ISSN 2347 - 3983

International Journal of Emerging Trends in Engineering Research (IJETER), Vol. 3 No.6, Pages : 430 - 436 (2015)
Special Issue of NCTET 2K15 - Held on June 13, 2015 in SV College of Engineering, Tirupati
http://warse.org/IJETER/static/pdf/Issue/NCTET2015sp77.pdf

Online load forecasting for power system planning using Artificial Neural Networks

Madhuranthaka T
Associate Professor
S V College Of Engineering
e-mail:tarasimadhu@gmail.com

Tagore Sairam K
Student
S V College Of Engineering
e-mail:tagoresairamforu@gmail.com

Muni Meena B
Student
S V College Of Engineering
e-mail:medhini.meena@gmail.com

ABSTRACT: Forecasting is phenomenon of expecting the future load on the power system. To provide uninterrupted power supply to consumers, there must have a clear idea about load variations on the power system. Load forecasting is the technique used for prediction of electrical load by estimating the load demand. In a deregulated market, it is much needed for a generating company to know about the market load demand for generating nearer to accurate power. There are many traditional techniques for load forecasting but to overcome the disadvantages of traditional methods, we proposed a new technique which uses artificial neural networks for estimation of future load. There are several neural networks used for load forecasting. In this paper we used multi layer perceptron network for load forecasting. This network is trained by using back propagation learning. In this paper Levenberg-Marquardt optimization technique which gives best learning rates is used. Back propagation learning algorithm is considered to solve load forecasting problem. Load forecasting problem is simulated using ANN toolbox in MATLAB 2013a version.

Keywords: Load Forecasting, Neural Network, Power System, Back Propagation, Multilayer Perceptron, Neural Network Time Series.

I INTRODUCTION

Load forecasting has always been the essential part of an efficient power system planning and operation. The total amount of electric power [in MW] consumed in an electrical power system must be balanced with an equal amount of generated power. There is no efficient way of storing large amounts of electrical energy. To maintain this power balance between production and consumption the power input to the power system must be controlled. Thus an accurate load forecasting not only reduces the generation cost in a power system, but also provides a good principle of effective operation.

With the deepen reform of electricity, the formation of power market and self-financing of electricity enterprises, power load forecasting becomes more and more important. To improve the accuracy of power, load forecasting is a valuable research. An accurate forecast of electricity prices has become a very important tool for producers and consumers. A producer needs to forecast electricity prices to optimally schedule its electric energy resources

The need for accurate load forecasts will increase in the future because of the dramatic changes occurring in the structure of the utility industry due to deregulation and competition. This environment compels the utilities to operate at the highest possible efficiency which requires accurate load forecasts. For decades, the problem of improving the accuracy of load forecasts has been an important topic of research.

Several electric power companies have adopted the conventional methods for forecasting the future load. In conventional methods, the models are designed based on the relationship between load power and factors influencing load power. The conventional method has the advantage that we can forecast load power with a simple prediction model.

However, since the relationship between load power and factors influencing load power is nonlinear, it is difficult to identify its nonlinearity by using conventional methods.

Various types of load forecasting methodologies have been adopted based on predicting time. Based on different predicting time, it can be divided into long-term load forecasting, mid-term forecasting, short-term forecasting and ultra-short-term forecasting. Midterm and long-term forecasting mainly used in power factory macro control, and their forecasting time ranges are respectively from one month to twelve months and from one month to ten months respectively. The short-term forecasting can be used in generators macroeconomic control, power exchange plan and so on. And the prediction is from one day to seven days in the future, or a little longer time. Whereas the ultra-short-term forecasting can predict the situation in a day or in an hour, and it's mainly used in Prevention and control emergency treatment and frequency control.

Generally speaking, long-term accuracy of the forecast will be lower, while short-term accuracy will be higher. So we adopted short term load forecasting and it is done by using neural networks.

Neural networks are capable of handling nonlinearities between electric load, time and weather factors that affect the load. But the weather factors somehow lack to fully handle unusual changes that occur in the environment. So we considered only time and load as parameters for load forecasting.

II DESIGN OF ARTIFICIAL NEURAL NETWORK

This section describes the step by step procedures for training the neural network to learn from the Year 2004 hourly load data and average temperatures, in order to forecast next day's load demand. The MATLAB ANN toolbox was utilized in designing the network architecture. The Multilayer Feed forward Network architecture with two layers was designed. The layers include the Hidden layer and the output layer. The input consists of daily 24 hour load data for 12 months of the year 2004 and daily average maximum temperature altogether making 25 inputs rows by 365 days. The output layer will be a day's 24 hours load forecast for the utility company. The Target data is the same as the input's daily 24 hours load data. The Transfer function used in the two layers is the log- sigmoid function for the hidden layers and the Purelin function at the output layers; this is to enable the network to be able to take care of any non- linearities in the input data and at the output, to be able to give a wide range of values. However, there is no theoretical approach to calculate the appropriate number of hidden layer nodes. This number was determined using a similar approach for training epochs i.e., by examining the Mean squared error (MSE) over a validation set for a varying number of hidden layer nodes whereupon a number yielding the smallest error was selected. The pattern of connectivity characterizes the architecture of the network. A unit in the output layer determines its activity by following a two-step procedure. First, it computes the total weighted input X_j, using the formula given below by examining the Mean squared error (MSE) over a validation set for a varying number of hidden layer nodes whereupon a number yielding the smallest error was selected. The pattern of connectivity characterizes the architecture of the network. A unit in the output layer determines its activity by following a two-step procedure. First, it computes the total weighted input W_{ij}, using the formula below

$$X_j = \sum_i y_i W_{ij}$$

Where Yiis the activity level of the W_{ij}^{th} unit in the previous layer and $W_{ij}$is the weight of the connection between the i th and the j th unit. Next, the unit calculates the activity $y_j$using some function of the total weighted input. Typically we use the sigmoid function below

$$y_j = \frac{1}{1 + e^{-x_j}}$$

Once the activities of all output units have been determined, the network computes the error E, which is defined by the expression below

$$E = \frac{1}{2}\sum (y_i - d_i)^2$$

Where y_j is the activity level of the jth unit in the top layer and dj is the desired output of the jth unit. The

Levenberg Marquardt (lm) back-propagation algorithm consists of six computational steps as described below:

1. It computes how fast the error changes as the activity of an output unit are changed. This error derivative (EA) is the difference between the actual and the desired activity (eq 4)

$$EA_j = \frac{\partial E}{\partial y_j} = y_j - d_j$$

2. It then computes how fast the error change as the total input received by an output unit is changed. This quantity (EI) is the answer from step 1 multiplied by the rate at which the output of a unit changes as its total input is changed.

$$EI_j = \frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \times \frac{dy_j}{dx_j} = EA_j y_j (1 - y_j)$$

3. It then computes how fast the error changes, as a weight on the connection into an output unit is changed. This quantity (EW) is the answer from step 2 multiplied by the activity level of the unit from which the connection emanates .

$$EW_{ij} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial x_j} \times \frac{dx_j}{dW_{ij}} = EI_j y_j$$

4. It then computes how fast the error change as the activity of a unit in the previous layer is changed. This crucial step allows back propagation to be applied to multilayer networks. When the activity of a unit in the previous layer changes, it affects the activities of all the output units to which it is connected. So to compute the overall effect on the error, we add together all these separate effects on output units. But each effect is simple to calculate. It is the answer in step 2 multiplied by the weight on the connection to that output unit.

$$EA_j = \frac{\partial E}{\partial y_j} = \sum_j \frac{\partial E}{\partial x_j} \times \frac{dx_j}{dy_j} = \sum_j EI_j W_{ij} .$$

By using steps 2 and 4, we can convert the EAs of one layer of units into EAs for the previous layer. This procedure can be repeated to get the EAs for as many previous layers as desired. Once the EA of a unit is known, we can use steps 2 and 3 to compute the EWs on its incoming connections.

5. It then advances to compute the H matrix and the gradient. These are necessary in order to approach second-order training speed without having to compute the Hessian matrix. The performance function has the form of a sum of squares (MSE) and as such the Hessian matrix can be approximated as given in

$$H = J^T J$$

and then the gradient can be computed using equation

$$g = J^T e$$

as follows:

where J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard back propagation technique that is much less complex than computing the Hessian matrix.

6. Finally, the LevenbergMarquadt-lmalgorithm uses this approximation to the Hessian matrix in the following Newton-like update .

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e .$$

Where $X_{k+1}$is the updated value of the network weight or bias and $X_k$is the current weight or bias value. When the scalar µ is zero, this is just Newton's

ISSN 2347 - 3983

International Journal of Emerging Trends in Engineering Research (IJETER), Vol. 3 No.6, Pages : 430 - 436 (2015)
Special Issue of NCTET 2K15 - Held on June 13, 2015 in SV College of Engineering, Tirupati
http://warse.org/IJETER/static/pdf/Issue/NCTET2015sp77.pdf

method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift toward Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always reduce in successive iterations of the algorithm.

III STEPS LOAD FORECASTING PROCEDURE

Step 1: Input and target data is copied into excel file

Step 2: convert all the values into per unit values and multiply per unit month, per unit date, per unit time to get per unit input

Step 3:Inputs and outputs are added to MATLAB workspace

Step 4: Type ntstool in command window to open time series tool box.

Step 5: Per unit input and output data is assigned into corresponding values

Step 6: Set the percentage of data used for validation and test data of the network

Step 7: Representation of hidden layers and delays of the network

Step 8: Saving of the results and generation of simple script.

Step 9: After saving the results click on simple script. The script is automatically generated. This is the script for the complete procedure we have done.

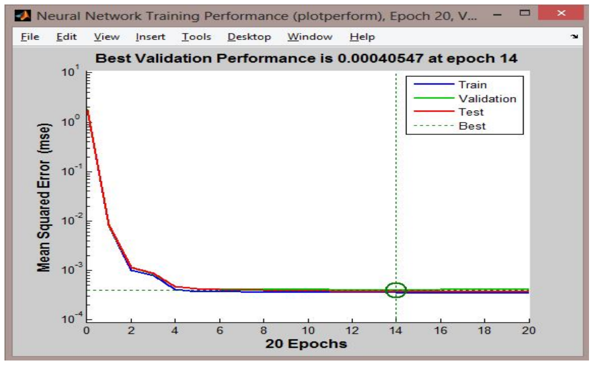
Step 10: The script generated is saved and is run by pressing F5. A numerous variables are stored in workspace. In variable $t_s$which is the target series has NaN at cell 8783. We have to predict that value based on previous values.

Step 11: Open Y_s(output series) in work space in that we can see that predicted value of 8783.

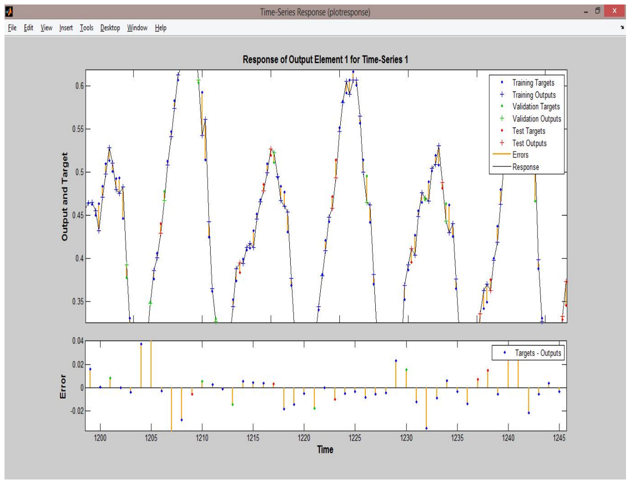
If the predicted output value in the variable y_s is used as input, the load prediction of next hour is achieved. By continuing this process we can get a continuous hourly load prediction.

IV.RESULTS

Figure shows the Performance plot of the network. We can see that mean square error is damped for increased iterations and reached its minimum value at 14th epoch.



In this figure the nonlinear load data is fitted into a graph and the graphs shows error, validation, testing and performance. It can be seen that the error is around 0.01 to 0.04 which shows the accuracy of the nonlinear curve fitting.



V. CONCLUSION

- A load forecasting model was designed using MATLAB R2013a ANN Toolbox.
- The implementation of the network architecture, training of the Neural Network and simulation of test results were all successful with a very high degree of accuracy resulting into hourly load output.
- The results suggest that ANN model with the developed structure can perform good prediction with least error and finally the neural network could be an important tool for one hour ahead load forecasting.

VI REFERENCES

[1] Modern power system analysis by D P Kothari and I J Nagarath ,Tata McGraw-Hill Education, 2011 - 776 pages

[2] Power system Analysis by John J Grainger, William D Stevenson, TMC Companies, McGraw-Hill Education (India) Pvt Limited, 01-Dec-2003 - 785 pages

[3] Power System Analysis and Design by B.R.Gupta, S. Chand Limited, 01-Jan-2008 - 642 pages

[4] Power System Analysis Operation and Control – A. Chakravarthi and S. Halder, PHI Learning Pvt. Ltd., 2010 - 1270 pages

[5] Neural Networks – James A Freeman and Davis Skapura, Pearson Education, 1991 – 461 pages.

[6] Neural Networks – Simon Hakins , Pearson Education, 1999 - 842 pages

[7] Introduction To Neural Networks Using MATLAB 6.0 By S N Sivanandam, S Sumathi, S N Deepa, 2006, 656 pages.

[8] Short-term Load Forecasting Using An Artificial Neural Network By K. Y. Lee, Y. T. Cha, J. H. Park.

[9] Short-term Load Forecasting Using Artificial Neural Network Muhammad Buhari, Member, Iaeng And Sanusi Sani Adamu.

[10] Composite Modeling For Adaptive Short-term Load Forecasting By J. H. Park Y. M. Park

[11] Short-term Load Forecast Of A Low Load factor Power System For Optimization Of Merit Order Dispatch Using Adaptive Learning Algorithm

[12] One-hour-ahead Load Forecasting Using Neural Network By TomonobuSenjyu

[13] Load data from www.Kaggle.com , the world's largest community of data scientists

ISSN 2347 - 3983

International Journal of Emerging Trends in Engineering Research (IJETER), Vol. 3 No.6, Pages : 430 - 436 (2015)
Special Issue of NCTET 2K15 - Held on June 13, 2015 in SV College of Engineering, Tirupati
http://warse.org/IJETER/static/pdf/Issue/NCTET2015sp77.pdf

Table1: Input Data of January Month2015

month	1	1	1	1	1	1	1	1	1
date	1	1	1	1	1	1	1	1	1
time	1	2	3	4	5	6	7	8	9
load	16,853	16,450	16,517	16,873	17,064	17,727	18,574	19,355	19,534
month	1	1	1	1	1	1	1	1	1
date	1	1	1	1	1	1	1	1	1
time	10	11	12	13	14	15	16	17	18
load	18,611	17,666	16,374	15,106	14,455	13,518	13,138	14,130	16,809
month	1	1	1	1	1	1			
date	1	1	1	1	1	1			
time	19	20	21	22	23	24			
load	18,150	18,235	17,925	16,904	16,162	14,750			

Table 2: Per Unit Values

Pu month	0.083333	0.083333	0.083333	0.083333	0.083333	0.083333	0.083333	0.083333
Pu date	0.032258	0.032258	0.032258	0.032258	0.032258	0.032258	0.032258	0.032258
Pu time	0.041667	0.083333	0.125	0.166667	0.208333	0.25	0.291667	0.333333
Pu load	0.421325	0.41125	0.412925	0.421825	0.4266	0.443175	0.46435	0.483875

Hidden neurons	Epoch	Time	Mean Square Error(MSE)	Gradient	Target output(t_s)	Actual output(y_s)	Error
1	14	1 sec	0.000458	0.000183	0.3878	0.3864	-0.0014
25	60	11 sec	0.000377	0.000120	0.3878	0.3889	0.0011
50	177	54 sec	0.000342	$1.81*10^{-5}$	0.3878	0.3793	0.0085
75	61	27 sec	0.000355	$7.54*10^{-5}$	0.3878	0.3923	0.047
100	39	23 sec	0.000380	0.000173	0.3878	0.3976	0.0098
125	55	41 sec	0.000359	0.000630	0.3878	0.4191	0.0313
150	24	22 sec	0.000349	0.000948	0.3878	0.3671	-0.0204
175	31	36 sec	0.000365	0.00o207	03878	0.3673	-0.0205
200	77	120 sec	0.000341	0.000229	0.3878	0.4112	0.4112

Table 3: Comparison Table