VOL. 4, NO. 9, NOVEMBER 2009 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

SHORT TERM LOAD FORECASTING USING ARTIFICIAL NEURAL NETWORK: A COMPARISON WITH GENETIC ALGORITHM IMPLEMENTATION

Pradeepta Kumar Sarangi[1], Nanhay Singh[2], R. K. Chauhan[3] and Raghuraj Singh[2]
[1]School of Computer Science, Apeejay Institute of Technology, Greater Noida, India
[2]Department of Computer Science and Engineering, HBTI, Kanpur, India
[3]Department of Computer Science and Applications, Kurukshetra University, India
E-Mail: pradeepta_sarangi@yahoo.com

ABSTRACT

Accurate load forecasting holds a great saving potential for electric utility corporations since it determines its main source of income, particularly in the case of distributors. Precise load forecasting helps the electric utility to make unit commitment decisions, reduce spinning reserve capacity and schedule device maintenance plan properly. It is therefore necessary that the electricity generating organizations should have prior knowledge of future demand with great accuracy. Some data mining algorithms play the greater role to predict the load forecasting. This research work examines and analyzes the use of artificial neural networks (ANN) and genetic algorithm (GA) as forecasting tools for predicting the load demand for three days ahead and comparing the results. Specifically, the ability of neural network (NN) models and genetic algorithm based neural networks (GA-NN) models to predict future electricity load demand is tested by implementing two different techniques such as back propagation algorithm and genetic algorithm based back propagation algorithm (GA-BPN).

Keywords: back propagation network (BPN), genetic algorithm (GA), short term load forecasting (STLF).

1. INTRODUCTION AND BACKGROUND

Forecasting is a phenomenon of knowing what may happen to a system in the next coming time periods. In electrical power systems, there is a great need for accurately forecasting the load and energy requirements because electricity generations as well as distribution are a great financial liability to the state exchequer. Accurate load forecast provides system dispatchers with timely information to operate the system economically and reliably. It is also necessary because availability of electricity is one of the most important factors for industrial development, especially for a developing country like India.

Care also has to be taken that the energy forecast is neither too conservative nor too optimistic. If the forecast is too conservative, then it is very likely that the generating capacity may fall short of the actual power demand, resulting in restrictions being imposed on the power supply that may be detrimental to the economic development of the country. On the other hand, if the forecast is too optimistic, it may lead to the creation of an excess generating capacity, resulting in more investment without getting any immediate returns.

A developing country like India cannot really afford either of the two above conditions owing to considerable pressure on its limited financial resources. Therefore, there is a very strong need for developing electrical load forecasting models using the latest available techniques. Various forecasting models have been developed so far using various graphical and statistical methods, the auto regressive and moving average models being the most popular.

Artificial Intelligence techniques, such as expert systems and neural networks have shown promising results in many systems. Recent progress in the applications of Artificial Neural Networks (ANN) technology to power systems in the areas of forecasting has made it possible to use this technology to overcome the limitations of the other methods used for electrical load forecasting. This is due to the fact that instead of relying on explicit rules or mathematical functions between past load and temperature variations, neural networks draw a link between input and output data. Thus the neural networks that deviate from relying on statistical models and large historical databases hold a good promise for the purpose of load forecasting.

Over the years, the application of Artificial Neural Network (ANN) in power industries has been growing in acceptance. This is because ANN is capable of capturing process information in a black box manner. Given sufficient input-output data, ANN is able to approximate any continuous function to arbitrary accuracy. This has been proven in various fields such as pattern recognition, forecasting, system identification, prediction, signal processing, fault detection and others. The development of a good ANN model depends on several factors. The first factor is related to the data being used. This is consistent with other black box models where model qualities are strongly influenced by the quality of data used. The second factor is the network architecture or model structure. Different network architecture results in different estimation performance. Commonly, multilayer perception and its variances are widely used in process estimation. The third factor is the model size and complexity.

What is required is an accurate model with best parameter settings. This is because a small network may not able to represent the real situation due to its limited

capability, while a large network may over-fit noise in the training data and fail to provide good generalization ability. Finally, the quality of a process model is also strongly dependent on network training. This stage is essentially an identification of model parameters that fits the given data and is perhaps the most important factor among all.

Several attempts have been proposed by various researchers to alleviate this training problem. These include imposing constraints on the search space, restarting training at many random points, adjusting training parameter and restructuring the ANN architecture [1]. However, some approaches are problem-specific and not well accepted and different researchers tend to prefer different methodologies. Among these, one of the more promising techniques is by introducing adaptation of network training using Genetic Algorithm (GA).

In this work, the GA-BPN model is used for extracting the best weight matrices for different layers of BPN thus forecasting the future power demand more accurately. For this reason, this work introduces evolution of connection weights in ANN using GA as means of improving adaptability of the forecasting.

2. RESEARCH OBJECTIVE & METHODOLOGY

The specific objectives of this research work are:

(i) Proposition of a neural network model for predicting the future power demand.
  a) Training of the model (using back propagation algorithm) so that each input produces a desired output.
  b) Testing of the developed model to get the values of future power demands.
(ii) Improvement of back propagation method by hybridizing GA and BPN for determination of optimum weight set.
(iii) Comparison of results of BPN and GA-BPN.

This research work concludes and analyze with the implementation of following methodologies:

- Construction an ANN architecture having three neurons in input layer, two neurons in hidden layer and one neuron in output layer i.e. a 3-2-1 architecture.
- Training of the network using back propagation algorithm with different learning rates, keeping the momentum factor as constant and under maximum permissible error.
- Finding the best learning rate for test patterns.
- Finding the forecasted values using the best value of learning rate.
- Hybridization of BPN with GA. i.e. extraction of weights for BPN by implementing genetic algorithm.
- Training of the GA-BPN network with different values of population size.
- Finding the best value of population size for GA-BPN forecasting.
- Finding the forecasted values for the best value of population size.

3. SIMULATION DESIGN

Simulation design includes (a) Research Data (b) Construction of Network Architecture (c) Requirement of minimum number of patterns and (d) Selection of Input Variables

3.1 Research data

The data used in this research is the daily load (unrestricted demand) data for the State of Delhi. The data (un-normalized) have been collected from "State Load Despatch Centre (SLDC), Delhi" from their website "http://www.delhisldc.org/report/monthly-power-data/July-2008 (block year 2008-09) in the month of February, 2009. The first fifteen days data have been used in this research. For training, the first nine days data have been used and next six days data have been used for testing purposes.

3.2 Construction of network architecture

Structure of the network affects the accuracy of the forecast. Network configuration mainly depends on the number of hidden layers, number of neurons in each hidden layer and the selection of activation function. No clear cut guide lines exist up to date for deciding the architecture of ANN. Mostly it is problem dependent. However, Gowri T. M. *et al.* [2] suggest that for a three layer ANN, the number of hidden neurons can be selected by one of the following thumb rules:

a. (i-1) hidden neurons, where 'i' is the number of input neurons;
b. (i+1) hidden neurons, where 'i' is the number of input neurons;
c. For every 5 input neurons, 8 hidden neurons can be taken. This is developed seeing the performance of network within 5 inputs, 8 hidden neurons and 1 output;
d. Number of input neurons / number of output neurons;
e. Half the sum of input and output neurons; and
f. P/i neurons, where 'i' is the input neurons and 'P' represents number of training samples.

In this research, we have used 3-2-1 architecture.

3.3 Requirement of minimum number of patterns

Gowri T. M. *et al.* suggest that the minimum numbers of patterns required are half the number of input neurons and the maximum is equal to the product of number of input neurons and number of hidden neurons or five times the number of input neurons, which ever is less. However, no justifications can be given to such thumb rules. In this work, performance of configurations is studied with six numbers of training patterns and three numbers of test patterns.

In this research we have used a total number of 15 input data divided into two sets; training set consisting of first 9 days data and the second one is the test set consisting of last 6 days data.

3.4 Selection of input variables

Papalexopoulos A.D. *et al.,* [3] state that, there is no general rule for selecting number of input variables. It largely depends on engineering judgment, experience and is carried out almost entirely on trial and error basis.

The importance of the factors may vary for different customers. According to Wang X. *et al.,* [4], for most customers historical data (such as weather data, weekend load, and previous load data) is most important for predicting demand in STLF. In practice, it is neither necessary nor useful to include all the historical data as input. Autocorrelation helps to identify the most important historical data. Wang describes the importance of recent daily loads in daily load prediction based on correlation analysis is described as below:

Table-1. Selection of input variables.

Factors	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Importance	1	2	3	5	7	6	4

The first line shows the recent daily load history. X_i represents the load consumed and the subscripts show the time index, i.e., 1 means yesterday (one day ago), 2 means the day before yesterday (two days ago), 7 means the same day of last week (7 days ago). In the second line the importance is described. 1 means the most important and 7 means the least important. These information and results help us to decide what factors should be included in input. According to Table-1, when we perform daily load prediction, we should include X_1, X_2, X_3, X_7 as our inputs. Here in this work, we have used three input variables i.e. recent three previous days load data.

4. BPN IMPLEMENTATION STRATEGY

The implementation of Back propagation algorithm and the equations used to calculate various intermediate values and error term are explained with the help of the Figure-1.

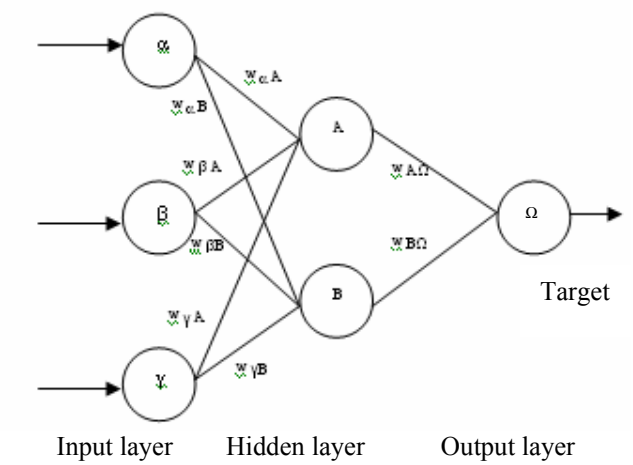

Figure-1. An ANN architecture with weights.

The back propagation algorithm can be explained in a simpler form as below with the help of Figure-1.

a) Calculate errors of output neurons

$$\delta_\Omega = out_\Omega (1- out_\Omega) (target- out_\Omega) \quad 4.1$$

b) Change output layer weights

$$W+_{A\Omega} = W_{A\Omega} + \eta\, \delta_\Omega\, out_A \quad 4.2$$

$$W+_{B\Omega} = W_{B\Omega} + \eta\, \delta_\Omega\, out_B \quad 4.3$$

c) Calculate (bak propagate) hidden layer errors

$$\delta A = out_A (1- out_A) (\delta_\Omega - W_{A\Omega}) \quad 4.4$$

$$\delta B = out_B (1- out_B) (\delta_\Omega - W_{B\Omega}) \quad 4.5$$

d) Change hidden layer weights

$$W+_{\alpha A} = W_{\alpha A} + \eta\, \delta_A\, in_\alpha \quad 4.6$$

$$W+_{\alpha B} = W_{\alpha B} + \eta\, \delta_B\, in_\alpha \quad 4.7$$

$$W+_{\beta A} = W_{\beta A} + \eta\, \delta_A\, in_\beta \quad 4.8$$

$$W+_{\beta B} = W_{\beta B} + \eta\, \delta_B\, in_\beta \quad 4.9$$

$$W+_{\gamma A} = W_{\gamma A} + \eta\, \delta_A\, in_\gamma \quad 4.10$$

$$W+_{\gamma B} = W_{\gamma B} + \eta\, \delta_B\, in_\gamma \quad 4.11$$

4.1 Results and discussions

For training purposes, we have used batch update method i.e. each pattern is fed once and the error is calculated for that pattern. Let this error be ε. Now, the next pattern is fed and error is calculated and this error is added to the error of previous pattern (ε) to form a global error. In this way all patterns are fed and the global error is calculated. Then the global error is compared with the maximum permissible error. If maximum permissible error is greater than the global error then the above procedure is repeated for all patterns till maximum permissible error is less than or equal to the global error.

The program for back propagation has been executed 45 times. The momentum factor has been kept constant (0.9) for all executions. The learning rate "η" has been taken from 0.2 to 1.0 with an increment of 0.1. For each value of "η" the program has been executed 5 times and the root mean square error (RMSE) value is calculated for each execution and then the average RMSE is calculated for that "η" value. The best result (the case having lowest RMSE value) among all these cases (indicated as bold italic letters) has been considered for experiment.

Various parameters used are:
Architecture: 3-2-1
Number of hidden layers: 01
Number of input neurons: 03
Number output neurons: 01
Transfer function: sigmoid
Error calculation method: RMSE
Momentum: 0.9
Error tolerance: 0.001
Stopping criteria: Number of Iterations >30000 OR
network error <= tolerance

The Root Mean Square Error (RMSE) has been calculated as:

RMSE = Square root of [(∑ (Target Value – Obtained Value)^2)/number of input patterns]

Table-2. BPN training result with different learning rate (η) value.

(η)	Run-1 RMSE	Run-2 RMSE	Run-3 RMSE	Run-4 RMSE	Run-5 RMSE	Average RMSE
0.2	0.0831	0.0831	0.0837	0.0949	0.0953	0.088090
0.3	0.1055	0.0642	0.0827	0.0914	0.0801	0.084818
0.4	0.0863	0.0975	0.0953	0.0881	0.0822	0.089934
0.5	0.0875	0.0866	0.0547	0.0835	0.0893	0.085807
0.6	0.0911	0.0836	0.0936	0.0551	0.0852	***0.081759***
0.7	0.0872	0.0822	0.0997	0.1065	0.0973	0.090084
0.8	0.0877	0.0855	0.0906	0.1077	0.0816	0.090677
0.9	0.0953	0.0930	0.0951	0.0931	0.0931	0.093978
1.0	0.0839	0.0850	0.0831	0.0825	0.0986	0.086673

Table-2 is used to find the best value of learning rate 'η'.

4.2 Forecasted values (BPN)

Once the network is trained and validated, the forecasted values have been calculated for both the architectures.

The forecasted values are given in Table-3 below.

Table-3. Forecasted values by BPN.

Network Tolerance: 0.001 **Momentum Factor**: 0.9 **Learning Rate**: 0.6				
Pattern No.	**Desired output**	**Forecasted values**	**Squared error**	**Error (RMSE)**
1	0.5876	0.5587	0.000835	0.081935
2	0.6672	0.7238	0.003204	
3	0.6892	0.8161	0.016104	

5. GA- BPN TRAINING METHODOLOGY

In this research, a real coding system has been adopted for coding the chromosomes in the present work. The network configuration of the BPN for the present work is 3-2-1. Therefore, the numbers of weights (genes) that are to be determined are 3 X 2 + 2 X 1 = 8. With each gene being a real number, and taking the gene length as 5, the string representing the chromosomes of weights will have a length of 8 X 5 = 40. This string represents the weight matrices of the input-hidden-output layers. An initial population of chromosomes is randomly generated. Weights from each chromosome have been extracted. The fitness function has been devised. For cross over, we have used a two-point cross over selected at random and the selection is made on the basis of fitness value (1/MSE). The various steps of GA are explained below.

- **[Start]** Generate random population of '*n*' chromosomes (suitable solutions for the problem)
- **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
- **[New population]** Create a new population by repeating following steps until the new population is complete
  - ❖ **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
  - ❖ **[Crossover]** Cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
  - ❖ **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
  - ❖ **[Accepting]** Place new offspring in the new
- **[Replace]** Use new generated population for a further run of the algorithm
- **[Test]** If the end condition is satisfied, stop, and return the best solution in current population
- **[Loop]** Go to step 2

The methodology for GA-based weight determination is explained below in Figure-2.

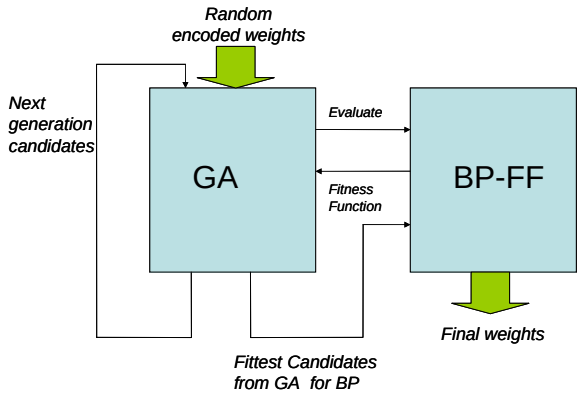

Figure-2. GA-based weight determination.

5.1 Results using GA-BPN method

The GA-BPN has been implemented with different population size. For each value of population, the program has been executed 5 times and the RMSE value has been calculated in the same manner as in case of BPN. The best result (the case having lowest RMSE value) among all these cases (indicated as bold italic letters) has been considered for comparison with other methods. The results have been calculated by using specific software programs developed for this purposes.
Various parameters used in this research are:

Architecture: 3-2-1
Number of hidden layers: 01
Number of input neurons: 03 (previous three days data)
Number output neurons: 01
Selection: Based on fitness value
Fitness function: 1/ (MSE)
Crossover: Two point Crossover
Stopping criteria: when fitness converged

The training results are shown in Table-4.

Table-4. GA-BPN training results with different population size (P).

GA-BPN- Results with different population size (P)						
P	**Run-1 RMSE**	**Run-2 RMSE**	**Run-3 RMSE**	**Run-4 RMSE**	**Run-5 RMSE**	**Average RMSE**
30	0.0669	0.0669	0.0669	0.0669	0.0669	0.066981
40	0.0783	0.0783	0.0783	0.0783	0.0783	0.078319
50	0.0794	0.0794	0.0794	0.0794	0.0794	0.079478
60	0.0703	0.0703	0.0703	0.0703	0.0703	0.070381
70	0.0604	0.0604	0.0604	0.0604	0.0604	0.060463
80	0.0713	0.0713	0.0713	0.0713	0.0713	0.071385
90	0.0205	0.0205	0.0205	0.0205	0.0205	***0.020585***
100	0.0454	0.0454	0.0454	0.0454	0.0454	0.045430

Once the training is over, the forecasted values for next days have been calculated. The values are given in Table-5 below.

Table-5. Forecasted values, GA-BPN.

Pattern No.	Actual / Target value	Forecasted value	Squared error	Error (RMSE)
1	0.5876	0.585405	4.82E-06	0.020585
2	0.6672	0.687654	0.000418	
3	0.6892	0.718321	0.000848	

Comparison of results of both the methods is given in the Table-6.

Table-6. Results Comparison (ANN-BPN and GA-BPN).

Techniques	Actual / target value	Forecasted value	Squared error	Error (RMSE)
ANN-BPN	0.5876	0.5587	0.000835	0.081935
	0.6672	0.7238	0.003204	
	0.6892	0.8161	0.016104	
GA-BPN	0.5876	0.585405	4.82E-06	0.020585
	0.6672	0.687654	0.000418	
	0.6892	0.718321	0.000848	

The graphical representation of Table-6 is given in Figure-3.

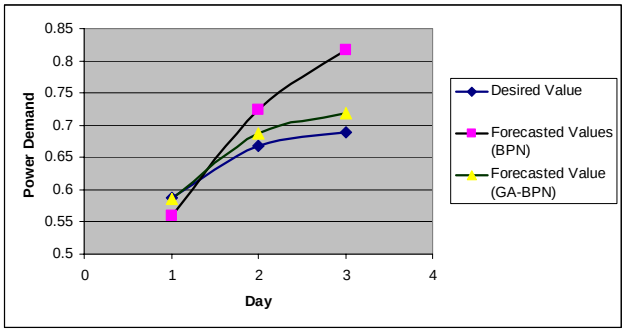

Figure-3. Result comparison, BPN and GA-BPN.

6. CONCLUSIONS

The performance of back propagation algorithm varies with change of learning rate. The performance of genetic algorithm based back propagation algorithm varies with change of population size. Genetic algorithm based back propagation method has been found as a strong forecasting tool for electrical load forecasting in case of short term load forecasting for SLDC, Delhi data is an advantageous position over back propagation method.

REFERENCES

[1] Sexton R.S., Dorsey R.E. and Sikander N.A. 2002. Simultaneous Optimization of Neural Network Function and Architecture Algorithm. Decision Support Systems. 1034: 1-13.

[2] Gowri T. M. and Reddy V.V.C. 2008. Load Forecasting by a Novel Technique using ANN. ARPN Journal of Engineering and Applied Sciences. 3(2): 19-25.

[3] Papalexopoulos A.D., Hao S. and Peng T.M. 1994. An Implementation of a Neural Network Based Load Forecasting Model for the EMS. IEEE Transactions on Power Systems. 9: 1956-1962.

[4] Wang X. and Tsoukalas L. H 2004. Load Analysis and Prediction for Unanticipated Situations Bulk Power System Dynamics and Control- VI. Cortina d'Ampezzo, Italy.

[5] Khan A.U., Bandopadhyaya T.K. and Sharma S. 2008. Genetic Algorithm Based Back Propagation Neural Network Performs better than Back propagation Neural Network in Stock Rates Prediction. International Journal of Computer Science and Network Security. 8(7):

[6] Mishra S. 2008. Short Term Load Forecasting using Neural Network Trained with Genetic Algorithm and Particle Swarm Optimization. 1st International Conference on Emerging Trends in Engineering and Technology, IEEE Computer Society. 978-0-7695-3267-7/08.

[7] Montana D. and Davis L. 1989. Training feed forward Neural Networks using Genetic Algorithms. Proceedings of 11th International Joint Conference on Artificial Intelligence. pp. 762-767.

[8] Peng M., Hubele N.F. and Karady G.G. 1992. Advancement in the Application of Neural Networks for Short-Term Load Forecasting. IEEE Transactions on Power Systems. 7: 250-257.

[9] Asar A. and Syed E., Hassnain R. 2000. Short term load forecasting from an artificial neural network perspective. INMIC 4th IEEE National Multi Topic Conference, Islamabad, Pakistan.

[10] Jain A.K., Mao J. and Mohiuddin K.M. 1996. Artificial Neural Networks: A Tutorial. IEEE Computer Special Issue on Neural Computing. pp. 31-44.