Short Term Load Forecasting using Neural Network trained with Genetic Algorithm & Particle Swarm Optimization

Sanjib Mishra
National Institute of Technology, Rourkela
E-mail: sanjib.mishra77@gmail.com

Sarat Kumar Patra
National Institute of Technology, Rourkela
E-mail: skpatra@nitrkl.ac.in

Abstract

Short term load forecasting is very essential to the operation of electricity companies. It enhances the energy-efficient and reliable operation of power system. Artificial neural networks have long been proven as a very accurate non-linear mapper. ANN based STLF models generally use Back propagation algorithm which does not converge optimally & requires much longer time for training, which makes it difficult for real-time application. In this paper we propose a smaller MLPNN trained by Genetic algorithm & Particle swarm optimization. The GA training gives better accuracy than BP training, where as it takes much longer time. But the PSO training approach converges much faster than both the BP and GA, with a slight compromise in accuracy. This looks to be very suitable for real-time implementation.

1. Introduction

Short term load forecasting is a time series prediction problem. It analyzes the pattern of future electrical load. The information is very crucial to determine hydro-thermal generation mixture, to allot transmission corridor, to decrease over all loss of grid, and to increase operational efficiency.

The load is decomposed into two components. One is weather dependent, and the other is weather independent. Each component is modeled separately and the sum of these two gives the total load forecast. The behavior of these two controls the total load pattern. The behavior of weather independent load is mostly represented by Fourier series or trend profiles in terms of the time functions. The weather sensitive portion of the load is arbitrarily extracted and modeled by a predetermined functional relationship with weather variables.

Feed forward neural net structures like multi layer perceptron, functional link, wavelet, recurrent or feedback structures like Hopfield, Elman, Multi Feedback & hybrid structures using fuzzy neural networks have been proposed in many papers with very high degree of predictive accuracy. But many of those papers have been tested on Macky-Glass series or some other smooth differentiable functions rather than actual load data. Actual load data putforths many challenges to design a predictive neural net structure Prominent of those challenges are, data pre-processing, input parameter selection, type of neural net structure selection, computational complexity and training algorithm Computational complexity is dependent on the structural complexity and training algorithm. Unlike back propagation genetic algorithm and particle swarm optimization training processes are computationally less intensive, and reason there of are explained in subsequent sections. For real time implementations, it is imperative that the structure should have minimal number of neurons and procedure of training method should have minimal number of computations. Considering all the above facts GA and PSO are definitely the ideal candidates for real time implementation. All most all the papers are silent about this.

This paper highlights the suitability of PSO over GA for real time implementation.

Following this introduction the remaining paper is organized as: - Section 2 provides overview of genetic algorithm with particular emphasis on MLP training, while Section 3 analyzes the PSO algorithm. Section 4 highlights the system model for load forecast. The experimental results are presented in Section 5, and the paper concludes in Section 6.

2. GA-ANN training approach

Genetic algorithm (GA) is a directed random search technique [1] that is widely applied in optimization problems [1-3]. This is especially useful for complex optimization problems where the number of parameters is large and the analytical solutions are difficult to obtain. GA can help to find out the optimal solution

978-0-7695-3267-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ICETET.2008.94

globally over a domain. This technique has been applied in different areas such as fuzzy control, path planning [4], modeling & classification [5] etc.

There are two kinds of genetic operators, namely crossover & mutation. For crossover mechanisms, two-point cross over, multipoint crossover, arithmetic crossover, & heuristic crossover have been reported [1], [6-8]. For mutation mechanisms, boundary mutation, uniform mutation, and nonuniform mutation can be found [1], [6-8]. Three steps are used to generate offspring: copying the parents, determining the mutations to be performed, and mutating the copy. In our work reported herein, GA is based on Steady-State Algorithm with Fitness - Proportional Selection procedure. The population is composed of 10 bit binary number chromosomes. The algorithm is described in figure 7 in a self explanatory manner.

3. PSO-ANN training approach

Particle swarm optimization is an idea based on human social behavior. Kennedy and Eberhert [8] in 1994 presented the concept. It models problem as a set of n particles each representing a dimension of solution space. These particles move in solution space in search of optimal solution. The particles follow three principles i.e. evaluating: learning through self experience, Comparing: learning through comparative study and Imitating: learning through adapting the best trend. PSO has many similarities with GA. But it does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm. One of the advantages of PSO is that it takes real numbers as particles unlike GA, which needs to change to binary encoding or special genetic operators have to be used. In GAs, chromosomes share information with each other. So, the whole population moves like one group towards optimal area. In PSO, only global best or local best gives information to others. It is a one way information sharing mechanism. Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases. In PSO, we have got a position parameter, a velocity parameter to be updated for all the dimensions of all the particles. The updation logic lies with the global best parameter of all the particles and the local best parameter of a single particle. Our PSO is based on continuous state variables with real values. The velocity and position updation is done as per the following equations. Details are explained in figure 8.

$$\mathbf{V_i^{k+1}} = \mathbf{w} * \mathbf{V_i^k} + \mathbf{c1}* \text{rand}_1 * (\mathbf{pbest_i} - \mathbf{X_i^k}) + c2* \text{rand}_2 * (\mathbf{gbest} - \mathbf{X_i^k})$$

$$\mathbf{X_i^{k+1}} = \mathbf{Xik} + \mathbf{V_i^{k+1}}$$

4. Network modeling

4.1. Analysis of load pattern

In Indian context load variation with respect to temperature variation is almost dependent upon the seasonal change. Seasonal change is always gradual. Since online MLPNN is trained every 24 hours to predict next 24 hours load pattern or trained every hour to predict next hour load, the temperature variations due to seasonal change need not to be taken as an explicit input parameter. Instead, surge in cooking load in morning hours and commercial, lighting and domestic load in evening hours should be incorporated appropriately as parameters for prediction.

4.2. Input & output for ANN model

In our analysis, the ANN model uses nine inputs including load at hour 'hr-1' , 'hr-2', 'hr-3' of same day, 'hr', 'hr-1', 'hr-2' of previous day, & 'hr', 'hr-1', 'hr-2' of same day of previous week. Only one output node is used representing a 24-hour ahead load forecast at hour 'hr' in the lead time. Fig. 1 shows the network topology used in case of genetic algorithm or particle swarm optimization trained neural network model. In case of back propagation algorithm 17 hidden neurons are used instead of four. The reason behind taking the specific inputs are as follows: It takes into consideration the hour of the day effect to map hourly load variation. Day of the week is taken into account to map weekly pattern of industrial and commercial load pattern on week days and weekends. Seasonal variation is gradual so previous day load pattern as an explicit input takes care of seasonal mapping.

4.3. Data set

For our analysis, we used daily load profile spread over September, 2006 to August, 2007 of Orissa Power Transmission Corporation Limited.

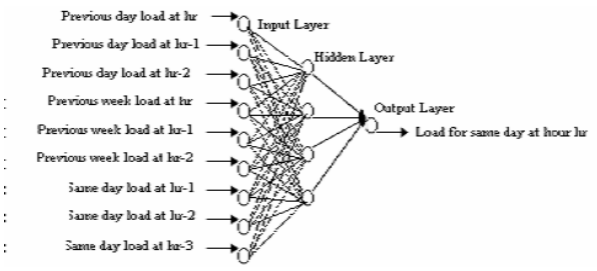

Figure 1. GA or PSO based ANN model

5. Experimental results

The acceptable criteria for a particular model is based upon the (i) minimum average percentage error (MAPE), (ii) number of hours in which it gives negative MAPE, (iii) time taken by the model to get trained. The acceptable criteria (i) & (iii) are self explanatory. The second criteria signifies the under estimation of required load. Under estimation of load may stress the generation units.

The Mean average percentage error (i.e MAPE) in case of MLP-BP was found to be 3.5543 % with logsig activation function, 17 nos. of hidden neurons, learning rate of 0.1, & Guyen-Widrow parameter initialization.

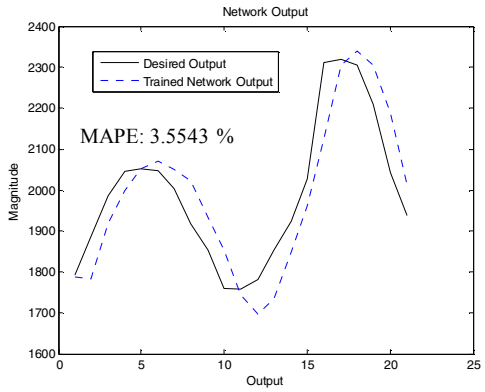

Figure 2. BP - MLPNN result

Instaed of 17 hidden neurons, when the MLP was trained with 4 hidden neurons the error shoot up to unacceptable level. Implementing a MLP with 17 hidden neurons with back propagation training method in real time is very resource consuming. So, the the MLP was tried with GA & PSO. It was found that PSO and GA gives better performance details are as follows.

In case of GA-MLP the best result was found to be, MAPE of 3.1943 %, with 4 nos. hidden neurons, and tansig activation function as shown in figure 3.

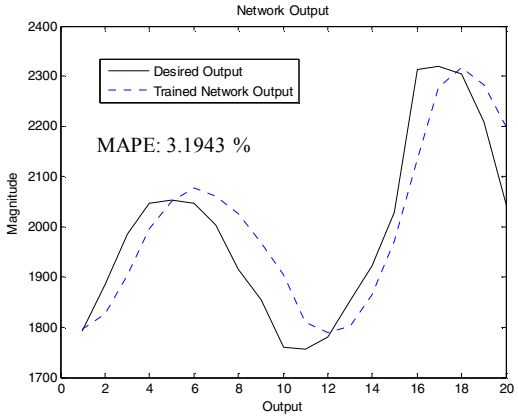

Figure 3. GA - MLPNN result

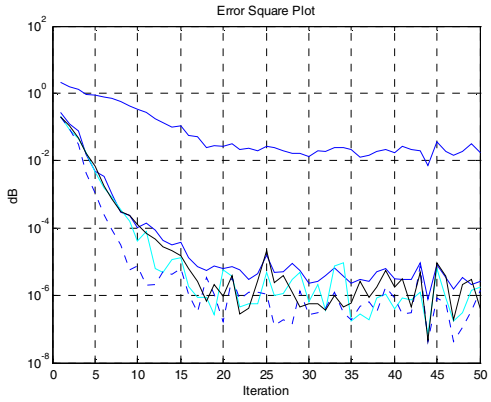

Figure 4. GA - MLPNN MSE plot

In figure 4. the top most curve shows the absolute mean square error and other curves show the contribution of hidden neurons. The error contributions are calculated from the back propagated Jacobian matrix.

Table 1. Performance of MLPNN-GA training

tansig	logsig	tanh	
89.3791	86.109	74.3246	Total Abs. % Error
14	4	2	No. of Min. % Error
11	9	10	Negative % Error
11	5	7	Error less than 3%
3.1934	4.3054	3.7162	MAPE %
131.5843	131.5808	109.1135	Computation Time

In case of PSO-MLP the best result was found to be, MAPE of 4.2118 %, with 4 nos. hidden neurons and logsig activation function.

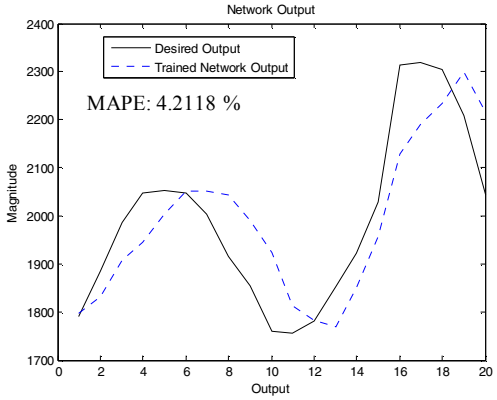

Figure 5. PSO - MLPNN result

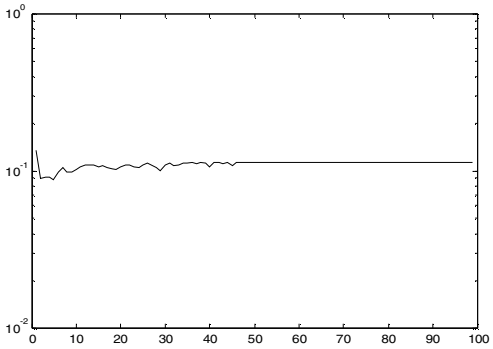

Figure 6. PSO - MLPNN MSE plot

Table 2. Performance of MLPNN-PSO training

tansig	logsig	tanh	
89.3791	84.3767	89.3791	Total Abs. % Error
9	12	9	No. of Min. % Error
8	10	8	Negative % Error
6	5	6	Error less than 3%
4.469	4.2118	4.469	Avg. hourly % Error
6.437982	6.262582	3.790151	Computation Time

6. Conclusion

The paper has demonstrated the use of GA & PSO algorithm for optimizing weights and biases of an ANN short term load forecasting model. The bottlenecks observed in the use of BP algorithm can be overcome by using GA or PSO, which shows improvement in convergence time, & simpler modeling. The GA approach gives better accuracy but takes little more time for training. Where as PSO is much faster in training but accuracy is bit lower. So, while designing a mission critical real time application for STLF, we should choose either GA or PSO training as per requirements of accuracy and speed. Further study will focus mainly on developing STLF models, requiring simpler structure and faster speed of convergence.

7. References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: Univ. Michigan Press, 1975.

[2] D. T. Pham and D. Karaboga, *Intelligent Optimization Techniques, Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks,* New York: Springer-Verlag, 2000.

[3] H. Juidette and H. Youlal, "Fuzzy dynamic path planning using genetic algorithms*"*, *Electron. Lett. vol. 36*, no. 4, pp. 374-376, Feb. 2000.

[4] M. Setnes and H. Roubus, "GA-fuzzy modeling and classification: Complexity and performance", *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 509-522, Oct. 2000.

[5] X. Wang and M. Elbuluk, "Neural network control of induction machines using genetic algorithm training", *IAS Annual Meeting*, vol. 3, 1996, pp. 1733-1740.

[6] L. Davis, "*Handbook of Genetic Algorithms",* New York: Van Nostrand Reinhold, 1991.

[7] M. Srinivas and L. M. Pattanaik, *Genetic Algorithms: A survey*, *IEEE Computer*, vol. 27, pp. 17-27, June 1994.

[8] R. C. Eberhert and J. Kennedy, "A new optimizer using particle swarm theory", *Proceeding of the Sixth International Symposium on Micro Machine and Human Science*, page 39-43,, Nagoya, Japan, 1995, IEEE Service Center, Piscataway, Nj.

[9] K. Jhon, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA: The MIT Press.

initialize **P** // **P**: initial population

iter // **iter**: number of iteration

begin

batch //**batch**: total no. of inputs

begin

calculate the MLP output and thus error

end

calculate MSE and thus fitness function ***f*(P(iter))**

P(iter) is sorted in decreasing fitness order of ***f*(P(iter))**

iter = **iter** + 1

select some pairs of parents $\mathbf{p_1}$ and $\mathbf{p_2}$ from **P(iter-1)** with higher fitness value in the fitness order

perform genetic operations (crossover and mutation) to produce $\mathbf{p_1'}$ and $\mathbf{p_2'}$

replace $\mathbf{p_1}$ and $\mathbf{p_2}$ with $\mathbf{p_1'}$ and $\mathbf{p_2'}$ **and** thus reproducing a new generation **P(iter)**

end

best individual from population **P** is chosen as the output

Figure 7. Procedure for Genetic Algorithm

initialize **S** with random **V** and **X** // **S**: initial swarm, **V**: velocity, **X**: position

initialize w, c1, c2 // w: weighting function, c1 = c2: weighting factors

$k \longrightarrow$ // k: number of iteration

begin

$\mathbf{X}_i^k$: position of a particle

$pbest_i$: personal best position of i^{th} particle

$\mathbf{V}_i^k$: velocity of i^{th} particle

gbest: global best of all the particles of **S**

batch $\longrightarrow$ //**batch**: total no. of inputs

begin

calculate the MLP output and thus error

end

calculate MSE for each particle in swarm

For each particle i in swarm

If ($\mathbf{X}_i^k$ < $\mathbf{pbest}_i$)

Then $\mathbf{pbest}_i = \mathbf{X}_i^k$

If ($\mathbf{pbest}_i$ < **gbest**)

Then **gbest** = $\mathbf{pbest}_i$

Position and velocity are then updated as per the following equations:

$\mathbf{V}_i^{k+1} = \mathbf{w} * \mathbf{V}_i^k + \boldsymbol{c}\mathbf{1}* \text{rand}_1 * (\mathbf{pbest}_i - \mathbf{X}_i^k) + \text{c2}* \text{rand}_2 * (\mathbf{gbest} - \mathbf{X}_i^k)$

$\mathbf{X}_i^{k+1} = \mathbf{X}_i^k + \mathbf{V}_i^{k+1}$

end

best individual from swarm **S** is chosen as the output

Figure 8. Procedure for Particle Swarm Optimization

Table 3. Parameters of GA & PSO used for training the MLP

Population size:	60	Swarm size:	10
Crossover probability:	80 percent	Particle dimension:	45
Mutation probability:	20 percent	Initial weight (w1):	0.9
Selection:	fitness-proportional	Final weight (w2):	0.4
Number of generations:	150	Constriction factor:	0.7298
Chromosome:	10 bit binary number	Weighting factor (c1 = c2):	1.4962
Initialization method:	Gaussian random	Particle position range:	0 – 0.1
Fitness function:	$1 / (1+MSE^{2)}$	Particle velocity range:	0 – 0.001
		Number of iterations (k_{max}):	100
		Initialization method:	Gaussian random
		Weighting function:	$w1-((w1-w2)/ k_{max}) * k$