ISSN 2278-3091
Volume 8, No.2, March - April 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse14822019.pdf
https://doi.org/10.30534/ijatcse/2019/14822019

Comparative Study on Backpropagation and Levenberg Marquardt Algorithm on Short Term Load Forecasting

Manish Kumar Singla[1], Jyoti Gupta[2], Parag Nijhawan[3]
[1,2,3]Thapar Institue of Engineering and Technology Patiala, India
Email: [1]msingla0509@gmail.com
[2]jgupta118207@gmail.com
[3]parag.nijhawan@rediffmail.com

ABSTRACT

Variable electrical load and ever increasing load demand needs to be predicted or forecasted in order to avoid the energy crisis. This paper presents a comparison between the back propagation algorithm and Levenberg marquardt algorithm for short term load forecasting. The live load data from a typical 66 kV sub-station of the Punjab State Power Corporation Limited (PSPCL) for a selected site is procured for the presented simulation study. The collected live data is divided into three categories, i.e. validation, training and testing for the simulation study considering neural network approach. To check the performance of the two proposed methods, different parameters or errors are used like MSE (Mean Square Error), RMSE (Root Mean Square Error), MAE (Mean Percentage Error), SSE (Sum of Square of Error) and MAPE (Mean Percentage Absolute Error). This would help to ascertain the fluctuation in electric load well in advance, and making a room for preparation to meet the sudden load demand thereby, meeting the expectations of an effective load forecasting with numerous applications in power system arena.

Key words: electric load, neural network, back propagation, Levenberg marquardt, short term load forecasting.

1. INTRODUCTION

Short-term load forecasting (STLF) starts with a short interval of time to couple of days which play a vital role in the secure and effective operation of power systems. It is helpful in solving the unit commitment, interchange evaluation, security assessment and hydrothermal coordination problems related to the power system network.

Forecast error in load predictions ends up in better-operating costs. Over prediction of load ends up in a unnecessary increase in reserves and thus operating costs. Under prediction of load results could lead to a failure in supplying the required reserves, thereby attracting higher costs because of the use of costly peaking units. Several STLF models have been proposed in the previous era and are mainly classified into two broad categories viz. regressive model [1-5].

Continuous technological advancements have lead artificial intelligence, to penetrated into almost every aspect of real day life. In this work, an attempt has been made to apply artificial neural network (ANN) to solve the STLF problem. The proposed technique provides an effective solution for accurate forecasting and effective training of the complete system. Neural network being a revolution in the artificial intelligence [6-10] eliminates the need of human experience as the system learns and matures by itself with the help of input and output relationships although, it does not allow the adoption of functional relationship between the previous load and the forecasted load. As of now, neural network gets trained by itself but previously past input-output patterns were required to be fed in the system for the effective training. Once trained, the system could effectively predict output according to the input.

The neural network based first STLF model comes in a picture three years back. Literature [6] gives a three year forecasting plan in terms of hours, peaks and regular days and the resulting error comes under two hundred while including three month load and temperature data as well.

2. SHORT TERM LOAD FORECASTING

Short-term load forecasting is essentially used for the small period of time, i.e. one hour to seven days to predict the load on the system. Electrical short-term load forecasting has a lot of importance [11-12]. As a result, it offers correct and beneficial managing of electrical utilities [13-14].

3. ARTIFICIAL NEURAL NETWORK

Motivation for the effective working in the field of neural network comes from the fact that the human brain can also be analyzed in a more accurate manner with the help of different methods except digital computers. It can also be considered as a complicated structure as well as non linear and parallel information processing system. It is more effective in comparison to conventionally developed digital computer due to having higher potential for developing its constituents called neurons for faster computational response.

Definition

Artificial neural network is highly inspired with the biological nerve cells as it composed of huge parallel processing system completely depends upon the interconnected network form by the every single neuron with its neighbor neuron to develop a capability of learning new things by analyzing previous situations.

Basic Neural Structure

The basic neuron structure of artificial intelligence is of two types i.e. Biological neural and artificial neural structure.

Biological Neural: Every single neuron varied in terms of shape and size in comparison to other and introducing cell body having several processes undergoing over its surface called neuritis re projected. The neurites, also known as dendrites, those are completely responsible for the bi directional connectivity. The basic structure of biological neuron is shown in Figure 1. The various parts of biological neuron are:

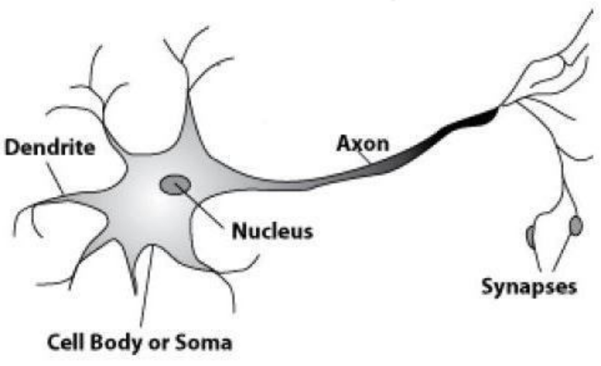

Figure 1: Biological Neural Network [15]

- Dendrite: The dendrites are the branched structure formed by the interconnected neurons to cause conduction of electrochemical stimulation within the neural based architecture. The electrochemical signals can be received with the help of synapses that are present throughout the dendrites for signal transfer. Due to further research it was found that dendrites support action potential and release neurotransmitters [15].

- Soma: This is where the signals from the dendrites are joined and passed on [15].

- Axon: Nerve fiber is known as axon. It is quite long, lean projection of nerve cells and also causes conduction of electrical impulse away from the soma [15]. Axons are also differentiated in comparison to dendrites in terms of shape or size and function. There are also few neurons having no axon still they transmit signal through their dendrites. No neurons in the artificial neural network contain more than a single axon but in some of the cases, several axons were found.

Artificial Neural: The artificial neural network is basically an interconnected network of several artificial constituents that worked in a similar manner as biological neurons called artificial neurons. The simple model of artificial neuron is shown in Figure 2.

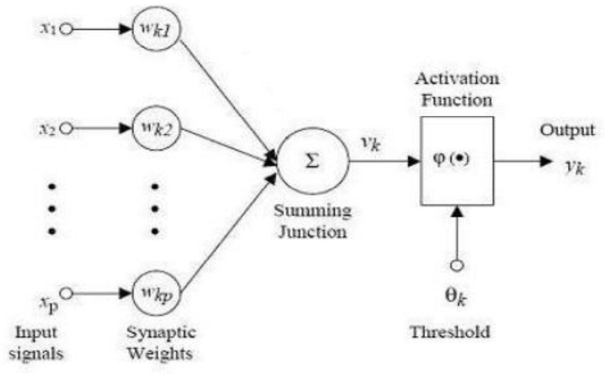

Figure 2: Artificial Neural Network [15]

Here, x_1, x_2,….,x_p are the p inputs to the artificial neuron. w_{k1}, w_{k2},……,w_{kp} are the weight attached to the input links.

Hence, the total input I received by the soma of the artificial neuron is

$$I = w_1x_1 + w_2x_2 + \dots + w_px_p$$
$$= \sum_{i=1}^{n} w_i x_i \quad (1)$$

4. BACKPROPAGATION

Back Propagation (BP) alludes to a wide group of ANN, whose design comprises of various interconnected layers. The learning algorithm of BP is based on the Deepest Descent technique. The proper number of hidden units limits the error of the non-linear function of high complexity. BP is a systematic technique of training multilayer ANN. It is designed on a high mathematical foundation and has excellent application potential [16]. The networks include sensory units that represent the input layer, one or more hidden layers of calculation nodes, and the output layer of the calculation nodes. Input signals on a layer-by-layer basis propagate through the network in the forward direction. These neural networks normally allude to as a multilayer perceptrons as shown in Figure 3.

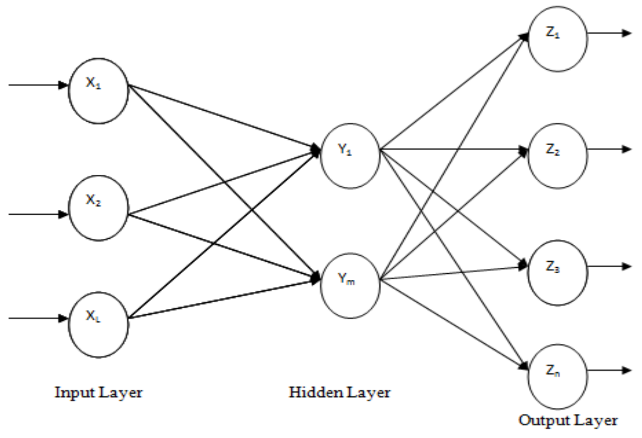

Figure 3: Multi-Layer Fed Forward Network [16].

Algorithm

Figure 4 represents the flowchart of the backpropagation training algorithm for a fundamental two-layer network as appeared in Figure 3. The BP algorithm is briefly explained below:

The error signal at the output for l^{th}neuron at n^{th}iteration:

$$\frac{\partial\xi(n)}{\partial y_l(n)} = \sum_m e_m \frac{\partial e_m(n)}{\partial v_m(n)}\frac{\partial v_m(n)}{\partial y_l(n)} \quad (2)$$

Total error energy obtained

$$\xi(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \quad (3)$$

The average square error energy is

$$\xi_{av} = \frac{1}{N}\sum_{n=1}^{N}\xi(n) \quad (4)$$

The input of the activation function with l^{th} neuron is

$$v_l(n) = \sum_{i=o}^{p} w_{li}(n)\, y_i(n) \quad (5)$$

The synaptic weight w_{l0} equals the bias b_l applied to neuron l. The functional signal $y_l(n)$ appearing at the output of l^{th} neuron at n^{th}iteration.

$$y_l(n) = \varphi(v_l(n)) \quad (6)$$

The chain rule is,

$$\frac{\partial\xi(n)}{\partial w_{li}(n)} = \frac{\partial\xi(n)}{\partial e_l(n)}\frac{\partial e_l(n)}{\partial y_l(n)}\frac{\partial y_l(n)}{\partial v_l(n)}\frac{\partial v_l(n)}{\partial w_{li}(n)} \quad (7)$$

$$\frac{\partial\xi(n)}{\partial e_l(n)} = e_l(n) \quad (8)$$

Differentiating both sides of Eq. (2) with respect to $y_j(n)$,

$$\frac{\partial e_l(n)}{\partial y_l(n)} = -1 \quad (9)$$

Differentiating Eq. (6) with respect to $v_j(n)$,

$$\frac{\partial y_l(n)}{\partial v_l(n)} = \varphi_l'(v_l(n)) \quad (10)$$

Differentiating Eq. (5) with respect to $w_{ji}(n)$,

$$\frac{\partial v_l(n)}{\partial w_{li}(n)} = y_l(n) \quad (11)$$

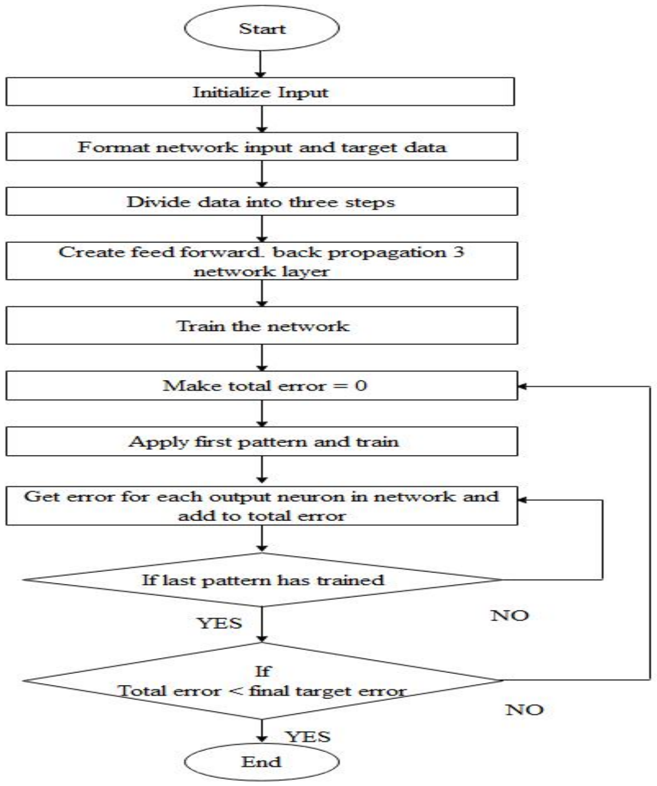
Start
Initialize Input
Format network input and target data
Divide data into three steps
Create feed forward. back propagation 3 network layer
Train the network
Make total error = 0
Apply first pattern and train
Get error for each output neuron in network and add to total error
If last pattern has trained
YES
NO
If Total error < final target error
NO
YES
End

Figure 4: Backpropagation Flowchart

Using Eq. (8) to (11) in (7),

$$\frac{\partial\xi(n)}{\partial w_{li}(n)} = -e_l(n)\varphi_l'(v_l(n))\, y_i(n) \quad (12)$$

By the delta rule

$$\Delta w_{li}(n) = -\eta\frac{\partial\xi(n)}{\partial w_{li}(n)} \quad (13)$$

η is the learning parameter of the backpropagation algorithm. Use Eq. (12) in (13),

$$\Delta w_{li}(n) = \eta\delta_l(n)\, y_i(n) \quad (14)$$

The local gradient $\delta_j(n)$ is defined by,

$$\delta_l(n) = \frac{\partial\xi(n)}{\partial v_l(n)}$$

$$= \frac{\partial \xi(n)}{\partial e_l(n)} \frac{\partial e_l(n)}{\partial y_l(n)} \frac{\partial y_l(n)}{\partial v_l(n)} \tag{15}$$

$$= e_l(n) \varphi_l'(v_l(n))$$

Case 1 Neuron *l* is an output node

When neuron *l* is located in the output layer of the network, it is supplied with the desired response. Use Eq. (2) to compute the error signal and as shown in Figure 5. For the calculation of local gradient, this technique was considered as the best suitable approach by using Eq. (15).

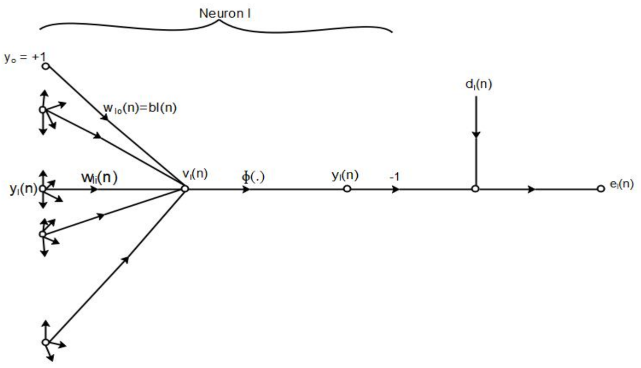

Figure 5: Output neuron j signal flow graph

The Case 2 Neuron *l* is a hidden node. When neuron *l* is located in a hidden layer of the network, there is no specified desired response of that neuron. Accordingly, the error signal for a hidden neuron would have to be determined in terms of the error signal of all the neurons as shown in Figure 6. According to Eq. (15) redefines the local gradient for hidden neuron *l* as

$$\delta_l(n) = -\frac{\partial \xi(n)}{\partial y_l(n)} \frac{\partial y_l(n)}{\partial v_l(n)} \tag{16}$$

$$\delta_l(n) = -\frac{\partial \xi(n)}{\partial y_l(n)} \frac{\partial y_l(n)}{\partial v_l(n)}$$, neuron *l* is hidden

$$\xi(n) = \frac{1}{2}\sum_{l \in C} e_m^2(n)$$, neuron *m* is an output node (17)

Differentiating Eq. (17) with respect to functional signal,

$$\frac{\partial \xi(n)}{\partial y_l(n)} = \sum_m e_m \frac{\partial e_m(n)}{\partial y_l(n)} \tag{18}$$

Using the chain rule rewriting the Eq. (18),

$$\frac{\partial \xi(n)}{\partial y_l(n)} = \sum_m e_m \frac{\partial e_m(n)}{\partial v_m(n)} \frac{\partial v_m(n)}{\partial y_l(n)} \tag{19}$$

From Figure. 6,

$$e_m(n) = d_m(n) - y_m(n) \tag{20}$$

$$= d_m(n) - \varphi_m(v_m(n))$$, neuron *m* is an output node

$$\frac{\partial e_m(n)}{\partial v_m(n)} = -\varphi'_m(v_m(n)) \tag{21}$$

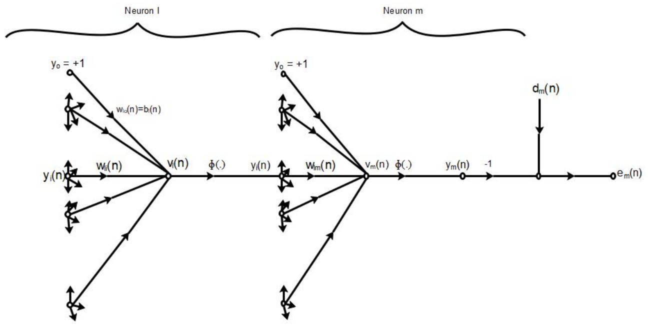

Figure 6: Output neuron k connected to hidden neuron j signal flow graph.

The induced local field for neuron *m*,

$$v_m(n) = \sum_{l=0}^{p} w_{ml}(n) y_l(n) \tag{22}$$

Differentiating Eq. (22) with respect to $y_l(n)$,

$$\frac{\partial v_m(n)}{\partial y_l(n)} = w_{ml}(n) \tag{23}$$

Using Eq. (21) and (22) in (19),

$$\frac{\partial \xi(n)}{\partial y_l(n)} = -\sum_m e_m(n) \varphi'_m(v_m(n)) w_{ml}(n) \tag{24}$$

$$= -\sum_m \delta_m(n) w_{ml}(n)$$

Getting back propagation formula using Eq. (24) in (16),

$\delta_l(n) = \varphi_l'(v_l(n)) \sum_m \delta_m(n) w_{ml}(n)$, neuron j is hidden (25)

- If neuron *l* is an output node, $\delta_l(n)$ equals the product of the derivative $\varphi_l'(v_l(n))$ and the error signal $e_l(n)$, with the neuron j as in Eq. (15).
- If the neuronis a hidden node $\delta_l(n)$ equals the product of the associated derivative $\varphi_l'(v_l(n))$ and the weighted sum computed for the neuron in the output layer that is connected to neuron *l*, as in Eq. (25).

Results and Discussion

Back propagation (BP) algorithm was considered as the important ingredient in the short term forecasting. For the forecasting purpose, hourly data was collected from the Punjab electricity board for the various days that helped in the training of the technique as well as for forecasting purpose and also the weather data is taken from the IMD Pune. The dew point temperature, dry bulb temperature, and humidity are taken as input and the load data is taken as the output. To train the network data, the data are divided into three parts, i.e. Validation, Training, and Testing. The BP algorithm is used to train the network and implemented in MATLAB. The sample data for one day is given in Table A.1. The inputs dew point, dry bulb temperature, humidity and load are presented in Figure 7.

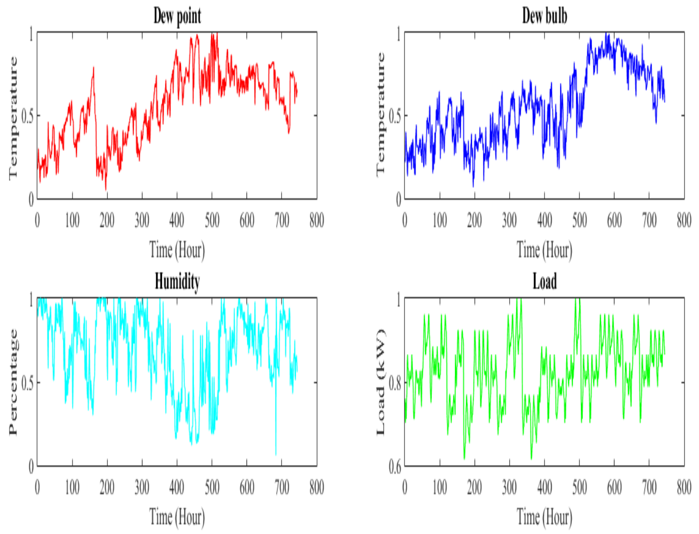

Figure 7: Representation of Load data

• **Dew Point Temperature:** It was basically a temperature value at which the air losses its control over the water vapor due to which some of the air molecules converted in to the water droplet and this particular temperature was lower than the temperature.

• **Dry Bulb Temperature:** whenever the thermometer was used for the measurement of temperature then it is called dry bulb temperature basically, it was atmospheric temperature read by the thermometer whenever it was exposed in the surrounding.

• **Humidity:** it was nothing but a water droplet which was present the atmosphere but they were completely invisible for the human was basically a water molecules in the gaseous state. As humidity increases, ability of body to resist the sweating capacity reduces due to reduction in the rate of evaporation of moisture from the body**.**

The network is trained for 5000 iterations with the presented data. In this, the training function is 'trainlm' and the activation function is sigmoid. It is observed that the MSE 0.0061, RMSE 0.0784, SSE 4.5838, MAPE 4.6759 and MAE 0.03081 after training it is observed that the actual load and forecasted load have variation or fluctuations as shown in Figure 8.

The short-term load forecasting is performed with the help of BP algorithm. Initially, the data for January month has been used in the normalized form. The updated values of weights have been obtained by training of network. The values of the initial and the final values of the weights between input and hidden layer are presented in Table 1. Table 2 presents the initial and final weights between hidden and an output layer. The size of neurons for input layer and hidden layer is selected as 3 and 21 respectively. The weights are initialized randomly.

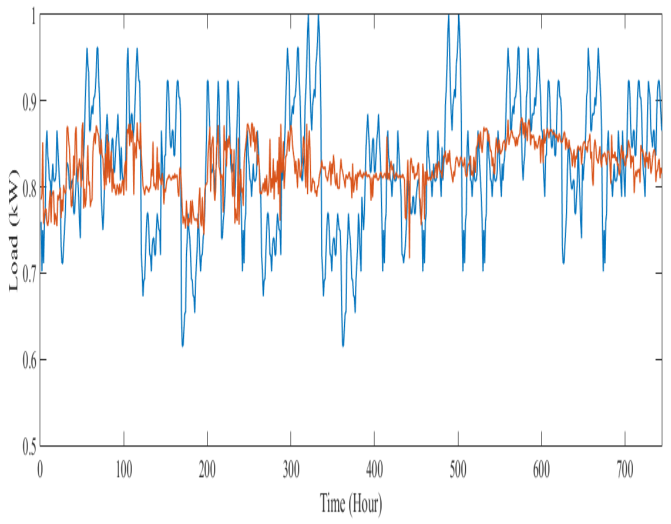

Figure 8: Representation of Actual and Forecasted Load

5. LEVENBERG MARQUARDT ALGORITHM

This depicts the well-ordered strategy for training the neural network to learn from the recent one month weather and temperature data. For outlining the network architecture the MATLAB ANN toolbox was used. The MATLAB ANN tool box is used for the Training, Testing and validation of the selected data. The selection of an optimum number of hidden layers is necessary as the increase in the number of hidden layers results in increased complexity of the ANN architecture, thereby, affecting its performance. The algorithm used for training the artificial neural network is Levenberg-Marquardt [12-13]. The Levenberg-Marquardt algorithm is better than Back Propagation algorithm because it has better convergence rate, the speed of iteration is more and it is more robust. The LM algorithm is very well suited for neural network training where the performance index is Mean Square Error. If initial guess is far from the mark, the LM algorithm can find an optimal solution. The main problem of Levenberg-Marquardt is a selection of the hidden layer size, which is selected by hit and trial method. In some cases the LM algorithm is slow to converge this particularly happens when the parameter is more than ten. It trains the network quicker as compared to the back propagation (BP) algorithm. Although it is more efficient, it requires more memory. The flow chart of Levenberg marquardt is shown in figure 9. The flow chart represents that how to train the network and test the network.

Table 2: Hidden layer to output weights

Input / Hidden	1	
	Initial weight	Final Weight
1	-0.40842	-1.46514
2	0.343397	0.752626
3	0.4595	1.455527
4	0.349397	-0.06603
5	-0.35115	-4.90491
6	-0.32571	-1.71731
7	-0.14839	-1.81978
8	0.181918	-0.47673
9	0.44912	3.267875
10	-0.36074	-1.58596
11	0.453228	0.562534
12	0.327723	-0.46041
13	-0.34338	-1.77566
14	0.078922	0.182285
15	-0.45145	1.054586
16	0.493923	7.432081
17	-0.2786	-1.76668
18	0.404807	3.120187
19	0.157243	-1.12301
20	-0.10462	-1.18771
21	-0.08841	6.37731

Table 1: Input to hidden layer weights

Input / Hidden	1		2		3		4	
	Initial weight	Final Weight	Initial weight	Final Weight	Initial weight	Final Weight	Initial weight	Final Weight
1	-0.34782	-1.80347	-0.43868	0.47153	0.405632	-0.02584	0.2499	1.263513
2	0.123935	0.289727	-0.31757	1.177436	0.200417	1.60001	-0.10008	-0.99536
3	-0.22342	-0.47633	-0.22485	0.212142	0.032243	0.102828	0.199506	0.149904
4	-0.48403	-4.87617	0.453785	9.75576	0.119968	15.20919	-0.41221	-6.54275
5	0.473032	0.825447	-0.34053	-1.09561	-0.38254	-0.31897	0.163085	-0.21496
6	-0.38572	-0.4897	-0.14099	-1.72231	-0.36486	-0.70368	-0.11197	0.995499
7	-0.27127	-0.22165	-0.16856	-0.00151	-0.08398	0.184493	-0.06536	0.093757
8	-0.38854	-2.30037	0.321911	0.745717	0.21104	-2.28622	0.294604	4.21837
9	0.473491	1.200004	-0.34527	-0.32894	0.210566	0.439223	-0.1994	-0.2442
10	-0.34622	-0.58212	0.030134	1.072627	0.109414	0.507845	-0.11479	0.000284
11	0.194902	0.107761	-0.11681	0.010768	0.25956	0.121974	0.378944	0.306346
12	-0.37715	-0.0205	0.441032	1.010085	0.084463	0.80182	0.404168	1.499915
13	0.218026	0.175655	0.451569	0.89646	-0.10817	-0.24252	0.285362	-0.13567
14	0.407962	0.463122	0.494323	1.611107	0.493919	0.628435	-0.19042	-0.81396
15	0.238622	-5.49304	-0.12286	9.250211	0.106339	9.479236	-0.4836	-2.02213
16	-0.26786	-0.81224	-0.11897	-1.98228	0.427006	0.895768	-0.33477	0.684218
17	-0.28366	-0.17115	0.169387	-0.85874	0.376344	5.69967	-0.24377	-0.81127
18	0.216537	0.29322	-0.37409	-0.74481	-0.41176	-0.17854	-0.02559	0.147388
19	0.287573	0.726257	-0.08337	-0.13256	-0.21717	0.189039	-0.11786	0.034249
20	-0.09315	4.299436	-0.37366	-5.59736	-0.44577	-4.29019	-0.46507	-1.38879
21	0.309573	-0.02732	0.008597	-1.77245	-0.34976	-0.89054	0.494925	0.765769

Results and Discussion

The results of single layer and multilayer network using the Levenberg-Marquardt technique are presented and discussed. The data set is divided into three parts, i.e. validation, training, and testing. The output considered is on hourly basis for the month of January.

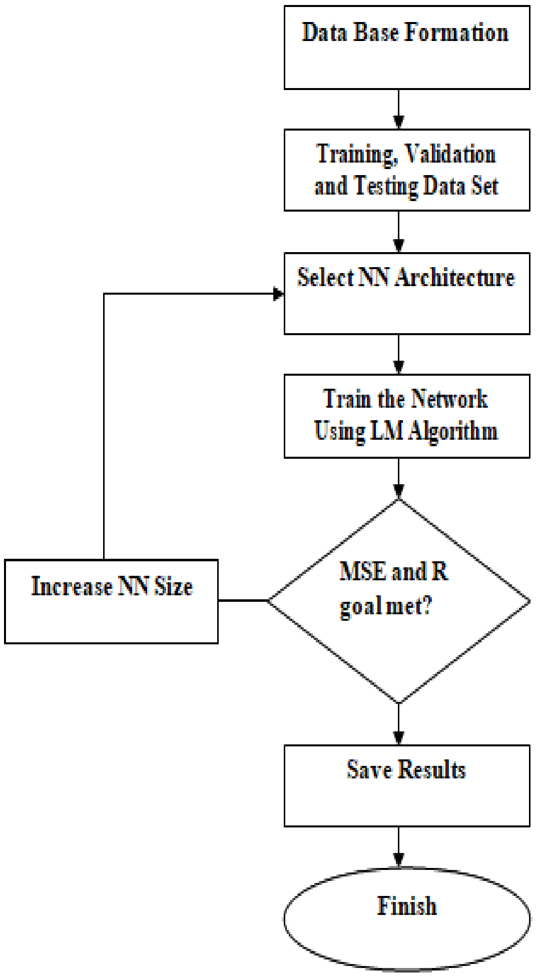

Figure 9 Levenberg Marquardt Flow chart

- **Single Layer Feed-Forward Network:**

In case of single layer network, there was single input layer as well as single output layer. Further, the neurons present in the input layer received the signals at the input terminal whereas the neurons present in the output layer received the output signal in a similar way. The input cells were connected to the similar output cell by the utilization of synaptic link carrying weight with it. Due to which this was considered as the feed forward neural network as the inverse operation cannot be possible in this network. Despite of the fact that the network has two layers still it was considering as a single layer due to single output layer receiving signal from input layer [17]. The data is forecasted with different sizes of the hidden layer and the best results are observed when hidden neuron size is 8 as shown in Table 3. The Figure 10 represents the actual load and forecasted load in the hidden neuron size of eight.

- **Multi Layer Feed-Forward Network:**

As its name indicates is formed from multilayers. So architectures of multilayer feed-forward network possessing an auxiliary layer considering between the input layer and the output layer. The hidden neurons present within the middle layer was considering for the computational purpose. The major importance of hidden layer as the computational work performed by the layer before the input signal received by the output terminal [17]. The input hidden layer weight was basically, a synaptic weight links formed by the combination of input neurons and the hidden neurons. In a similar way, whenever the output neurons formed a combination with the hidden layer neurons it was considered as the hidden output layer weight. The Table 4 shows that while changing the neuron size of the hidden layer the error is also changed and the Figure 11 shows the actual load and forecasted a load of the hidden layer having less error.

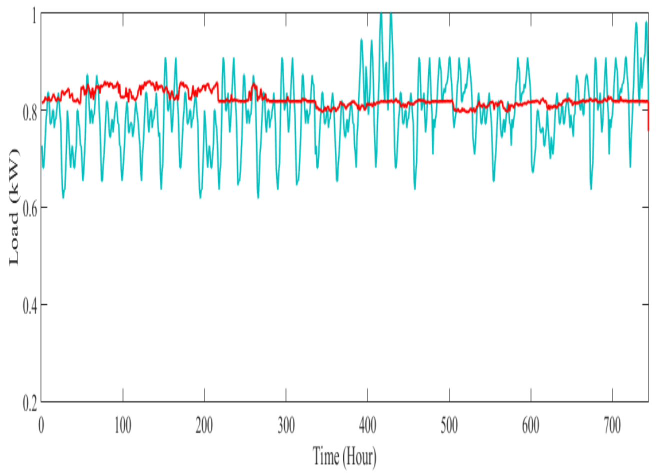

Figure 10 represents the actual load and forecasted a load of hidden neuron size eight.

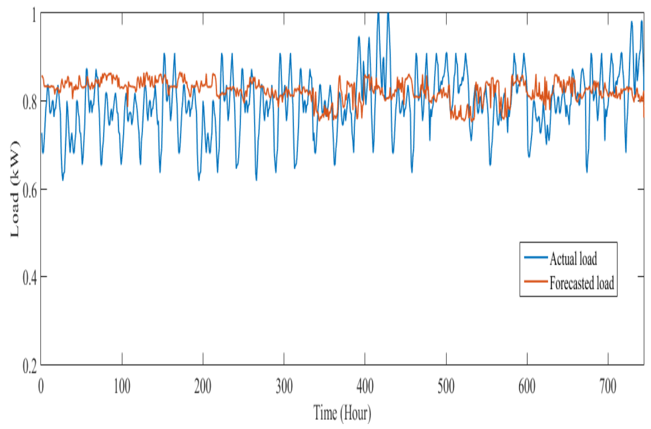

Figure 11 represents the actual load and forecasted load of the hidden neuron size six and nine.

Table 3 represents the different sizes of the hidden neuron and error.

Hidden Neuron size	3	6	9	12	15	18	21
MSE	0.0063	0.0071	0.0075	0.0069	0.0071	0.0073	0.0072
RMSE	0.0796	0.0843	0.0866	0.0834	0.0843	0.0857	0.0852
SSE	4.7154	5.2939	5.5845	5.1836	5.2872	5.4720	5.4042
MAPE	4.6491	4.4284	5.1941	5.2852	4.7721	5.1128	5.1123
MAE	0.0300	0.02819	0.0342	0.0349	0.0309	0.0335	0.0336

Table 4 represents the different size of the neuron and error in multilayerfeed-forward network.

Hidden layer 1	3	6	9	12	15	18	21
Hidden layer 2	6	9	12	15	18	21	24
MSE	0.0070	0.0067	0.0071	0.0069	0.0078	0.0072	0.0075
RMSE	0.0835	0.0819	0.0843	0.0831	0.0883	0.0849	0.0866
SSE	5.1828	5.0044	5.2870	5.1616	5.7679	5.3903	5.5582
MAPE	5.0423	4.6761	4.7905	4.7070	5.8124	5.1323	4.5431
MAE	0.0331	0.0303	0.0311	0.0305	0.0392	0.0339	0.0291

CONCLUSION

The error is calculated with the help of the input and output data, in this paper. In this work, load forecasting is done with the help of neural network tool box. The error of the BhaiRoopa feeder was obtained. The weather data is taken from the IMD pune and load data is taken from one of the 66kV substations of PSPCL situated in BhaiRoopa. The most widely used technique in ANN is BP algorithm and LM algorithm. The forecasting has been evaluated on the basis of calculating MSE, RMSE, SSE, MAE and MAPE between the actual value and forecasted value. Three inputs namely Dew point temperature, Dry bulb temperature and Humidity have been taken as input. The effect of change in number of hidden neurons and number of hidden layers is also studied. The observations made are the BP algorithm results into lower error compared to LM algorithm for same input. One hidden layer is sufficient for the formulation of Load forecasting problem; the increase in hidden neurons increases the error. From the error so observed, it can be safely concluded that the proposed work gives fairly accurate results. Electrical load forecasting helps to reduce the generation cost, spinning reserve capacity and increase the reliability of the power system. The unit commitment, economic allotment of generation is an important application of short term electrical load forecasting.

ACKNOWLEDGEMENT

Thapar Institute of Engineering and Technology Patiala, India has well equipped laboratories for effective academic and research work. The authors are really grateful to the Thapar Institute of Engineering and Technology for providing the necessary support and facilities to carry out the work presented in this manuscript.

APPENDIX

A.1. Sample of data of the network for one day.

Table A.1. Sample of data

Date	Hours	Dew Point	Dry Bulb	Humidity	Load
1/1/2015	1	6.5	7.4	94	80
1/1/2015	2	7.7	9.4	89	75
1/1/2015	3	11.9	12.6	95	75
1/1/2015	4	10.2	10.2	100	78
1/1/2015	5	8.6	9	97	82
1/1/2015	6	8.2	9	95	86
1/1/2015	7	4.9	5.4	97	88
1/1/2015	8	4	4	100	92
1/1/2015	9	8.4	8.4	100	92
1/1/2015	10	7.2	7.2	100	89
1/1/2015	11	7.1	8	94	85
1/1/2015	12	6.5	8.2	89	85
1/1/2015	13	7.8	7.8	100	86
1/1/2015	14	9.2	10.4	92	88
1/1/2015	15	6.5	7.8	91	88
1/1/2015	16	7.8	9.8	87	84
1/1/2015	17	8.2	8.2	100	86
1/1/2015	18	9.6	10	97	86
1/1/2015	19	5.2	5.2	100	88
1/1/2015	20	10	10	100	90
1/1/2015	21	7.1	8.4	91	92
1/1/2015	22	8.6	9.8	92	90
1/1/2015	23	10.6	11	97	87
1/1/2015	24	9.6	10.8	92	85

REFERENCES

[1] Asbury, C. E. **"Weather load model for electric demand and energy forecasting"**. IEEE Transactions on Power Apparatus and Systems, vol. 94(4), pp. 1111-1116 (1975). https://doi.org/10.1109/T-PAS.1975.31945

[2] Hubele, N. F., & Cheng, C. S. **“Identification of seasonal short-term load forecasting models using statistical decision functions”**. IEEE Transactions on Power Systems, vol. 5(1), pp. 40-45 (1990). https://doi.org/10.1109/59.49084

[3] Chheepa, T. K., &Manglani, T. **“A Critical Review on Employed Techniques for Short Term Load Forecasting”**. International Research Journal of Engineering and Technology, vol. 4(6) (2017).

[4] Bolzern, P., & Fronza, G. **“Role of weather inputs in short-term forecasting of electric load”**. International Journal of Electrical Power & Energy Systems, vol. 8(1), pp. 42-46 (1986). https://doi.org/10.1016/0142-0615(86)90024-4

[5] Yang, Y., Wu, J., Chen, Y., & Li, C. **“A new strategy for short-term load forecasting”**. In Abstract and Applied Analysis.Hindawi (2013). https://doi.org/10.1155/2013/208964

[6] Liu, N., Babushkin, V., & Afshari, A. “**Short-term forecasting of temperature driven electricity load using time series and neural network model”**. Journal of Clean Energy Technologies, vol. 2(4), pp. 327-331 (2014). https://doi.org/10.7763/JOCET.2014.V2.149

[7] Rahman, S., & Bhatnagar, R. **“An expert system based algorithm for short term load forecast”**. IEEE Transactions on Power Systems, vol. 3(2), pp. 392-399 (1988). https://doi.org/10.1109/59.192889

[8] Jabbour, K., Riveros, J. F. V., Landsbergen, D., & Meyer, W. **“ALFA: Automated load forecasting assistant”**. IEEE Transactions on Power Systems, vol. 3(3), pp. 908-914 (1988). https://doi.org/10.1109/59.14540

[9] Kuusisto, S., Lehtokangas, M., Saarinen, J., & Kaski, K. **“Short term electric load forecasting using a neural network with fuzzy hidden neurons”**. Neural Computing & Applications, vol. 6(1), pp. 42-56 (1997). https://doi.org/10.1007/BF01670151

[10] Ho, K. L., Hsu, Y. Y., Chen, C. F., Lee, T. E., Liang, C. C., Lai, T. S., & Chen, K. K. **“Short term load forecasting of Taiwan power system using a knowledge-based expert system”**. IEEE Transactions on Power Systems, vol. 5(4), pp. 1214-1221 (1990). https://doi.org/10.1109/59.99372

[11] Kiartzis, S. J., Bakirtzis, A. G., & Petridis, V. **“Short-term load forecasting using neural networks”**. Electric Power Systems Research, vol. 33(1), pp. 1-6 (1995). https://doi.org/10.1016/0378-7796(95)00920-D

[12] Singla, M., **“Load Forecasting Using Artificial Neural Network”** (Doctoral dissertation) (2018).

[13] Singla, M. K., & Gupta, J. **“Load Forecasting Using Back Propagation Algorithm”**. International Journal of Engineering and Techniques, vol. 4, pp. 169-175 (2018).

[14] Singla, M. K., & Hans, S. **“Load Forecasting using Fuzzy Logic Tool Box”**. Global Research and Development Journal for Engineering, vol. 38, pp. 12-19 (2018).

[15] Mujeeb, R. **“Introduction to artificial neural network and machine learning”**. Palakkad: Government engineering college, sreekrishnapuram (2012).

[16] Zurada, J. M. **“Introduction to artificial neural systems”.** St. Paul: West publishing company vol. 8 (1992).

[17] Rajashekaran, S., and G. A. Vijayalksmi. **“Neural networks, fuzzy logic and genetic algorithms”**, (2004).