www.ijemr.net ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

Volume-7, Issue-3, May-June 2017

International Journal of Engineering and Management Research

Page Number: 21-27

BPNN-ADE Algorithm for the Time Series Data Forecasting

Jaya Singh[1], Pratyush Tripathi[2]

[1]Department of Electronics & Communication Engineering, M.Tech Scholar, K.I.T, Kanpur, INDIA

[2]Assistant Professor, Department of Electronics & Communication Engineering, K.I.T, Kanpur, INDIA

ABSTRACT

Time series is a collection of data recorded over a period of time (weekly, monthly, quarterly), an analysis of history, which can be used by management to make current decisions and plans based on long-term forecasting. It usually assumes past pattern to continue into the future. Time series forecasting is an important area in forecasting. Artificial Neural Networks (ANNs) have the ability of learning and to adapt to new situations by recognizing patterns in previous data. Efficient time series forecasting is of utmost importance in order to make better decision under uncertainty. Over the past few years a large literature has evolved to forecast time series using different artificial neural network (ANN) models because of its several distinguishing characteristics.

The back propagation neural network (BPNN) can easily fall into the local minimum point in time series forecasting. A hybrid approach that combines the adaptive differential evolution (ADE) algorithm with BPNN, called ADE–BPNN, is designed to improve the forecasting accuracy of BPNN. ADE is first applied to search for the global initial connection weights and thresholds of BPNN. Then, BPNN is employed to thoroughly search for the optimal weights and thresholds.

Two comparative real-life series data sets are used to verify the feasibility and effectiveness of the hybrid method. In comparative example 2, the Lynx series indicates the number of Lynx trapped per year in the river district in northern Canada. Lynx is a kind of animal. The proposed ADE–BPNN can effectively improve forecasting accuracy relative to basic BPNN; differential evolution back propagation neural network (DE-BPNN), and genetic algorithm back propagation neural network (GA-BPNN).

Keyword-- Time series forecasting, Back propagation neural network, Differential evolution algorithm, DE and GA

I. INTRODUCTION

Time series forecasting (TSF) is the process of predicting the future values based solely on the past values. Traditionally TSF has been performed using various statistical-based methods. The major drawback of most of the statistical models is that, they consider the time series are generated from a linear process. Therefore several computational intelligence methods have been used to forecast the time series.

Based on the number of time series involved in forecasting process, TSF may be univariate (forecasts based solely on one time series) or multivariate (forecasts depend directly or indirectly on more than one time series). Irrespective of the type of TSF, it became an important tool in decision making process, since it has been successfully applied in the areas such as economic, finance, management, engineering etc. Traditionally these TSF has been performed using various statistical-based methods [15]. The major drawback of most of the statistical models is that, they consider the time series are generated from a linear process.

Out of various models, artificial neural networks (ANNs) have been widely used because of its several unique features. First, ANNs are data-driven self-adaptive nonlinear methods that do not require a priori specific assumptions about the underlying model. Secondly ANNs have the capability to extract the relationship between the inputs and outputs of a process i.e. they can learn by themselves. Artificial neural networks(ANNs) have been extensively studied and used in time series forecasting (Adebiyi, Adewumi, & Ayo, 2014; Bennett, Stewart,& Beal, 2013; Geem & Roper, 2009; Zhang, Patuwo, & Hu, 2001; Zhang & Qi, 2005). Zhang et al. (1998) presented a review of ANNs.The advantages of ANNs are their flexible nonlinear modeling capability, strong adaptability, as well as their learning and massive parallel computing abilities (Ticknor, 2013). Specifying a particular model form is unnecessary for ANNs; the model is instead adaptively formed based on the features presented by the data. In this work evolutionary neural networks (trained using evolutionary algorithms) are used for TSF, so that better forecast accuracy can be achieved. Two evolutionary algorithms like genetic algorithm and differential evolution

are considered. For comparison results obtained from evolutionary algorithms are compared with results obtained from extended back propagation algorithms.

II. OVERVIEW OF DE, ADE AND GA BPNN

The detailed analysis of BPNN for time series forecasting and GA, DE and ADE-BPNN methods. This study uses adaptive DE (ADE) to select appropriate initial connection weights and thresholds for BPNN to improve its forecasting accuracy.

2.1 BACK PROPAGATION NEURAL NETWORK

Back propagation is a systematic method for training multi-layer artificial neural networks. It is a multi-layer forward network using extend gradient –descent based delta learning rule, commonly known as back propagation (of errors) rule. Back propagation provides a computationally efficient method for changing the weights in a feed forward network, with differentiable activation function units.

BPNN is well known for its back propagation-learning algorithm, which is a mentor-learning algorithm of gradient descent, or its alteration (Zhang et al., 1998). According to the theory, the connection weights and thresholds of a network are randomly initialized first. Then, by using the training sample, the connection weights and thresholds of the network are adjusted to minimize the mean square error (MSE) of the network output value and actual value through gradient descent.

When the MSE achieves the goal setting, the connection weights and thresholds are determined, and the training process of the network is finished. However, one flaw of this learning algorithm is that the final training result depends on the initial connection weights and thresholds to a large extent. Hence, the training result easily falls into the local minimum point rather than into the global optimum; thus, the network cannot forecast precisely.

To overcome this shortcoming, many researchers have proposed different methods to optimize the initial connection weights and thresholds of traditional BPNN.Yam and Chow (2000) proposed a linear algebraic method to select the initial connection weights and thresholds of BPNN. Intelligent evolution algorithms, such as the genetic algorithm (GA) (Irani & Nasimi, 2011) and particle swarm optimization (PSO) (Zhang, Zhang, Lok, & Lyu, 2007), have also been used to select the initial connection weights and thresholds of BPNN. The proposed models are superior to traditional BPNN models in terms of convergence speed or prediction accuracy.

Back-propagation neural network is one of the most common neural network structures, as it is simple and effective. The hidden and output layer nodes adjust the weights value depend on the error in classification. A single hidden layer Back Propagation Neural Network (BPNN) consists of an input layer, a hidden layer, and an output layer. Adjacent layers are connected by weights, which are always distributed between -1 and 1. The most common means to determine the appropriate number of input and hidden nodes is via experiments or by trial and error based on the minimum mean square error of the test data [12].In the current study, a single hidden layer BPNN is used for one step- ahead forecasting. Several past observations are used to forecast the present value. That is, the input is $y_{t-n}, y_{t-n+1}, \ldots\ldots\ldots, y_{t-2}, y_{t-1}$ and y_t is the target output. The input and output values of the hidden layer are represented as Equations (1) and (2), respectively, the input and output values of the output layer are represented as Equations (3) and (4), respectively.

The equations are given as follows:

$$I_j = \sum_{i=t-n}^{t-1} w_{ij} * y_i + \beta_j \quad (1)$$

$$y_j = f_n(I_j) \quad (2)$$

Where, j=1, 2…….h

$$I_0 = \sum_{j=1}^{h} w_{0j} * y_j + \alpha_0 \quad (3)$$

$$y_t = f_0(I_0) \quad (4)$$

Where I denotes the input; y denotes the output; y_t is the forecasted value of point t; n and h denote the number of input layer nodes and hidden layer nodes, respectively; w_{ij} denotes the connection weights of the input and hidden layers; and w_{0j} denotes the connection weights of the hidden and output layers, β_j and α_0 are the threshold values of the hidden and output layers, respectively, which are always distributed between -1 and 1. Here f_n and f_0 are the activation functions of the hidden and output layers, respectively.

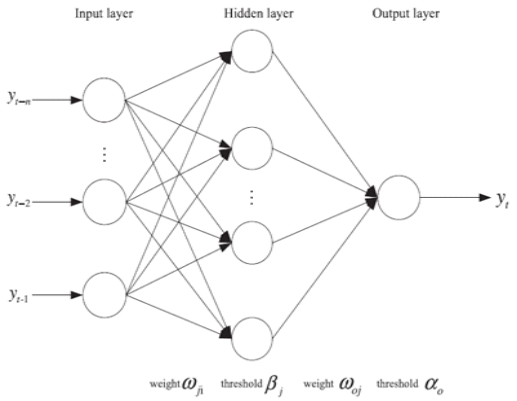

Figure 1: Single hidden layer BPNN structure

2.2 DE, ADE and GA

In this section the detailed analysis of BPNN for time

series forecasting, its impact and definition, parameter to analysis GA, DE and ADE-BPNN methods are included

2.2.1 Standard DE

The standard DE consists of four main operations: initialization, mutation, crossover, and selection.

A) Initialization

Real number coding is used for the DE. In this operation, several parameters, including population size N, chromosome length D, scaling or mutation factor F, crossover rate CR, and the range of gene value $[U_{max}, U_{min}]$, are initialized. The population is randomly initialized as:

$$x_{ij} = U_{min} + rand * (U_{max} - U_{min}) \quad (5)$$

Where i = 1, 2... N, j = 1, 2... D and rand is a random number with a uniform probability distribution.

B) Mutation

For each objective individual x_i^G, i = 1, 2. . . N, the standard DE algorithm generates a corresponding mutated individual, which is expressed:

$$U_i^{G+1} = x_{r_1}^G + F * (x_{r_2}^G - x_{r_3}^G) \quad (6)$$

Where the individual serial numbers r_1, r_2, and r_3 are different and randomly generated. None of the numbers is identical to the objective individual serial number i. Therefore, the population size NP4. The scaling factor F, which controls the mutation degree, is within the range of [0,2], as mentioned [6].

C) Crossover

The crossover operation method, which is shown in Equation (7), generates an experimental individual as follows:

$$U_{ij}^{G+1} = \begin{cases} v_{ij}^{G+1}, if\ r(j) \le CR\ or\ j = rn(i) \\ x_{ij}^G, \qquad otherwise \end{cases} \quad (7)$$

Where r(j) is a randomly generated number in the uniform distribution [0, 1], and j denotes the j_{th} gene of an individual. The crossover rate CR is within the range of [0, 1], which has to be determined by the user. The randomly generated number rn(i) e [1, 2, . . .,D] is the gene index. This index is applied to ensure that at least one dimension of the experimental individual is from the mutated individual. Equation (7) shows that the smaller the CR is, the better the global search effect.

D) Selection

A greedy search strategy is adopted by the DE. Each objective individual x_i^G has to compete with its corresponding experimental individual U_i^{G+1}, which is generated after the mutation and crossover operations. When the fitness value of the experimental individual U_i^{G+1} is better than that of the objective individual x_i^G, U_i^{G+1} will be chosen as the offspring; otherwise, x_i^G directly becomes the offspring. Setting the minimum problem as an example, the selection method is shown in Equation (8), where *f* is the fitness function such as a cost or forecasting error function. The equation is given as follows:

$$x_i^{G+1} = \begin{cases} U_i^{G+1}, if\ f(U_i^{G+1} < f\ x_i^G) \\ x_i^G, \qquad otherwise \end{cases} \quad (8)$$

2.2.2 Adaptive Differential Evolution (ADE-BPNN)

The mutation factor F determines the scaling ratio of the differential vector. If F is too big, then the efficiency of the DE will be low; that is, the global optimal solution acquired by the DE exhibits low accuracy. By contrast, if F is too small, then the diversity of the population will not be ensured as the algorithm will mature early. Consequently, we propose the adaptive mutation factor shown in Equation (9). F changes as the algorithm iterates. It is large during the initial stage, which can guarantee the diversity of the population. During the later stage of the algorithm, the smaller mutation factor can retain the excellent individuals.

$$F = F_{min} + (F_{max} - F_{min}) * e^{1-\frac{GenM}{GenM-G+1}} \quad (9)$$

Where F_{min} denotes the minimum value of the mutation factor, F_{max} denotes the maximum value, GenM is the maximum iteration number, and G is the present iteration number.

2.2.3 Genetic Algorithm (GA)

In GA, the evolution starts from a population of completely random individuals and occur in generations. In each generation, the fitness of the whole population is evaluated; multiple individuals are stochastically selected from the current population (based on their fitness), and modified (mutated or recombined) to form a new population [21,16]. The new population is then used in the next iteration of the algorithm.

A) *Selection* :- Chromosomes are selected from the population to become parents to crossover. The problem is how to select these chromosomes. There are many methods to select the best chromosomes, such as, roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and many others. Every method has some merits as well as some limitations. In this thesis, Roulette wheel selection is used to select the chromosomes. Lastly, elitism is used to copy the best chromosome (or a few best chromosomes) to new population. Elitism helps in increasing the performance of GA, because it prevents losing the best found solution.

B) Crossover: - Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way to do this is to choose randomly some crossover point and interchange the value before and after that point.

C) Mutation: - Mutation takes place after crossover. Mutation changes randomly the new offspring. For binary encoding, we can switch a few randomly chosen bits from 1 to 0 or 0 to 1. Mutation ensures genetic diversity within population. This entire process is continued until the convergence criterion is satisfied.

III. PROPOSED METHODOLOGY

***3.1 Data Segmentation*: -** To achieve good forecast accuracy and higher degree of generalization on neural network models hold out method is considered. For this the data is divided into three sets: train, validation and test sets. The train set is used to train the network, the validation set is used to evaluate the training performance (used to determine the stopping criterion) and the test set is used to evaluate the true accuracy of prediction.

3.2 Model Of BPNN Using Adaptive Differential Algorithm: - The initial connection weights and thresholds of BPNN are selected by combining ADE with BPNN. The ADE is used to preliminarily search for the global optimal connection weights and thresholds of BPNN. The optimal results of this step are then assigned to the initial connection weights and thresholds of BPNN. Therefore, each individual in the ADE corresponds to the initial connection weights and thresholds of BPNN.

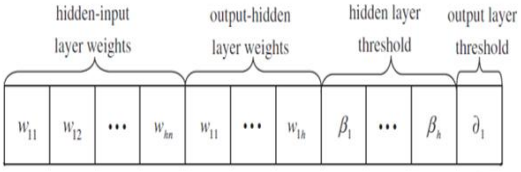

Figure 2: Weights and thresholds of BPNN

The dimension number D is identical to the sum of the numbers of weights and thresholds. That is h * n + o * h + h + o, where n, h and o denote the number of input layer nodes, hidden layer nodes and output layer nodes, respectively. In the one-step-ahead forecasting problem, o = 1. For the BPNN, the search space for connection weights and thresholds is within the range of [-1, 1]. The BPNN uses the Levenberg–Marquardt (LM) method to search for the optimal connection weights and thresholds locally. Therefore, the forecasting model is determined.

A group of weights and thresholds is obtained from each ADE iteration, an output value $y_t (t = 1; 2; \ldots k;$ k is the number of predictions) is generated based on the group of weights and thresholds. The difference between the output value $\hat{y}_t$ and the actual value y_t is used as the fitness function. In general, the mean square error (MSE) or the mean absolute percentage error (MAPE), which is given by Equations (10) and (11), respectively, is chosen as the fitness function.

$$MSE = \frac{\sum_{t=1}^{k}(\hat{y}_t - y_t)^2}{K} \quad (10)$$

$$MAPE = \frac{\sum_{t=1}^{k}(\hat{y}_t - y_t)/y_t}{K} \quad (11)$$

IV. SIMULATION RESULT AND DISCUSSION

Simulation results are done to evaluate the DA, GE and ADE-BPNN models. The proposed ADE–BPNN is programmed by using the software MATLAB.

Two real-life cases are considered to verify the feasibility and effectiveness of the proposed ADE–BPNN model. BPNN has the advantages of flexible nonlinear modeling capability, strong adaptability, as well as their learning and massive parallel computing abilities. So the two cases are suitable for verifying the feasibility and effectiveness of BPNN and ADE–BPNN. One-step-ahead forecasting is considered in both cases.

4.1 Case 1: - Electric load data forecasts

The electric load data consist of 64 monthly data. For a fair comparison with the study of Zhang et al. (2012) [26], the current research used only 53 load data. Several methods can be employed to measure the accuracy of a time series forecasting model. For such prediction, the forecasting accuracy is examined by calculating three frequently used evaluation metrics: the mean square error (MSE), the mean absolute percentage error (MAPE), and the mean absolute error (MAE).

According to Zhang et al. (1998) [23], the most appropriate BPNN parameters are determined after several trials are performed using the same network structure. The maximum training number is 2000; the goal error of training, of which MSE is selected in the aforementioned learning function LM, is set to 0.005; the learning rate is 0.01; the activation function of the hidden and output layers are logsig and purelin functions, respectively. The training function is train LM, which is embedded in the learning function LM. The range of connection weights and thresholds is specified within [-1, 1].

The iteration processes of ADE–BPNN, DE–BPNN and GA–BPNN, all of which including parts. The test sample is forecasted with the best forecasting mode.The actual values and the out-of-sample forecasting loads obtained by different forecasting models, including are the BPNN, DE–BPNN, GA–BPNN and ADE–BPNN.

4.1.1 Resulting graphs of case 1: - Resulting graphs of case 1 using the mean square error (MSE) for DE, ADE and GA algorithms are shown in figure 3, figure 4 and figure 5 respectively. In these figures Y axis shows the actual, forecasted and error data whereas X axis shows years. The iteration is stopped when the iteration number reaches GenM (50).

ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

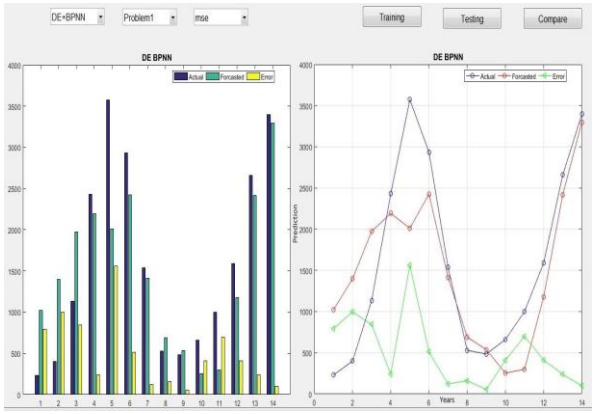

Figure 3 Testing DE+BPNN using MSE

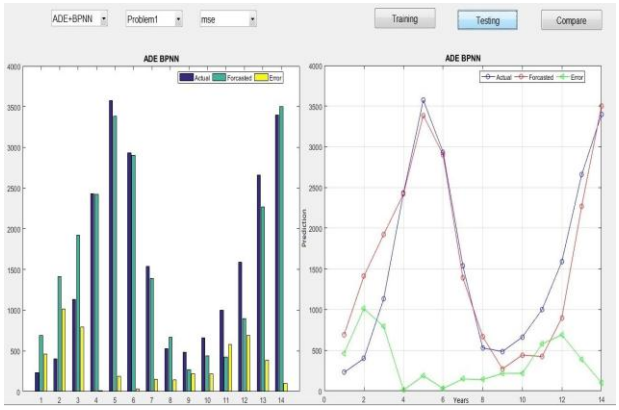

Figure 4 Testing ADE+BPNN using MSE

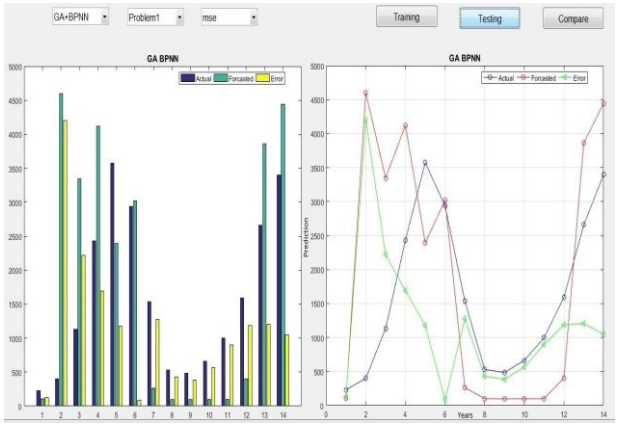

Figure 5 Testing GA+BPNN using MSE

4.2 Case 2: - Canadian Lynx series forecasts

The Lynx series indicates the number of Lynx trapped per year in the river district in northern Canada. Lynx is a kind of animal. The logarithms (base 10) of the data are used in the study.

4.2.1 Resulting graphs of case 2: - Resulting graphs of case 2 using the mean square error (MSE) for DE, ADE and GA algorithms are shown in figure 6, figure 7 and figure 8 respectively. In these figures Y axis shows the actual, forecasted and error data whereas X axis shows years. The iteration is stopped when the iteration number reaches GenM (50).

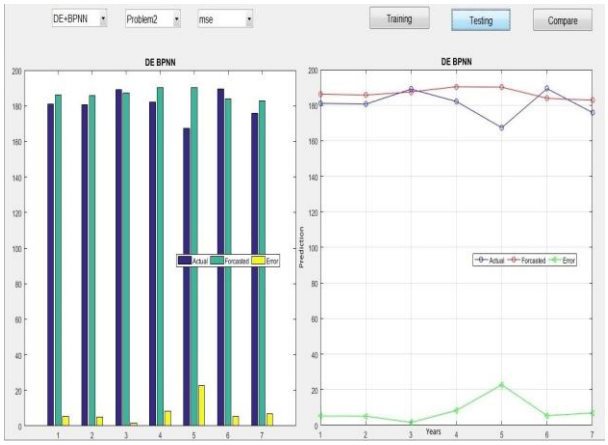

Figure 6 Testing DE+BPNN using MSE

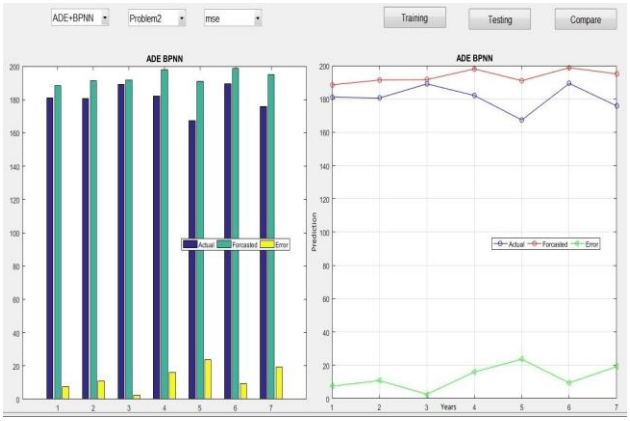

Figure 7 Testing ADE+BPNN using MSE

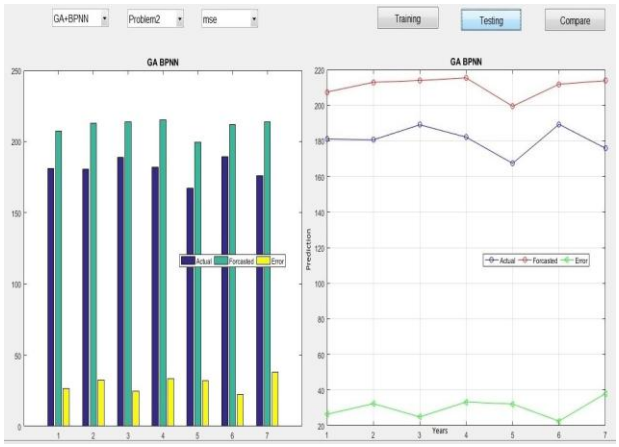

Figure 8 Testing GA+BPNN using MSE

4.2 Graph for comparison between DE, ADE and GA algorithm :-

The figure 9 and 10 shows the comparison between ADE, DE and GA algorithm for case 1 and case 2

respectively. The forecasted results shows that the performance of ADE+BPNN is best among other proposed models.

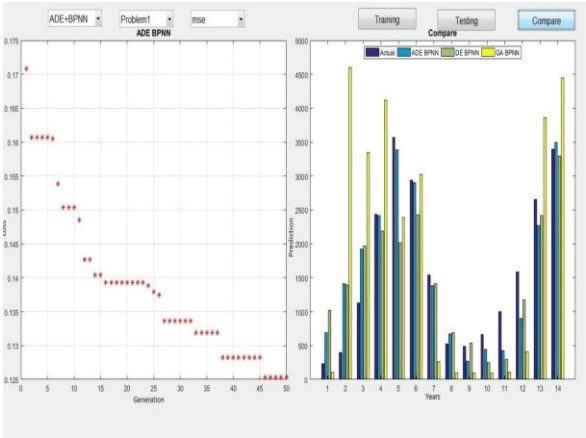

Figure 9 Compare ADE, DE and GA using mse for case 1

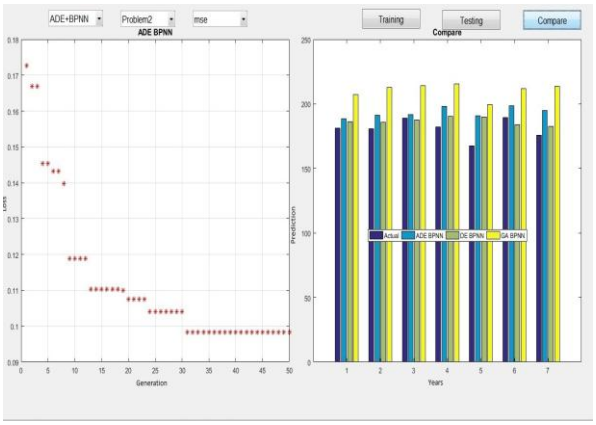

Figure 10 Compare ADE, DE and GA using mse for case 2

V. CONCLUSION

A hybrid forecasting model, called ADE–BPNN, which uses ADE to determine the initial weights and thresholds in the BPNN model, is proposed to improve the accuracy of BPNN in time series forecasting. ADE is adopted to explore the search space and detect potential regions. Two real-life cases are used to compare the forecasting performance of ADE–BPNN with those of other popular models and to verify the feasibility and effectiveness of ADE optimization.

In comparative example 1, the historical monthly electric load data in Northeast China, which include 64 data, are employed. The data set is divided into two subsets, namely, the training set (87 percent data) and the test set (13 percent data). The data exhibit a strong growth trend and an obvious monthly cyclic tendency. The computational results show that the proposed ADE can effectively improve the forecasting accuracy of BPNN compared with the basic BPNN model. One reason for the superior performance of ADE–BPNN is that the intelligence forecasting models of BPNN exhibit good nonlinear fitting capacity. Another reason is that the ADE can determine the appropriate initial parameters of the BPNN model, which can effectively improve forecasting accuracy.

In comparative example 2, the Lynx series indicates the number of Lynx trapped per year in the river district in northern Canada. Lynx is a kind of animal. The logarithms (to the base 10) of the data are used in the study. The Canadian Lynx data, which consist of 114 annual observations, the former 100 data are designated as training data and are applied for ADE optimization and BPNN training. The latter 14 data are assigned as test data and are used to verify the effectiveness of the hybrid model. The proposed ADE–BPNN model is superior to existing basic models (GA-BPNN and DA-BPNN) and some hybrid algorithms in literature in terms of MSE and MAE.

VI. FUTURE SCOPE

In the future, other advanced optimization algorithms can be used to select the best appropriate structure and parameters for the BPNN to handle complex forecasting problems for enterprises in the network economic era. Other intelligent algorithms such as quantum evolution algorithms and genetic simulated annealing algorithms also show good performances to solve complex optimization problems. Evolutionary Algorithms have emerged as strong candidates for the solution of large scale optimization problems. Their multi-point search capabilities are especially suited for parallelization on the modern computing machines. They have been utilized in many domains and success stories abound in the literature. Quantum Evolutionary Algorithms (QEA) is a recent branch of EAs. QEA is a population-based probabilistic Evolutionary Algorithm that integrates concepts from quantum computing for higher representation power and robust search. Traditional GA has strong global search ability in solving problems, but also has defects such as premature and weak local search ability. On the other hand, SA has strong local search ability and no premature problem. Therefore, the combination of GA and SA can overcome the defects of each of the two methods, bring into play their respective advantages, and improve the solving efficiency. This algorithm is named hybrid genetic simulated annealing algorithm (HGSAA).

REFERENCES

[1] Agatonovic Kustrin S. and R. Beresford. Basic concepts of artificial neural net- work (ANN) modeling

and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 22:717–727, 2000.
[2] Andreou Andreas S., Georgopoulos Efstratios F. and Likothanassis Spiridon D. "*Exchange-Rates Forecasting: A Hybrid Algorithm Based on Genetically Optimized Adaptive Neural Networks*" Computational Economics 20: 191–210, 2002.
[3] Chiou J.P. and "*A variable scaling hybrid differential evolution for solving largescale power dispatch problems", IET Generation, Transmission and Distribution*, vol. 3, Iss. 2, pp. 154-163, 2009.
[4] Chris M. Bishop. Neural networks and their applications. *Review of Scientific Instruments*, 65(6):1803–1832, 1994.
[5] Christos Stergiou and Dimitrios Siganos. Neural networks. *SURPRISE 96 Journal*, 4, 1996.
[6] Cui, L. G., Wang, L., & Deng. RFID technology investment evaluation model for the stochastic joint replenishment and delivery problem. *Expert Systems with Applications*, 41(4), 1792–1805, 2014.
[7] Das Swagatam and Konar Amit, "*Automatic image pixel clustering with an improved differential evolution", Applied Soft Computing, Elsevier*, 9, 226-236, 2009.
[8] Das Swagatam and Konar Amit, "*Automatic Clustering Using an Improved Differential Evolution Algorithm" IEEE Transactions on systems*, Man and Cybernetics-Part A: Systems and Humans, 2007.
[9] F. Rosenblatt. "*The perceptron: a probabilistic model for information storage and organization in the brain"*. Psychological Review, 65:386–408, 1958.
[10] Gupta Jatinder N. D. and Sexton Randall S., "*Comparing back propagation with a genetic algorithm for neural network training", Omega (PERGAMON)* 679-684, 1999.
[11] H. Demuth, M. Beale, M. Hagan , Neural Network Toolbox User's Guide, *The Mathworks*, Version 6, 2008.
[12[Hosseini, H. G., Luo, D., & Reynolds, K. J. The comparison of different feed forward neural network architectures for ECG signal diagnosis. *Medical Engineering and Physics*, 28(4), 372–378, 2008.
[13] Khashei, M., & Bijari, M. . A new class of hybrid models for time series forecasting. *Expert Systems with Applications*, 39(4), 4344–4357, 2012
[14] M. Minsky and S. Papert. Perceptrons. Cambridge, Mass.: *MIT Press*, 1969.
[15] Makridakis S.G, Wheelright S.C, & Hyndman R.J , *Forecasting: Methods and Applications*.
[16] McCulloch W. S. and W. H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
[17] Neri, F., & Tirronen, V. Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review*, 33(1–2), 61–106, (2010).
[18] Qin A. K., Huang V. L. and Suganthan P. N., "*Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization" IEEE Transactions on Evolutionary Computation*, 2008.
[19] Ryszard Tadeusiewicz, & Sieci neuronowe (Neural Networks). Akademicka Oficyna Wydawnicza, Warszawa, PL, 1993.
[20] Storn Rainer and Price Kenneth, "*Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces*" J.Global optim., vol. 11, pp. 341-359, 1997.
[21] Varadarajan M. and Swarup K. S., "*Differential evolution approach for optimal power dispatch", Applied Soft Computing, Elsevier,* 8, 1549-1561, 2008.
[22] Wang, L., He, J., & Zeng, Y. R., differential evolution algorithm for joint replenishment problem using direct grouping and its application. *Expert Systems*, 29(5), 429–441, 2012.
[23] Zhang, G. P., Patuwo, B. E., & Hu, M. Y. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1), 35–62, 2012.
[24] Zhang, G. P., Patuwo, B. E., & Hu, M. Y. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1), 35–62, 1998.
[25] Zhang, L., & Subbarayan, G., an evaluation of back-propagation neural networks for the optimal design of structural systems: Part I. Training procedures. *Computer Methods in Applied Mechanics and Engineering*, 191(25), 2873–2886, 2002.
[26] Zhang, W. Y., Hong, W. C., Dong, Y., Tsai, G., Sung, J. T., & Fan, G. F. Application of SVR with chaotic GASA algorithm in cyclic electric load forecasting. *Energy*, 45(1), 850–8, 2012