International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P), Volume-7, Issue-5, May 2017

Time Series Forecasting Using Back Propagation Neural Network with ADE Algorithm

Jaya Singh, Pratyush Tripathi

***Abstract*— 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. 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).**

***Index Terms*—Time series forecasting, Back propagation neural network, Differential evolution algorithm, DE and GA.**

I. INTRODUCTION

Time series is a set of observations measured sequentially through time. Based on the measurement time series may be discrete or continuous. Time series forecasting (TSF) is the process of predicting the future values based solely on the past values. 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 [1]. The major drawback of most of the statistical models is that, they consider the time series are generated from a linear process. However, most of the real world time series generated are often contains temporal and/or spatial variability and suffered from nonlinearity of underlying data generating process. Therefore several computational intelligence methods have been used to forecast the time series. 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. Finally ANNs are universal approximates that can approximate any nonlinear function to any desired level of accuracy, thus applicable to more complicated models [2].

Because of the aforementioned characteristics, ANNs have been widely used for the problem of TSF. 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.

Jaya Singh, Department of Electronics & Communication Engineering, M.Tech Scholar, Kanpur Institute of Technology, Kanpur, India

Pratyush Tripathi, Associate Professor, Department of Electronics & Communication Engineering, Kanpur Institute of Technology, Kanpur, India.

II. LITERATURE REVIEW

Time series forecasting is an important area in forecasting. One of the most widely employed time series analysis models is the autoregressive integrated moving average (ARIMA), which has been used as a forecasting technique in several fields, including traffic (Kumar & Jain, 1999), energy (Ediger & Akar, 2007), economy (Khashei, Rafiei, & Bijari, 2013), tourism (Chu, 2008), and health (Yu, Kim, & Kim, 2013). ARIMA has to assume that a given time series is linear (Box & Jenkins, 1976). However, time series data in real-world settings commonly have nonlinear features under a new economic era (Lee & Tong, 2012; Liu & Wang, 2014a, 2014b; Matias & Reboredo, 2012). Consequently, ARIMA may be unsuitable for most nonlinear real-world problems (Khashei, Bijari, & Ardali, 2009; Zhang, Patuwo, & Hu, 1998). 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.

This data-driven approach is suitable for many empirical data sets, wherein theoretical guidance is unavailable to suggest an appropriate data generation process. The forward neural network is the most widely used ANNs. Meanwhile, the back propagation neural network (BPNN) is one of the most utilized forward neural networks (Wang, Zeng, Zhang, Huang, & Bao, 2006). BPNN, also known as error back

propagation network, is a multilayer mapping network that minimizes an error backward while information is transmitted forward. A single hidden layer BPNN can generally approximate any nonlinear function with arbitrary precision (Aslanargun, Mammadov, Yazici, & Yolacan, 2007). This feature makes BPNN popular for predicting complex nonlinear systems.

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.

As a novel evolutionary computational technique, the differential evolution algorithm (DE) performs better than other popular intelligent algorithms, such as GA and PSO, based on 34 widely used benchmark functions (Vesterstrom & Thomsen, 2004). Compared with popular intelligent algorithms, DE has less complex genetic operations because of its simple mutation operation and one-on-one competition survival strategy. DE can also use individual local information and population global information to search for the optimal solution (Wang, Fu, & Zeng, 2012; Wang, Qu, Chen, & Yan, 2013; Zeng, Wang, Xu, & Fu, 2014). DEs and improved DEs are among the best evolutionary algorithms in a variety of fields because of their easy implementation, quick convergence, and robustness (Onwubolu & Davendra, 2006; Qu, Wang, & Zeng, 2013; Wang, He, & Zeng, 2012). However, only a few researchers have used the DE to select suitable BPNN initial connection weights and thresholds in time series forecasting. Therefore, this study uses adaptive DE (ADE) to select appropriate initial connection weights and thresholds for BPNN to improve its forecasting accuracy. Two real-life time series data sets with nonlinear and cyclic changing tendency features are employed to compare the forecasting performance of the proposed model with those of other forecasting models.

III. BPNN For Time Series Forecasting

A single hidden layer Back Propagation Neural Network (BPNN) consists of an input layer, a hidden layer, and an output layer as shown in Figure 1. Adjacent layers are connected by weights, which are always distributed between -1 and 1. A systematic theory to determine the number of input nodes and hidden layer nodes is unavailable, although some heuristic approaches have been proposed by a number of researchers [3]. None of the choices, however, works efficiently for all problems. 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 [4].

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.

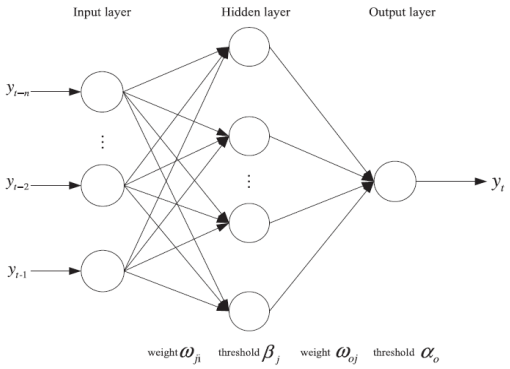

Figure 1: Single hidden layer BPNN structure

Generally, the activation function of each node in the same layer is the same. The most widely used activation function for the output layer is the linear function because the nonlinear activation function may introduce distortion to the predicted output. The logistic and hyperbolic functions are frequently used as the hidden layer activation functions. [13]

IV. Differential Evolution

DE [4] is a population based stochastic search which can be efficiently used as a global optimizer in the continuous search domain. DE has been successfully applied in diverse fields such as large scale power dispatch problem [7], global numerical optimization [8], power loss minimization [6] and pattern reorganization. DE also has been extensively used in different types of clustering like image pixel clustering [9], text document clustering and dynamic clustering for any unknown datasets [10]. Like any other evolutionary algorithms, DE also starts with a population of NP D-dimensional parameter vectors. Two other parameters used in DE are scaling factor F and cross over rate CR.
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, length of chromosome 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 follows:

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

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, thestandard DE algorithm generates a corresponding mutatedindividual, 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 [11].

c) Crossover

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

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

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:

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

V. Genetic Algorithm

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 [12]. 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.

VI. Adaptive Differential Evolution

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.

VII. ADE-BPNN Model

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 as shown in figure 2.

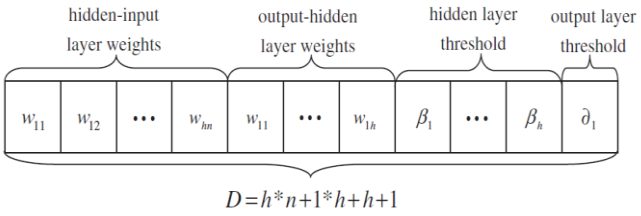

Figure 2: Structure of an individual

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 $\widehat{y_t}$ 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.

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

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

The flowchart of the proposed ADE–BPNN is shown in figure 3, and the procedures are as follows.

Step 1: Initialization. The parameters, namely, population size, maximum iteration number, minimum and maximum mutation factors, crossover factor, and gene range, are set. The initial population is generated by using $x_{ij} = U_{min} + rand * (U_{max} - U_{min})$.

Step 2: The iteration is assessed to determine whether it is completed. If the present smallest fitness value reaches the accuracy requirement l or G is identical with the maximum iteration number, then ADE iteration is stopped. The optimum individual is acquired; otherwise, the procedure proceeds to the next step.

Step 3: The offspring individual x_i^G is generated according to the adaptive mutation, crossover, and selection methods.

Step 4: Step 3 is repeated and the offspring population is generated.

Step 5: The fitness values of the offspring population are evaluated. The smallest fitness value is the present optimal value and the corresponding individual is the present global best individual

Step 6: Set G = G + 1. Return to Step 2.

Step 7: The optimum individual from ADE is assigned as the initial connection weights and thresholds of BPNN. The network is trained with the training sample, and thus, the best-fitting network is created.

Step 8: The network is applied to forecast the test sample.

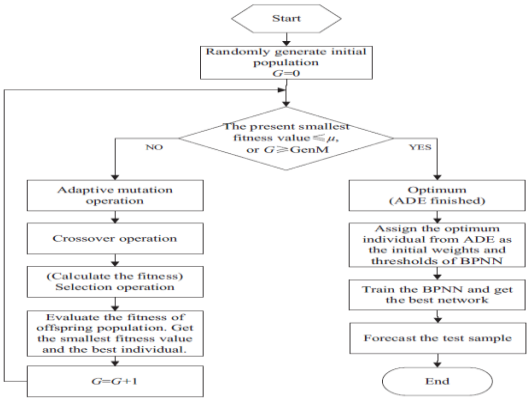

Figure 3: The flowchart of ADE–BPNN algorithm

VIII. Simulation Results

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.

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) [13], 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 root mean square error (RMSE), the mean absolute percentage error (MAPE), and the mean absolute error (MAE).

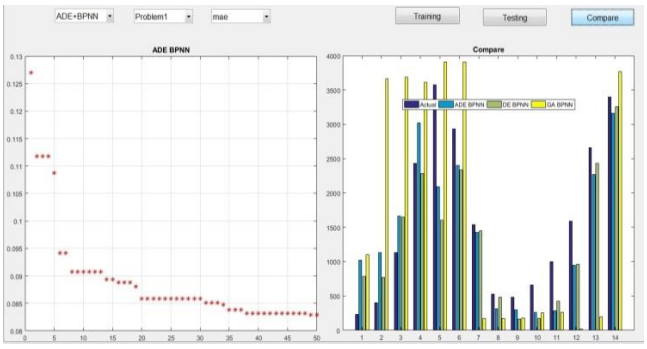

Figure 4: Compare ADE+BPNN mae for case 1

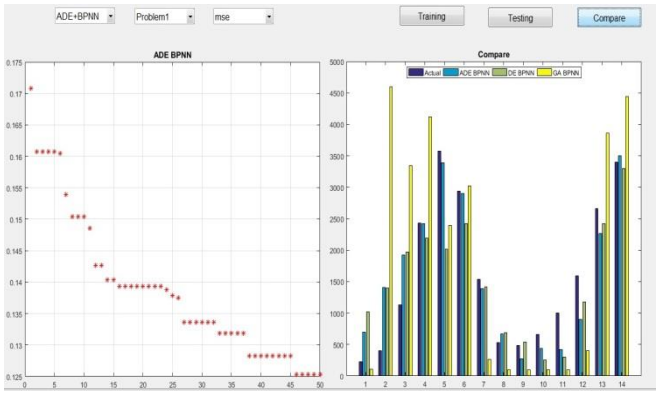

Figure 5: Compare ADE+BPNN mse for case 1

Case 2:- Canadian Lynx series forecasting

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 toexisting basic models (GA-BPNN and DA-BPNN) and some hybrid algorithms in literature in terms of MSE and MAE.

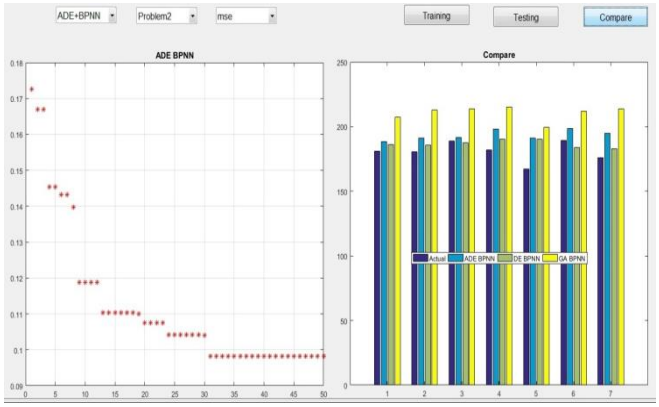

Figure 6: Compare ADE+BPNN mse for case 2

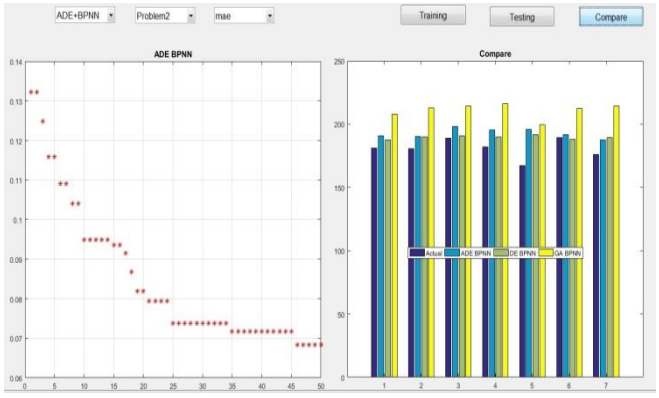

Figure 7: Compare ADE+BPNN mae for case 2

IX. 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.

References

[1] S.G. Makridakis, S.C. Wheelright, R.J. Hyndman, Forecasting: Methods and Applications.
[2] G.P. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with artificial neural networks: The state of art, International Journal of Forecasting 14 (1988) 35-62.
[3] Zhang, L., & Subbarayan, G. (2002). 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.
[4] Hosseini, H. G., Luo, D., & Reynolds, K. J. (2006). The comparison of different feed forward neural network architectures for ECG signal diagnosis. Medical Engineering and Physics, 28(4), 372–378.
[5] 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.
[6] Varadarajan M. and Swarup K. S., "Differential evolution approach for optimal power dispatch" , Applied Soft Computing, Elsevier, 8, 1549-1561, 2008.
[7] 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.
[8] 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.
[9] Das Swagatam and Konar Amit, "Automatic image pixel clustering with an improved differential evolution", Applied Soft Computing, Elsevier, 9, 226-236, 2009.
[10] Das Swagatam, Abraham Ajith and Konar Amit, "Automatic Clustering Using an Improved Differential Evolution Algorithm" IEEE Transactions on systems, Man and Cybernetics-Part A: Systems and Humans, 2007.
[11] Cui, L. G., Wang, L., & Deng, J. (2014). RFID technology investment evaluation model for the stochastic joint replenishment and delivery problem. Expert Systems with Applications, 41(4), 1792–1805.
[12] 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.
[13] Zhang, W. Y., Hong, W. C., Dong, Y., Tsai, G., Sung, J. T., & Fan, G. F. (2012). Application of SVR with chaotic GASA algorithm in cyclic electric load forecasting. Energy, 45(1), 850–858.

Jaya Singh, M.Tech Scholar, Department of Electronics &Communication Engineering, Kanpur Institute of Technology, Kanpur, India.

Pratyush Tripathi, Assistant Professor, Department of Electronics & Communication Engineering, Kanpur Institute of Technology, Kanpur, India.