{\min})(f_i-f_{\min})}{f_{av}-f_{\min}}, & f_i\le f_{av}\\ \omega_{\max}, & f_i>f_{av}\end{cases} \tag{22}$$

where $\omega_{\min}$, $\omega_{\max}$ respectively represents the minimum and maximum value of inertia weight coefficient $\omega$; $f_{av}$ is average value of current entire population; $f_{\min}$ is minimum value of its population; $f_i$ is particle *i*'s current fitness value.

### 3.3. Neural Network Based on Improved Particle Swarm

The proposed perturbation particle swarm optimization based on chaotic neural networks uses a single hidden layer network structure as a basis to build improved neural networks, and the output of the network is as follow. See Equation (23).

$$Y=g(V[g(W^TX)+B_1]+B_2) \tag{23}$$

where $X=(x_1, x_2, \ldots, x_m)^T$ is model input vector, $Y$ is output vector, $W=(w_{ij})_{m\times n}$ is connection between the input layer and the hidden layer in weight matrix, $V=(v_1, v_2, \ldots, v_n)^T$ is the connection weights vector between the hidden layer and output layer, $B_1=(b_1, b_2, \ldots, b_n)^T$ is threshold vector from the input layer to the hidden layer, while $B_2=(b)$ is threshold vector from the hidden layer to the output layer.

Transfer function $g(h)$ in hidden layer uses Tansig function, the mathematical expression:

$$g(h)=(1-e^{-2h})/(1+e^{-2h}) \tag{24}$$

Neural network model uses the mean squared error

(MSE) as the evaluation of training effectiveness indicators:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y-\bar{Y})^2 \tag{25}$$

where $Y$ is the expected output.

In addition, we use real-coded connecting weights and thresholds neural in network expressing particle parameters. Specific encoding can be summarized as follows: Suppose neural network's input layer nodes are $m$, the hidden layer nodes are $n$, the output layer nodes are $s$, and the particle swarm individual string length is:

$$L = n\times m + s\times n + n\times s \tag{26}$$

So, the connected input layer and hidden layer weights matrix are:

$$IW = \begin{bmatrix} IW_{11} & IW_{12} & \dots & IW_{1m} \\ IW_{21} & IW_{22} & \dots & IW_{2m} \\ \dots & \dots & \dots & \dots \\ IW_{n1} & IW_{n2} & \dots & IW_{nm} \end{bmatrix} \tag{27}$$

Threshold vector from the input layer to the hidden layer is $B_1 = [b_{11}, b_{12}, ..., b_{1n}]^T$; so weights matrix connected hidden layer and output layer are:

$$LW = \begin{bmatrix} LW_{11} & LW_{12} & \dots & LW_{1m} \\ LW_{21} & LW_{22} & \dots & LW_{2m} \\ \dots & \dots & \dots & \dots \\ LW_{s1} & LW_{s2} & \dots & LW_{sm} \end{bmatrix} \tag{28}$$

Threshold vector from the hidden layer to the output layer is $B_2 = [b_{21}, b_{22}, ..., b_{2s}]^T$. The encoded form of the particles is:

$$X = [IW_{11}...IW_{nm}LW_{11}...LW_{sn}b_{11}...b_{1n}b_{21}...b_{2s}] \tag{29}$$

Therefore, the network consumption analysis neural network models based on chaos disturbance particle swarm specific processes are:

1) Establish a network consumption forecasting model based on improved neural network to determine the network topology and to enter the training sample after normalization;
2) The position and velocity of initialization's population, we determine the number of particles $N$, inertia weight $\omega$ and shrinkage factor $K$, acceleration constant $c_1$, $c_2$ and $gb$, $pb_n$'s initial value;
3) Do the population mapping, weight and threshold value as the initial parameter values of neural network, then we can obtain the network output from training

   sample's input and also cam calculate actual output square error. Taking MSE as a measure of each particle's fitness value and then we can find the individual extreme value of each particle;
4) Evaluate each particle's individual extreme value, pick the optimal individual extreme value as this iteration's global extremes value $gb$, and record the serial number of the optimal particle value, and the extremes value are as the most optimal parameters of neural predicted network of the next iteration;
5) Estimate whether $gb$ and the number of the current iteration satisfied termination condition, if it meets the condition, then we will exit and go to optimization step (7);
6) Update each particle's velocity and position and calculate the fitness value of each particle, if the value is better than the current value, then the $pb_n$ will as the current individual extremes. Compare all the particles' individual extreme value to current global optimum values; if it is better, then we will set $pb_n$ as this particle's position. And we will update the inertia weight, then go to step (3);
7) According to obtained optimal particle's position, we will take it as the initial value of neural network, then we can input test samples to predict.

## 4. Algorithm Performance Simulations

In order to verify the effectiveness of improved algorithm, we will have simulation experiments. Firstly, we use sphere function, Rosenbrock function and Rastrigrin function to test improved particle swarm algorithm's performance. In this experiment, the population size of PSO are $n = 40,\ c_1 = c_2 = 2,\ w_{\max} = 0.9,\ w_{\min} = 0.3,$ iteration termination condition's accuracy is $10^{-15}$ or reaches to maximum iteration number of 500. Standard PSO and improved particle swarm optimization all these three functions' results are:

| Iterations | PSO Optimizing value | Error | IM-PSO Optimizing value | Error |
|---|---|---|---|---|
| 50 | 12.5 | 32.1% | 13.1 | 15.3% |
| 100 | 26.7 | 28.3% | 28.3 | 12.4% |
| 150 | 38.1 | 29.1% | 42.6 | 11.5% |
| 200 | 45.3 | 30.2% | 49.0 | 14.2% |
| 250 | 59.2 | 22.6% | 64.2 | 13.7% |
| 300 | 71.3 | 26.8% | 77.9 | 13.8% |
| 350 | 88.7 | 27.0% | 98.1 | 11.0% |

**Tab. 1.** The results for the test.

| Iterations | PSO Optimizing value | Error | IM-PSO Optimizing value | Error |
|---|---|---|---|---|
| 50 | 64.3 | 21.1% | 78.3 | 2.3% |
| 100 | 73.2 | 19.4% | 88.9 | 4.1% |
| 150 | 92.5 | 16.4% | 96.1 | 5.2% |
| 200 | 107.1 | 20.9% | 109.2 | 7.3% |
| 250 | 115.9 | 15.3% | 121.3 | 2.4% |
| 300 | 128.3 | 13.2% | 136.2 | 1.3% |
| 350 | 139.3 | 16.7% | 144.1 | 6.2% |

**Tab. 2.** The results for the test.

| Iterations | PSO Optimizing value | Error | IM-PSO Optimizing value | Error |
|---|---|---|---|---|
| 50 | 2.4 | 10.2% | 3.2 | 1.3% |
| 100 | 6.3 | 5.3% | 10.1 | 0.6% |

| | | | | |
|---|---|---|---|---|
| 150 | 10.2 | 7.3% | 15.2 | 1.3% |
| 200 | 12.3 | 6.2% | 19.4 | 2.2% |
| 250 | 15.5 | 7.7% | 22.1 | 1.8% |
| 300 | 19.4 | 8.4% | 25.3 | 0.9% |
| 350 | 22.1 | 4.9% | 29.6 | 1.3% |

**Tab. 3.** The results for the test.

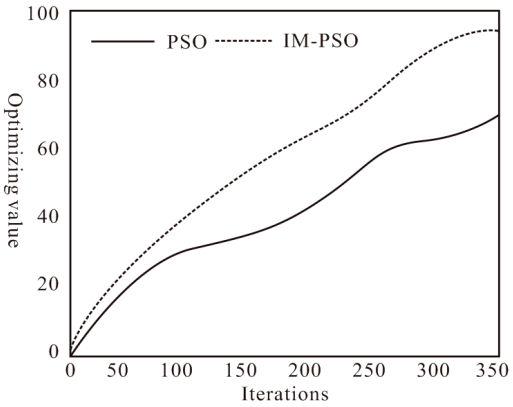


**Fig. 2.** Sphere function optimization process is relatively.

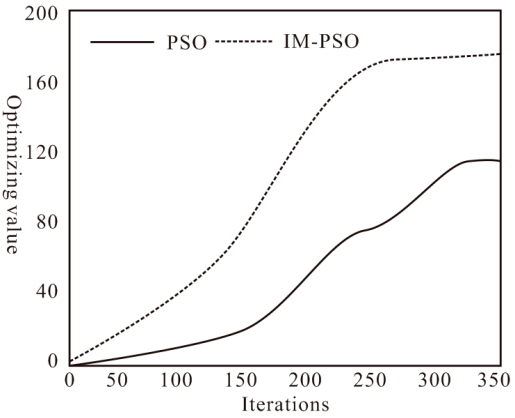


**Fig. 3.** Rosenbrock function optimization process is relatively.

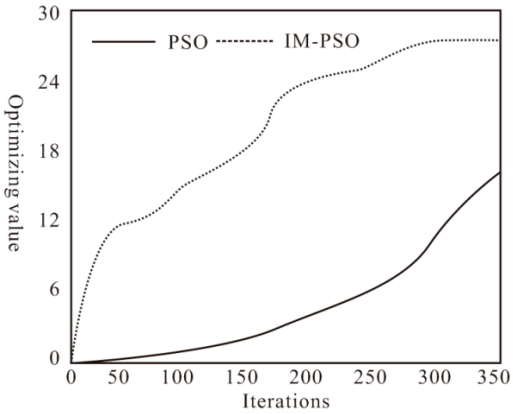


**Fig. 4.** Rastrigrin function optimization process is relatively.

It can be found that compared with the original PSO algorithm, the improved PSO algorithm proposed in this paper is faster and easier to achieve the optimal value.

Then we apply the improved PSO algorithm in the network consumption forecast model, respectively compare to neural network model and standard particle swarm algorithm, we can predict its error, and the results can be seen in Figure 4.

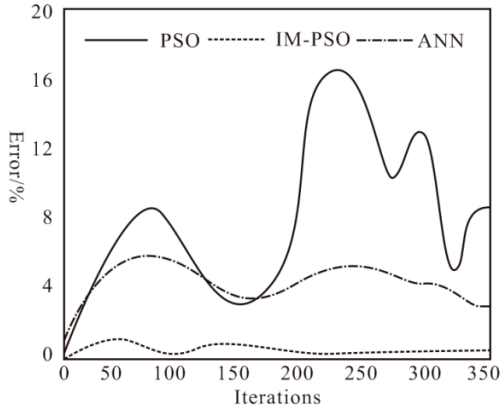


**Fig. 5.** Parameters noiseless case compared with the estimated.

As can be seen from the simulation results, the network consumption analysis network model based on chaotic disturbance particle swarm, compared to the traditional neural network model and standard neural network model's particle swarm optimization, its forecast accuracy has been greatly improved.

## 5. Conclusion

With the optimization of the network environment and enhancement of consumption pattern, the future will have a rapid development, so it is extremely urgent in analysis and forecast of network consumption. This paper presents a network consumption model based on chaotic disturbance particle optimized swarm. It can be seen from the simulation results that the proposed improved strategy greatly enhances the network consumption forecast's accuracy, and it can boost network consumption, and also it will provide a reference for standardized network consumption order.

**Acknowledgement**
This work was supported by the project of college's humanities and social sciences of Shandong Province (J13WF75).

## References

1. L. Yong, Comparison and application examples on several common resident consumption prediction models, Consumer Economy, vol. 3, pp. 29-31 (2014)
2. F. Ying, Consumer confidence index for consumer predict empirical test, Statistics and Decision, vol. 15, pp. 15-19 (2013).
3. M. Hailin, Study on energy consumption prediction based on improved grey model, Journal of Dalian University of Technology, vol. 51(4), pp. 493-497 (2014).
4. Z. Airong, Study on gas concentration prediction method based on wavelet analysis and BP neural network, Mining Machinery, vol. 35(6), pp. 258-260 (2014).
5. X. Yijun, Application status of the fuzzy neural network in prediction of welding performance, Electric Welding Machine, vol. 44(6), pp. 46-50 (2014).
6. C. Guimei, Predictive modelling of blast furnace temperature by using distributed neural network model, Journal of Iron and Steel Research, vol. 26(6), pp. 27-30 (2014).
7. L. Da, Ethernet delay predictive control industry based on wavelet neural network, Dalian University of Technology, vol. 54(2), pp. 246-250 (2014).
8. F. Zhaofeng, RBF neural network multi-step predictive control for nonlinear systems, Control and Decision, vol. 29(7), pp. 1274-1278 (2014).
9. W. Yang, Transmission model of micro-blog network information based on gray prediction, Journal of Anhui Normal University (Natural Science Edition), vol. 37(2), pp. 129-133 (2014).
10. W. Xuancheng, Logistics predict hybrid model based on seasonal decomposition and neural network, Statistics and Decision,vol . (11), pp. 80-82 (2014).
11. S. Yongzeng, Prediction of short-term traffic flow based on chaos particle optimization swarm wavelet neural network, Computer Applications and Software, vol. 31(6), pp. 84-86 (2014).
12. H. Min, Research on multivariate chaotic time series prediction by using mRSM model, Acta Automatica Sinica, vol. 40(5), pp. 822-829 (2014).
13. Z. Ke, Chaotic neural network model for software reliability, Computer Science, vol. 41(4), pp. 172-177 (2014).