Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, Qingdao, 11-14 July 2010

ARTIFICIAL NEURAL NETWORK FOR LOAD FORECASTING IN SMART GRID

HAO-TIAN ZHANG, FANG-YUAN XU, LONG ZHOU

Energy System Group，City University London，Northampton Square，London，UK
E-MAIL: abhb@city.ac.uk, abcx172@city.ac.uk, long.zhou.1@city.ac.uk

Abstract:

It is an irresistible trend of the electric power improvement for developing the smart grid, which applies a large amount of new technologies in power generation, transmission, distribution and utilization to achieve optimization of the power configuration and energy saving. As one of the key links to make a grid smarter, load forecast plays a significant role in planning and operation in power system. Many ways such as Expert Systems, Grey System Theory, and Artificial Neural Network (ANN) and so on are employed into load forecast to do the simulation. This paper intends to illustrate the representation of the ANN applied in load forecast based on practical situation in Ontario Province, Canada.

Keywords:

Load forecast; Artificial Neuron Network; back propagation training; Matlab

1. Introduction

Load forecasting is vitally beneficial to the power system industries in many aspects. As an essential part in the smart grid, high accuracy of the load forecasting is required to give the exact information about the power purchasing and generation [2] in electricity market, prevent more energy from wasting and abusing and making the electricity price in a reasonable range and so on. Factors such as season differences, climate changes, weekends and holidays, disasters and political reasons, operation scenarios of the power plants and faults occurring on the network lead to changes of the load demand and generations [1].

Since 1990, the artificial neural network (ANN) has been researched to apply into forecasting the load [2]. “ANNs are massively parallel networks of simple processing elements designed to emulate the functions and structure of the brain to solve very complex problems” [3]. Owing to the transcendent characteristics, ANNs is one of the most competent methods to do the practical works like load forecasting.

This paper concerns about the behaviors of artificial neural network in load forecasting. Analysis of the factors affecting the load demand in Ontario, Canada is made to give an effective way for load forecast in Ontario.

2. Back Propagation Network

2.1. Background

Because the outstanding characteristic of the statistical and modeling capabilities, ANN could deal with non-linear and complex problems in terms of classification or forecasting [4]. As the problem defined, the relationship between the input and target is non-linear and very complicated. ANN is an appropriate method to apply into the problem to forecast the load situation.

For applying into the load forecast, an ANN needs to select a network type such as Feed-forward Back Propagation, Layer Recurrent and Feed-forward time-delay and so on. To date, Back propagation is widely used in neural networks, which is a feed-forward network with continuously valued functions and supervised learning [2]. It can match the input data and corresponding output in an appropriate way to approach a certain function which is used for achieving an expected goal
with some previous data in the same manner of the input.

2.2. Architecture of back propagation algorithm

Figure 1 shows a single Neuron model of back propagation algorithm. Generally, the output is a function of the sum of bias and weight multiplied by the input. The activation function could be any kinds of functions. However, the generated output is different.

Owing to the feed-forward network, in general, at least one hidden layer before the output layer is needed. Three-layer network is selected as the architecture, because this kind of architecture can approximate any function with a few discontinuities [5]. The architecture with three layers is shown in Figure 2 below:

978-1-4244-6527-9/10/$26.00 ©2010 IEEE

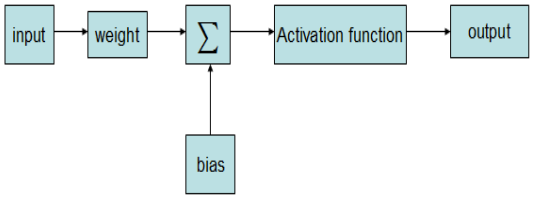

Figure 1. Neuron model of back propagation algorithm

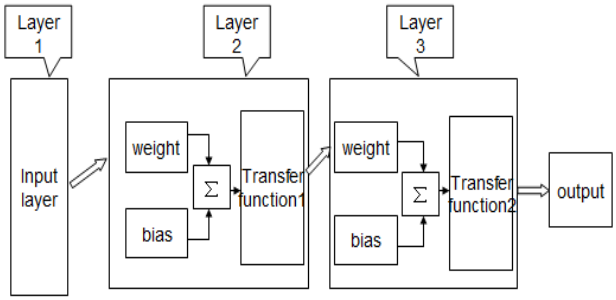

Figure 2. Architecture of three-layer feed-forward network

Basically, there are three activation functions applied into back propagation algorithm, namely, Log-Sigmoid, Tan-Sigmoid, and Linear Transfer Function. The output range in each function is illustrated in Figure 3 below.

2.3. Training function selection

Algorithms of training function employed based on back propagation approach are used and the function was integrated in the Matlab Neuron network toolbox.

3. Training Procedures

3.1. Background analysis

The neural network training is based on the load demand and weather conditions in Ontario Province, Canada which is located in the south of Canada. The region in Ontario can be divided into three parts which are southwest, central and east, and north, according to the weather conditions. The population is gathered around southeastern part of the entire province, which includes two of the largest cities of Canada, Toronto and Ottawa.

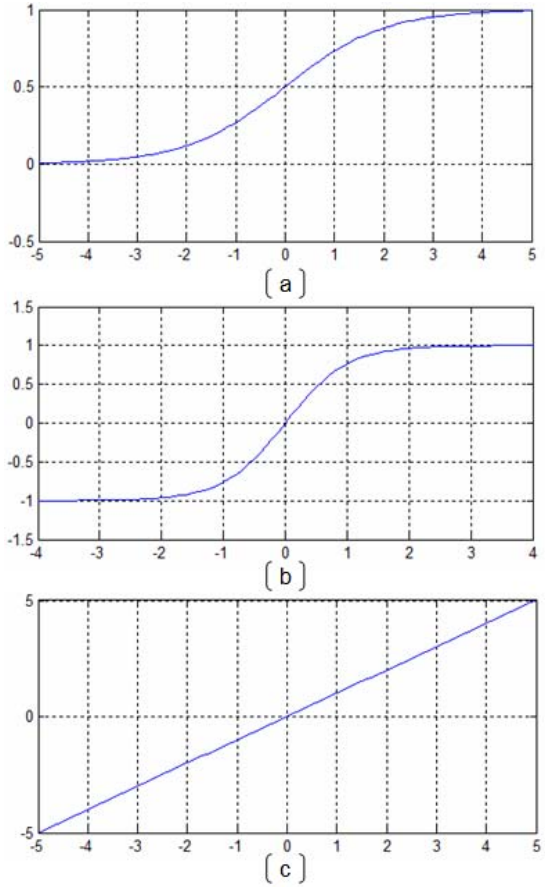

Figure.3. Activation functions applied in back propagation
(a)Log-sigmoid (b)Tan-sigmoid (c)linear function

TABLE.I. TRAINING FUNCTIONS IN MATLAB'S NN TOOLBOX[6]

Function name	Algorithm
trainb	Batch training with weight & bias learning rules
trainbfg	BFGS quasi-Newton backpropagation
trainbr	Bayesian regularization
trainc	Cyclical order incremental training w/learning functions
traincgb	Powell -Beale conjugate gradient backpropagation
traincgf	Fletcher-Powell conjugate gradient backpropagation
traincgp	Polak-Ribiere conjugate gradient backpropagation
traingd	Gradient descent backpropagation
traingdm	Gradient descent with momentum backpropagation
traingda	Gradient descent with adaptive lr backpropagation
traingdx	Gradient descent w/momentum & adaptive lr backpropagation
trainlm	Levenberg-Marquardt backpropagation
trainoss	One step secant backpropagation
trainr	Random order incremental training w/learning functions
trainrp	Resilient backpropagation (Rprop)
trains	Sequential order incremental training w/learning functions
trainscg	Scaled conjugate gradient backpropagation

3.2. Data Acquisition

The required training data can be divided into two parts: input vectors and output targets. For load forecasting, input

vectors for training include all the information of factors affecting the load demand change, such as weather information, holidays or working days, fault occurring in the network and so on. Output targets are the real time load scenarios, which mean the demand presented at the same time as input vectors changing.

Owing to the conditional restriction, this study only considers the weather information and logical adjustment of weekdays and weekends as the factors affecting the load status. In this paper, factors affecting the load changing are listed below:

(1). Temperature (° C)
(2). Dew Point Temperature (° C)
(3). Relative Humidity (%)
(4). Wind speed (km/h)
(5). Wind Direction (10)
(6). Visibility (km)
(7). Atmospheric pressure (kPa)
(8). Logical adjustment of weekday or weekend

According to the information gathered above, the weather information in Toronto taken place of the whole Ontario province is chosen to provide data acquisition. The data was gathered hourly according to the historical weather conditions remained in the weather stations. Load demand data also needs to be gathered hourly and correspondingly. In this paper, 2 years weather data and load data is collected to train and test the created network.

3.3. Data Normalization

Owing to prevent the simulated neurons from being driven too far into saturation [7], all of the gathered data needs to be normalized after acquisition. Like per unit system, each input and target data are required to be divided by the maximum absolute value in corresponding factor. Each value of the normalized data is within the range between -1 and +1 so that the ANN could recognize the data easily. Besides, weekdays are represented as 1, and weekend are represented as 0.

3.4. Neural network creating

Toolbox in Matlab is used for training and simulating the neuron network.

The layout of the neural network consists of number of neurons and layers, connectivity of layers, activation functions, and error goal and so on. It depends on the practical situation to set the framework and parameters of the network.

The architecture of the ANN could be selected to achieve the optimized result. Matlab is one of the best simulation tools to provide visible windows. Three-layer architecture has been chosen to give the simulation as shown in Figure 2 above. It is adequate to approximate arbitrary function, if the nodes of the hidden layer are sufficient [8].

Due to the practical input value is from -1 to +1, the transfer function of the first layer is set to be tan sigmiod, which is a hyperbolic tangent sigmoid transfer function. The transfer function of the output layer is set to be linear function, which is a linear function to calculate a layer's output from its net input [9]. There is one advantage for the linear output transfer function: because the linear output neurons lead to the output take on any value, there is no difficulty to find out the differences between output and target.

The next step is the neurons and training functions selection. Generally, Trainbr and Trainlm are the best choices around all of the training functions in Matlab toolbox.

Trainlm (Levenberg-Marquardt algorithm) is the fastest training algorithm for networks with moderate size. However, the big problem appears that it needs the storage of some matrices which is sometimes large for the problems [10] [11]. When the training set is large, trainlm algorithm will reduce the memory and always compute the approximate Hessian matrix with $n \times n$ dimensions [11]. Another drawback of the trainlm is that the over-fitting will occur when the number of the neurons is too large. Basically, the number of neurons is not too large when the trainlm algorithm is employed into the network.

Trainbr (Bayesian regularization) is a modified algorithm of the Levenberg-Marquardt training method to create networks which generalize well so that the optimal network architecture can be easily determined. Impacts from effectively used weights and biases of the network can be seen clearly by this algorithm. And the number of the effective weights and biases will not change too much when the dimension of the network is getting large. The trianbr algorithm has the best performance after the network input and output normalized into the range from -1 to +1. An important thing when using trainbr should be mentioned is that the algorithm should not stop until the effective number of parameters has converged [12]. More details are available in Matlab neural network toolbox.

Number of neurons in the first layer also can be selected to optimize the network so that an expected result can be made. Generally speaking, the more complicated architecture of the network is, the more accurate the output result will be, however, the higher chances will the algorithm such as trainlm with over-fitting. In this paper, the number of neurons

is 8 in trainlm algorithm, and 30 in trainbr algorithm.

3.5. Neural network training

Before training, the network needs to be initialized first. The network initialization is not only influencing the final local minimum, but also affecting the speed of convergence, the convergence probability and generalization [13].

The information on weather conditions in 2007 hourly and weekday and weekend logic in Ontario are defined as training input; the load demand changes in 2007 hourly in Ontario are defined as training target. The training performances of trainlm algorithm and trainbr algorithm are shown in Figure 4 and 5, respectively. As can be seen in these plots, the mean squared error is decreasing from a large value to a smaller value.

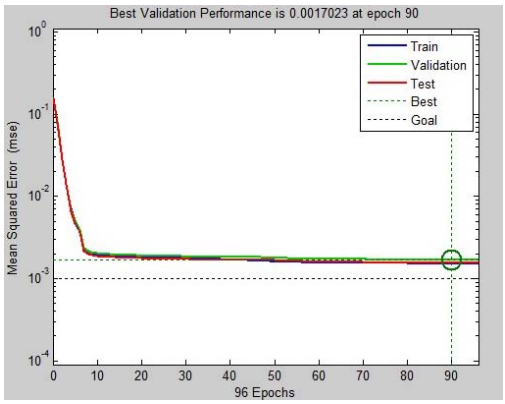

Figure 4 trainlm algorithm performance plot

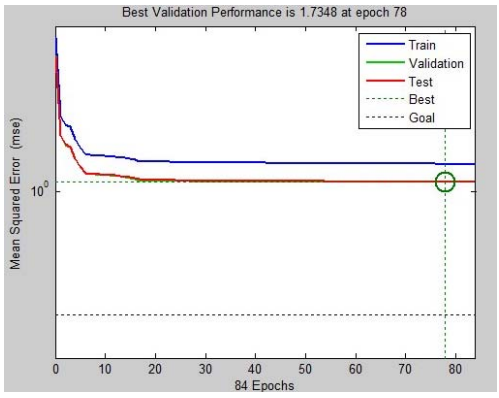

Figure 5. trainbr algorithm performance plot

For both training algorithms, namely, trainbr and trainlm, the procedure will stop when any of the conditions occurs:

(1). Epochs reached the maximum value

(2). Time approaches to the preinstalled value

(3). Goal error is minimized

(4). Gradient is decreased to min_grad

(5). Mu exceeds mu_max

(6). Validation performance has increased more than max_fail times since the last time it decreased (when using validation) [14]

There is no difficult to find out that the trainlm performance plot stopped because of meeting the error goal which is set as 0.001; the trainbr performance stopped owing to the validation check times is more than the max_fail times.

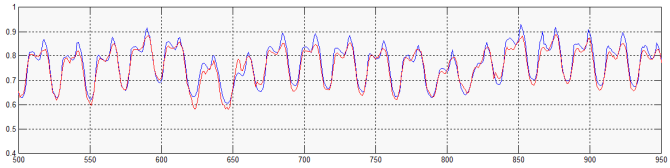

Figure 6. Training result and training target by trainbr algorithm with 8 neurons

From Figure 6, comparison of training result and training target are made to check out the performance of the algorithm applied on load forecasting. It is obvious that the training result meet the target in general. The network test simulation should be made in order to find out the performance in a real problem.

3.6. Neural network simulation

The network is required to check whether it can achieve the expectation after training. Another set of input vectors and demand scenarios are needed to test the network. Comparison needs to be made to check out the difference between the test output and real demand.

In this project, the information on weather conditions in 2008 hourly and weekday and weekend logic in Ontario are used as simulation input, and the load demand scenarios in 2008 hourly in Ontario are used as the simulation target. After the simulation, a set of output could be obtained through the trained neural network. The simulation output and the simulation target are used to check the mean squared error to analyze the extent of succeed with neural network application. Mean squared error could be calculated as:

$$\mathrm{MSE} = \frac{\mathrm{Mean}\left(\mathrm{se}\right)}{\max\left(\mathrm{test\ target}\right)}$$

Where Mean(se) is the mean value of the difference between the simulation output and the test target;

Max(test target) is the maximum value of the test

target.

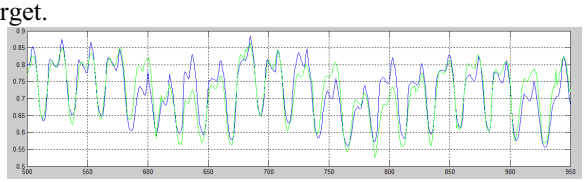

Figure 7. The simulation result of the trainbr algorithm with 8 neurons

Figure 7 shows a sample of the simulation results which is applying the same network as the training simulation in Figure 6. The green track is the test simulation result and the blue track is the real load demand which is provided by electric industry in Ontario. The horizontal is presenting time, and ordinate is presenting the load which has been normalized. The less the mean squared error is, the better the created neuron network can perform.

4. result comparison

Table 2 illustrates the MSE of the trainoss algorithm, trainbr algorithm and trainlm algorithm when the number of the neurons in the hidden layer is increasing. Each network has been trained 10 times to achieve the global minimum. It is obvious that trainlm and trainbr have better performance than trainoss.

TABLE.II. MEAN SQUARED ERRORS COMPARISON OF TRAINBR, TRAINOSS AND TRAINLM

Neurons	MSE(%)		
	TrainOSS	TrainBR	TrainLM
5	5.76	5.24	5.55
8	6.31	5.17	5.19
10	5.95	5.15	5.21
18	5.62	5.17	5.17
30	5.69	5.25	5.21

Figure 8-9 demonstrate two of the best results with different training algorithms and different neuron number. As can be seen in both figures, most of the test simulation results could meet the target very well. However, there are still some simulation part didn't follow the real target. It may be because the factors which haven't been taken into account, such as disasters, failure of the electric network or some national holidays which is not mentioned on input vectors.

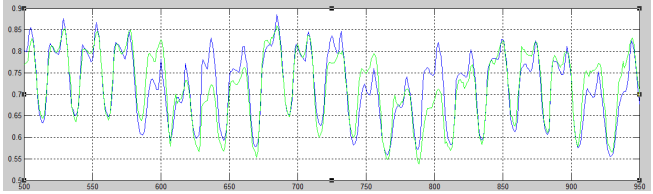

Figure 8. the simulation result of the trainlm algorithm with 8 neurons

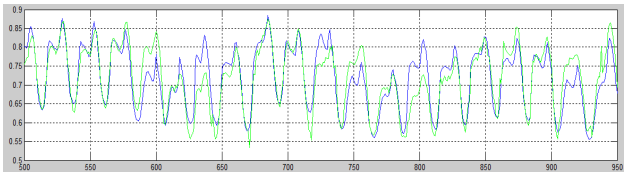

Figure 9. The simulation result of the trainbr algorithm with 10 neurons

Figure 10 aims to compare the two algorithms with the same number of neurons applied into the networks. The blue track is the test simulation target, the red one is the result simulated by trainlm, and the green one is the result simulated by trainbr. The simulation result of trainbr is much closer to the test target than that of trainlm, although the error goal of trainbr is much larger. The trend of the trainbr followed the general track of the test target, whereas the trainlm can only simulate the maximum and minimum value per cycle.

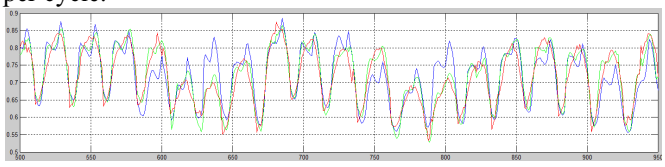

Figure 10 The simulation results of both trainbr and trainlm with 8 neurons

The comparison of trainlm with 8 neurons and 30 neurons is presented in Figure 11. Theoretically, more neuron applied into the neural network could make the performance better. However, over-fittings could occur much more obviously. In Figure 11, red track which is representing the trainlm with 30 neurons is much closer to the test target which is represented in blue. Over-fitting is the main problem which cannot be avoided.

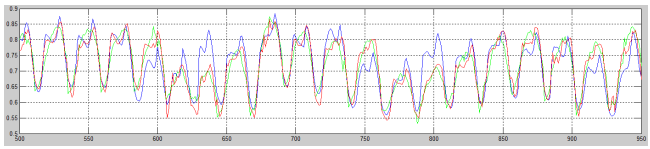

Figure 11 The simulation results of trainlm with 8 neurons and 30 neurons

From the performance, trainbr could be the optimized algorithm that can be employed in load forcast by back propagation. There is an argument that the MSE of the trainlm algorithm could be less than that of trianbr when the neurons are increasing. However, there is a big problem could not be neglected, namely, over-fitting, which could decrease the quality of the simulation.

5. Conclusion

This paper focused on the behaviors of different training algorithms for load forecasting by back propagation algorithm in Neural Network. Due to the characteristic of imitating the mode of human beings' thinking, ANN can learn the relationship between input and output. Thus, same function of the relationship can be applied into practical situation to find out the output according to the information known already as input. After research, trainbr algorithm which is integrated in Neural Network Toolbox in Matlab is regarded as one of the best choice to do load forecast. If the accurate results are required to forecast the load, more neurons are needed to apply into the network architecture. On the other hand, over-fitting must be considered about to ensure the network simulate the load situation well. Owing to the condition limitation, the input vectors did not take all of the information into account. A few of the simulation part didn't meet the real demand very well, even large squared error occurred. If the information was gathered enough and the networks were trained more meticulous, better result could be obtained to apply into load forecasting for smart grid.

Acknowledgement

The authors would like to thank Professor Loi Lei Lai for his supervision at City University London.

References

[1] Grzegorz Dudek "Artificial Immune System for Short-Term Electric Load Forecasting," Rutkowski et al. (Eds.): ICAISC 2008, LNAI 5097, pp. 1007–1017, 2008.

[2] Eugene A. Feinberg and Dora Genethliou, "Load Forecasting," in Applied Mathematics for Restructured Electric Power Systems: Optimization, Control, and Computational Intelligence (J. H. Chow, F.F. Wu, and J.J. Momoh, eds.), Spinger, pp. 269-285, 2005.

[3] Loi Lei Lai, "Object-Oriented Analysis, Design and Development of Artificial Neural Networks" in "Intelligent System Applications in Power Engineering" QA76.618L35, pp. 1-35, 1998.

[4] Kevin N. Gurney, "Neural networks – an overview" in "An introduction to neural networks" ISBNs: 1-85728-673-1 HB, 1-85728-503-4 PB, pp. 1-6, 1997.

[5] Howard Demuth, Mark Beale and Martion Hagan, "Backpropagation" in "Neural Network Toolbox 6 User's Guide, Matlab" pp. 5-10 – 5-14.

[6] nn_summary. "http://www.cs.ucr.edu/~vladimir/cs171" visited on 3rd of April.

[7] S.J. Kiartizis, A.G. Bakirtzis and V. Petridis "Short-term load forecasting using neural networks" Elsevier Science S.A. 1995.

[8] Derrick Nguyen and Bernard Widrow "Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights", nninitialization. "http://www.stanford.edu/class/ee373b" visited on 3rd of April.

[9] Nikhil, Bestamin Özkaya, Ari Visa, Chiu-Yue Lin, Jaakko A. Puhakka, and Olli Yli-Harja, "An Artificial Neural Network Based Model for Predicting H2 Production Rates in a Sucrose-Based Bioreactor system," World Academy of Science, Engineering and Technology 37, 2008.

[10] H. T. Li, Y. Deng and J. T. Niu, "Artificial Neural Networks-Modeling, Programming and Application in Material Hot Working," ACTA METALLURGICA SINICA (ENGLISH LETTERS) Vol. 13 No. 2 pp 825-831 April 2000.

[11] Howard Demuth, Mark Beale and Martion Hagan, "Fast Training, Backpropagation" in "Neural Network Toolbox 6 User's Guide, Matlab" pp. 5-30 – 5-33.

[12] Howard Demuth, Mark Beale and Martion Hagan, "Improving Generaliztion, Backpropagation" in "Neural Network Toolbox 6 User's Guide, Matlab" pp. 5-55 – 5-58.

[13] Mercedes Fernandez-Redondo and Carlos Hernandez-Espinosa "Weight Initialzation Methods for Mutilayer Feedforward," European Symposium on Artificial Neural Networks Bruges (Belgium), pp. 119-124, 25-27 April 2001.

[14] Howard Demuth, Mark Beale and Martion Hagan, "Functions-Alphabetical List" in "Neural Network Toolbox 6 User's Guide, Matlab" pp. 16-531