Comparative Study of Short Term Load Forecasting Using Multilayer Feed Forward Neural Network With Back Propagation Learning and Radial Basis Functional Neural Network

Amit Tiwari*[1], Adarsh Dhar Dubey[2] and Devesh Patel[3]

ABSTRACT

The term load forecast refers to the projected load requirement using systematic process of defining load in sufficient quantitative detail so that important power system expansion decisions can be made. Load forecasting is necessary for economic generation of power, economic allocation between plants (unit commitment scheduling), maintenance scheduling & for system security such as peak load shaving by power interchange with interconnected utilities. With structural changes to electricity in recent years, there is an emphasis on Short Term Load Forecasting (STLF).STLF is the essential part of power system planning & operation. Basic operating functions such as unit commitment, economic dispatch, and fuel scheduling & unit maintenance can be performed efficiently with an accurate forecast. Short term load forecasting can help to estimate load flows & to make decisions that can prevent overloading. Timely implementations of such decisions lead to improvement of network reliability & to the reduced occurrences of equipment failures & blackouts.

The aim of short term load forecasting is to predict future electricity demands based, traditionally on historical data and predicted weather conditions. Short term load forecasting in its basic form is a statistical problem, where in the previous load values (time series variables) and influencing factors (casual variables) are used to determine the future loads.

1. INTRODUCTION

Load forecasting is an important part of energy management systems. Load forecasting refers to projected load requirement using systematic process of defining load in sufficient quantative detail so that important power system decisions can be made. a total forecast is obtained by combining forecasts for various classes of customers such as residential commercial, industrial and others. Depending on the time zone of planning strategies the load forecasting can be divided into following four categories:

1. Very short time load forecast: These are required for on line operation and control of system. The time ranges from a few second to a minute. Generally these forecasts are more accurate due to the fact that the lead time is very small.
2. Short time load forecast: Short term load forecast have special importance in operation, control and techno commercial decisions. It is made on hourly basis for lead time of an hour to a week ahead.
3. Midterm load forecast: It ranges from a week ahead to six months. Generally energy estimates are made in this range. Energy sales and fuel purchase agreements are based on these forecasts.

1.* Amit Tiwari, Assistant Professor-I, Electrical & Electronics Engineering Department, School of Management Sciences,Techanical Campus, Lucknow, India. e-mail : amit11823@gmail.com

2. Adarsh Dhar Dubey, Assistant Professor-I, Electrical & Electronics Engineering Department, School of Management Sciences, Techanical Campus, Lucknow, India. e-mail : add0252@rediffmail.com

3. Devesh Patel, Assistant Professor-I, Electrical & Electronics Engineering Department, School of Management Sciences,Techanical Campus, Lucknow, India. e-mail : devesh.mpec@gmail.com

 S-JPSET : Vol. 7, Issue 1, ISSN : 2229-7111 (Print) & ISSN : 2454-5767 (Online)

4. Long term load forecast: Long term load forecast ranges from a year to 20 years. Theses in general are estimate of peak demand in future years. It is very much needed in system generation and transmission planning.

Short term load forecasting plays an important role in power systems. Accurate short term load forecasting has a significant influence on proper system operational efficiency such as unit commitment, annual hydro thermal maintenance scheduling, hydro thermal coordination, demand side management, interchange evaluation, security assessment and other purposes.. Improvements in the accuracy of short term load forecasts can result in significant financial savings for utilities and co generators. Short term load forecasting can help to estimate load flows and to make decisions that can prevent overloading. Timely implementations of such decisions lead to the improvement of network reliability and to reduced occurrences of equipment failures and blackouts**.**

2. MULTILAYER FEED FORWARD NETWORK WITH BACK PROPAGATION LEARNING

A neural network consists of a number of neurons that are connected in massively parallel architecture. Just as there are many inputs (stimulation levels) to a neuron, there should be many input signals to our processing element. All of them should come into PE simultaneously. In response, a neuron either fires or does not fire depending on some threshold level. Each input will be given a relative weighting, which will affect the impact of the input. This is something like the varying synaptic strengths of the biological neurons. Weights are adaptive coefficients within the network that determine the intensity of the input signal. All of the products will be summed and compared to some threshold to determine the output. If the sum of the inputs is greater than threshold value, the processing element generates a signal. If the sum of the inputs is less than the threshold value, no signal is generated.

A class of neural networks that overcomes limitations of single layer linear networks and is able to model non linear relationships between the inputs and the outputs is the Multilayered Feed forward Network with learning carried out using the Back Propagation Rule. This network consists of a set of input units, a set of output units and one or more layers of intermediate units. These intermediate unit layers are sometimes referred to as a hidden unit layers since the unit in them do not directly communicate with the environment.

In the multilayer network, the first set of neurons connecting to the inputs serve only as distribution points. They perform no input summation. The training instance set for the network must be presented many times in order for the interconnection weights between the neurons to settle into a state for correct classification of input patterns. While the network can recognize patterns similar to those they have learned, they do not have the ability to recognize new patterns. This is true for all supervised learning networks. In order to recognize new patterns, the network needs to be retrained with these patterns along with previously known patterns. If only new patterns are provided for retraining, then old patterns may be forgotten. In this way learning is not incremental over time. This is a major limitation for supervised learning networks. Another limitation is that the back propagation network is prone to local minima, just like any other gradient descent algorithm.

A Multilayer perceptron is able to model nonlinear relationships between the inputs and outputs. This network consists of a set of 'n' input units, a set of 'm' output units and one or more layers of 'N' intermediate units. These intermediate unit layers are called hidden unit layers. Hidden units $v_j, j = 1,.....,N$ and output units $y_i, i = 1, 2,.....,m$ process their input as the case of the single layer net. Thus

S-JPSET : Vol. 7, Issue 1, ISSN : 2229-7111 (Print) & ISSN : 2454-5767 (Online)

$$v_j = g(\sum_{k=1}^{n} w_{jk} x_k).and.y_i = g(\sum_{j=1}^{n} w_{ij} v_j) \quad .....(2.1)$$

With j=1,……,N and i=1,……., m

The learning rule is a generalization of the delta rule for multi- layer networks. It carries out a minimization of the mean square error E which is now a function of both weight matrices w and W

$$E(w,W) = \sum_{i=1}^{n} (y_i^{\mu}(w,W) - y_{i\ target}^{\mu})^2 \quad .....(2.2)$$

The algorithm carries out a steepest descent correction on the matrix giving Δw and ΔW. Thus

$$\Delta W_{ij} = -\eta \frac{dE}{\partial W_{ij}} and.\Delta w_{jk} = -\eta \frac{dE}{\partial w_{jk}} \quad .....(2.3)$$

Assuming both the hidden and output units to have sigmoid function $g(h) = \frac{1}{1+\exp(-\beta h)}$ as then activation function, the expression for ΔW_{ij} and Δw_{jk} can be derived as

$$\Delta W_{ij} = \eta \Delta_i^{\mu} v_j^{\mu} with.\Delta_i^{\mu} = y_i^{\mu}(1-y_i^{\mu})[y_i^{\mu} - y_{i\ target}^{\mu}]^2$$

$$\Delta w_{jk} = \eta \delta_j^{\mu} x_k^{\mu}.With.\delta_j^{\mu} = v_j^{\mu}(1-v_j^{\mu}) \sum_{i=1}^{n} \Delta_i^{\mu} W_{ij}$$

.....(2.4)

Since the weight errors Δ_i^{μ} and δ_j^{μ} are successively back propagated from output layer to hidden layer, this algorithm is known as 'Error Back – Propagation'. The main difficulties with these networks are that they sometimes get stuck into local minimum and also their convergence is slow. The slow convergence can be overcome partially by using momentum terms. To overcome both the problems a second order optimization technique based on Newton's method called Levenberg – Marquardt rule has been used. The L-M update rule is –

$$\Delta W = (J^T J + \mu I)^{-1} J^T e \quad .....(2.5)$$

Where J is the jacobian matrix of derivatives of each error to each weight, 'μ' is a scalar and 'e' is the error vector. If the scalar 'μ' is very large the above expression approximates gradient descent, while if it is small the above expression becomes the Gauss – Newton method. The Gauss – Newton method is faster and more accurate near an error minimum so the aim is to shift towards the G-N method as quickly as possible. Thus, μ is decreased after each successful step and increased only when a step increases the error.

2.1 FFNN Implementation For A Gujrat Utility

The performance is studied on the historical data of a Gujrat State utility. This data includes hourly loads and temperatures of two (T1 and T2) major load centers. The data of three weeks were taken for fixing the parameters of the various models (historical data). The forecaster was used to forecast the hourly load up to a week. The models were designed to give the optimum performance. The parameters of the models were finalized after the several trial and error efforts to give the optimum performance. Though each of these models have different numbers of input variables but the error performance was best for the individual method of forecasting. The time series formulation had T1 and T2 at the forecast hour as exogenous variables, whereas, load at hour k-1 and k-2 was taken as the time series inputs, where, k is the hour of forecast. There were separate models for forecasting weekdays and weekend days. The FFNN had 35 input variables containing loads at k-1, k-2, k-3, k-25, k-26, k-27, k-168, k-169, k-170 as load variables, and temperature of both places at k, k-1, k-2, k-3 along with 07 inputs for day type. The networks with 10 hidden neurons were selected on basis of trial and error. For the Gujrat utility system, network architecture with 35 input nodes, 10 neurons and a single output is selected after trial and error approach

with different sets of inputs and number of neurons. The input set consisted of following:

1. Day type: 07 input. For example, 1000000 for Monday, 0100000 for Tuesday and so on.
2. Hour type: The 24 hour of the day was coded into 05 bit binary number.
3. Load: 09 inputs were used. They are 03 hour prior load to the hour to be forecasted i.e. L(k-1), L(k-2), L(k-3), where k is the instant at which the load is to be forecasted. Other similar inputs are L(k-25), L(k-26), L(k-27), L(k-168), L(k-169), L(k-170).
4. Temperature: 14 inputs were used. These were $T_1(k)$, $T_1(k–1)$, $T_1(k–2)$, $T_1(k–3)$ for Jamnagar city hourly temperature and $T_2(k)$, $T_2(k–1)$, $T_2(k–2)$, $T_2(k–3)$ for Bhuj city hourly temperature. Apart from these temperatures of the day, previous day and day a week before for both the places were taken as inputs.

Single output node is taken which provides the forecasted load at a particular hour. Network with 10 hidden layer nodes were found sufficient for this problem. The learning rate selected was .001

3. RADIAL BASIS FUNCTION NETWORK

A Radial Basis Function (RBF) Network consists of two layers, a hidden layer with nonlinear neurons and an output layer with linear neurons. Thus the transformation from the input space to the hidden unit space is non-linear whereas the transformation from the hidden unit space to the output space is linear. The basis functions in the hidden layer produce a localized response to the input i.e. each hidden unit has a localized receptive field. The basis function can be viewed as the activation function in the hidden layer. The network itself is used to select its input variables and parameters. The network has a characteristic of convergence to the lowest possible training error for given set of network parameters and input variables. The advantage of this network lies in selection of input variables on the basis of network performance and this selection includes the load time series and weather variables. The training of the network is considerably fast and does not need monitoring of training process for non-convergence and parameter tuning, during design and testing.

The basic model of a RBF network is shown in Fig.2.1. It is different from a feed forward network. A radial basis neuron receives the vector distance between its weight vector (cluster center) 'w' and the input vector multiplied by the bias ' b_g ' unlike the sum of product of inputs and respective synaptic weights in case of feed forward network.

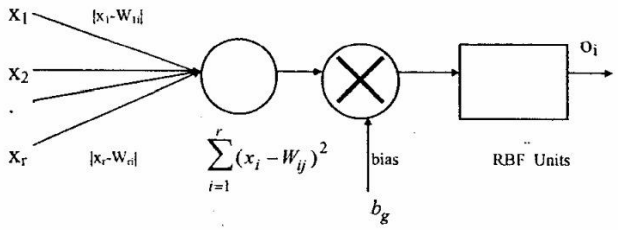

Fig 2.1: Radial Basis Function Network

The RBF unit, transfer function in case of perceptron, is similar to Gaussian density function, which is defined by center position and the bias. The output of the R B F unit is given by.

$$o_i = \exp-(\sum_{i=1}^{r}(x_{ip} - W_{ij})^2 b_g) \qquad \text{.....(3.1)}$$

Where

W_{ij} = Center of ith R B F unit for input variable j

b_g = Bias of the R B F unit

X_{jp} =jth variable of the input pattern

The most common basis function chosen is a Gaussian function, in which case

The activation level Oj of hidden unit j is calculated by

$$o_j = \exp[\frac{-(X - W_j).(X - W_j)}{2\sigma_j^2}] \qquad \text{.....(3.2)}$$

Where X is the input vector, W_j is weight vector associated with hidden unit j (i.e., the center of its Gaussian function), and σ^2 is the normalization factor. The outputs of the hidden unit lie between 0 and 1; the closer the input to the center of the Gaussian, the larger the response of the node. Because the node produces an identical output for inputs with equal distance from the center o f the Gaussian, it is called a Radial basis.

The activation level O_j of an output unit is determined by

$$O_j = \Sigma\ W_{ji}\, O_{I+} \qquad \text{.....(3.3)}$$

Where Wji is the weight from hidden unit i to output unit j . The output units form a linear combination of-the nonlinear basis-functions, and thus the overall network performs a nonlinear transformation of the input.

The normalization factor represents a measure o f the spread of the data in the cluster associated with the hidden unit. It is commonly determined by the average distance between the cluster center and the training instances in that cluster i.e. for hidden unit j.

$$\sigma_j^2 = \frac{1}{M}\sum (X - W_j)(X - W_j) \qquad \text{.....(3.4)}$$

where X is a training pattern in the cluster, Wj is the center o f the cluster associated with hidden unit j, and M is the number of training instances in that cluster.

Such a network maps f: $R^n \rightarrow R$ according to

$$f_r\ (\overline{x}) = \lambda_0 + \sum_{i=1}^{n_r} \lambda_i \phi\ \left(\left\| \overline{x} - \overline{c_i} \right\| \right) \qquad \text{.....(3.5)}$$

where $\overline{x} \in R^n$ is the input vector, Φ (.) is a non-linear function, ||. || denotes the Euclidean norm, $\lambda_i, 0 \le i \le n_r$ are the weights, $c_i \in R^n, 1 \le i \le n_r$, are known as the RBF centers.

A common choice of the non-linearity is the Gaussian function Φ (v) = exp ($-v^2$), Designing an RBF network involves choosing the centers $\overline{c_i}$ from the data points $\overline{x}$. For this Orthogonal Least Square (OLS) Algorithm is used.

This method is based on linear regression models according to which a desired response(n) is defined as

$$d(n) = \sum_{i=1}^{M} x(n)a + e(n), n = 1,2,.....,N \qquad \text{.....(3.6)}$$

Where the a_i, are the model parameters, the x_i(n) are the regressors and e(n) is the residue (error). Using matrix notation we can write

$$\overline{d} = \overline{X}\overline{a} + \overline{e} \qquad \text{.....(3.7)}$$

Where

$$\overline{d} = [d(1), d(2),.....d(N)]^T$$

$$\overline{a} = [a_1, a_2,.....a_M]^T$$

$$\overline{X} = [\overline{X_1}, \overline{X_2}.....\overline{X_M}]$$

$$x_i = [\overline{x_i}(1), \overline{x_i}(2).....\overline{x_i}(N)]^T, 1 \le i \le M$$

$$\overline{e} = [e(1), e(2),.....e(N)]^T$$

The OLS method involves the transformation o f the $\overline{X}_1, \overline{X}_2, \overline{X_M}$ into a corresponding set of orthogonal basis vectors denoted by $\overline{u_1}, \overline{u}_2 ... \overline{u_M}$ using the transformation

$$\overline{u_1} = \overline{x_1}$$

$$d_{ik} = \frac{\overline{u_i^T}\,\overline{x_k}}{\overline{u_i^T}\,\overline{u_k}}, ...1 \le i \le k$$

$$\overline{u_k} = \overline{x_k} - \sum_{i=1}^{k-1} \alpha_{ik}\, \overline{x_i} \;.....where...k = 2,.....M$$

.....(3.8)

Thus the OLS learning method chooses the RBF centers $\overline{c_1}, \overline{c_2}, ... \overline{c_M}$ as a subset of the training data vectors $\overline{X}_1, \overline{X}_2, \overline{.....X_M}$ where M <N . The centers are determined one at a time until a network of adequate performance is constructed. At each step

 S-JPSET : Vol. 7, Issue 1, ISSN : 2229-7111 (Print) & ISSN : 2454-5767 (Online)

of the procedure, the increment to the explained variance of the desired response is maximized. This way the OLS method results in a RBF network with less number of hidden neurons than that of an RBF network with randomly selected centers.

3.1 RBFNN Implementation for a Gujrat Utility

The numerical simulation was performed on load data of state of Gujrat. The hourly temperature data of two cities, Jamnagar and Bhuj were used in the simulation. The training set consists of three week historical data prior to the forecast hour and a week was taken as the test pattern. To arrive at an appropriate network single, weekly and hourly architectures were tried. The hourly structure was found to be most promising. The load curve also indicates strong dependence of hourly load. Separate network is selected for each hour. The weekdays and weekend days load characteristics are very much different; therefore, separate hourly structures for weekend days and weekdays are taken. The holiday load is forecasted using the weekend day structure. Thus there are basically two types of structures each o f twenty-four networks. As far as the size o f training sample is concerned the ratio of data used for training to that o f forecast remains same. We observed in our study that in short term forecasts the nearness o f data used for training to that to be forecasted is an important factor.

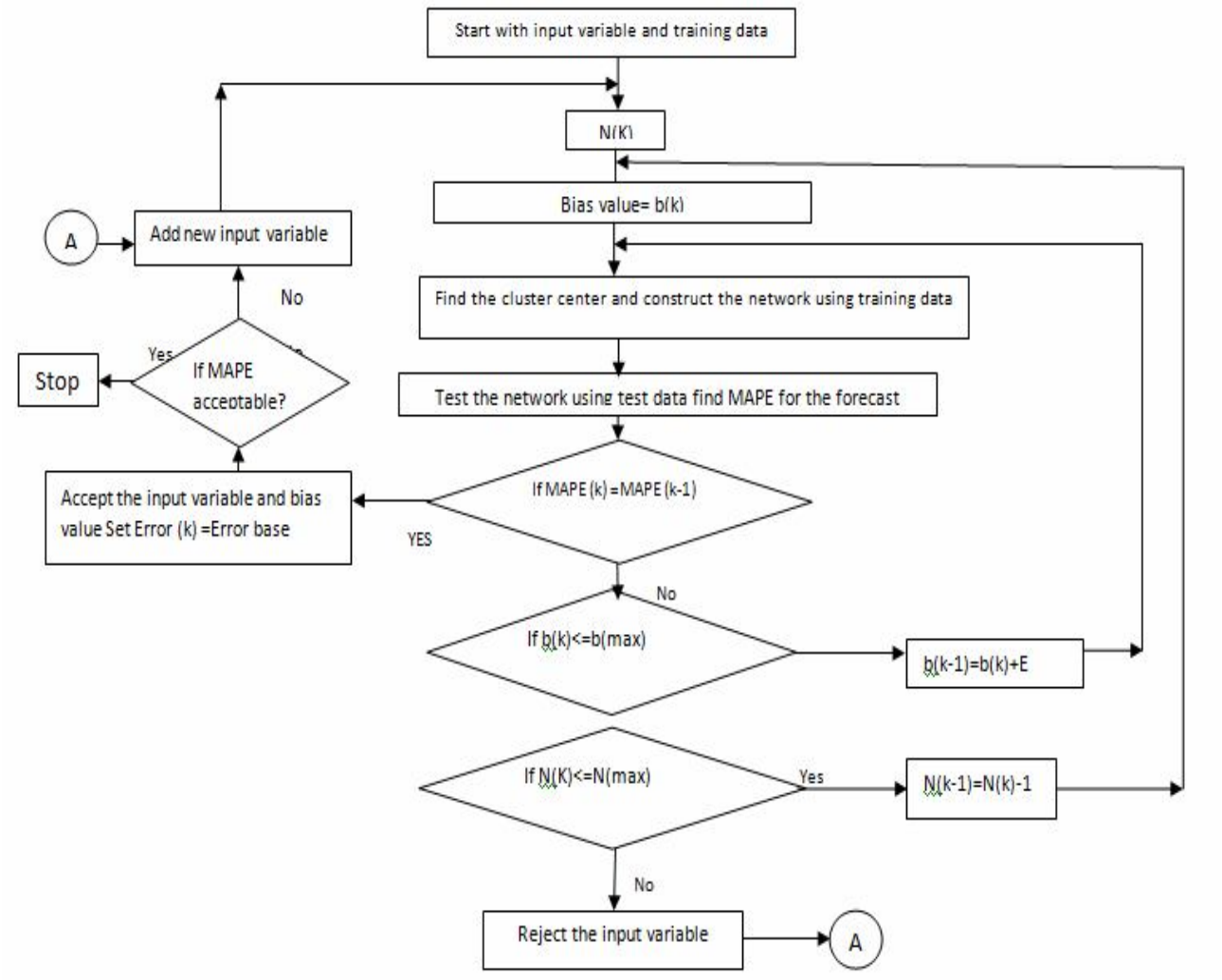

Fig3.1: Flow Chart for Feature and Architecture Selection

S-JPSET : Vol. 7, Issue 1, ISSN : 2229-7111 (Print) & ISSN : 2454-5767 (Online)

The input selection was made starting with first three hour load values prior to the hour to be forecasted i.e., $L_{(K-1)}$, $L_{(K-2)}$, $L_{(K-3)}$, where k is the instant at which forecast is to be made. The first three hour loads were selected heuristically. The results and inference drawn for various inputs in selection procedure for feature and architecture selection is illustrated for the system in table 1 for weekend day. The process was terminated when no appreciable improvement in the architecture selection was found.

The input variables, which do not improve the network performance, may worsen the performance of the network either in terms of accuracy or the training time. This is an important observation. This is equally true for feed forward networks where the input selection is often performed on basis o f engineering judgment. It is obvious from table that only load series data is required set for proper forecasting for this particular system with hourly structure for weekend days. Thus, the selected input variables for weekend structure are $L_{(K-1)}$, $L_{(K-2)}$, $L_{(K-3)}$, $L_{(K-4)}$, $L_{(K-168)}$load series values.

A similar procedure is applied for weekday structure and the input variable selected by the method are $L_{(K-1)}$, $L_{(K-2)}$, $L_{(K-3)}$, $L_{(K-4)}$, $L_{(K-168)}$ load series values and $T_{(k)}$, $T_{(k-1)}$, $T_{(k-2)}$ temperature time series values of one placer It is worth noting that hourly temperature data of other place (T_2) was also used in the process but were rejected. Case was similar for the daily average temperature for the two cities. The final architecture selected from the tablet corresponds to the final base case i.e. six inputs and 3 RBF units with bias value of 2.374. The architecture parameters for the weekdays are found to be one with 9 inputs and 5 RBF unit search with bias value of 0.7928.

4. RESULTS AND DISCUSSIONS

4.1 Case Short Term Load Forecasting on Seasonal Transition Weeks for a Gujrat Utility:

COMPARATIVE PERFORMANCE:

Table-4.1 : Forecast Errors (MAPE) for a Selected Week in Winter (Dec. 17-23)

Winter (Dec. 17-23)		
DAY	**Forecast Errors (MAPE) by FFNN**	**Forecast Errors (MAPE) by RBFNN**
Monday	1.2432	1.1105
Tuesday	0.9937	1.1171
Wednesday	1.4074	1.0776
Thursday	1.6119	1.5838
Friday	1.8903	1.0727
Saturday	1.0621	1.6459
Sunday	1.1672	0.7494
Average	1.3393	1.1939

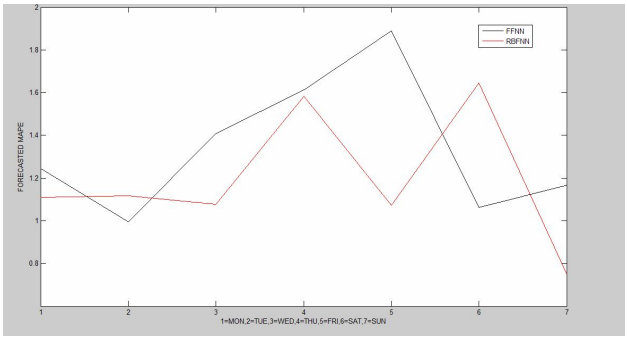

Fig 4.1 : Comparative Plot Showing Forecast Errors (MAPE) for a Selected Week in Winter (Dec.17-23)

Table-4.2 : Forecast Errors (MAPE) for a Selected Week in Spring (Mar. 25-31)

Spring (Mar. 25-31)		
DAY	**Forecast Errors (MAPE) by FFNN**	**Forecast Errors (MAPE) by RBFNN**
Monday	1.4455	0.9606
Tuesday	0.9916	1.1114
Wednesday	1.3930	1.0856
Thursday	2.4804	1.7412
Friday	1.2875	0.7082
Saturday	0.6662	0.7226
Sunday	0.7371	2.2876
Average	1.2854	1.2310

 S-JPSET : Vol. 7, Issue 1, ISSN : 2229-7111 (Print) & ISSN : 2454-5767 (Online)

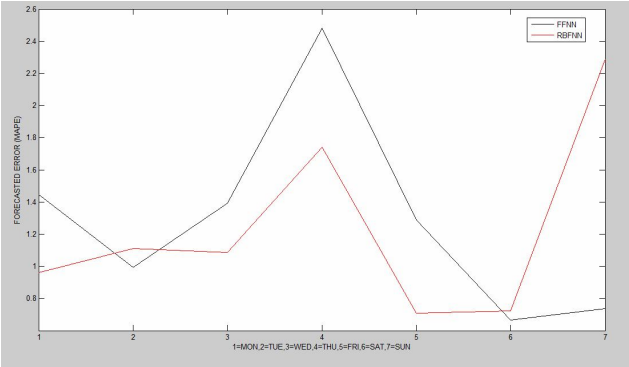

Fig 4.2 : Comparative Plot Showing Forecast Errors (MAPE) for a Selected Week in Spring (Mar. 25-31)

Table-4.3 : Forecast Errors (MAPE) for a Selected Week in Summer (June 19-25)

Summer (June 19-25)		
DAY	**Forecast Errors (MAPE) by FFNN**	**Forecast Errors (MAPE) by RBFNN**
Monday	0.9046	0.9608
Tuesday	1.0007	1.1732
Wednesday	1.3168	1.1359
Thursday	1.0421	1.4486
Friday	0.9248	0.9218
Saturday	0.7331	1.1496
Sunday	1.6191	1.6583
Average	1.0773	1.2068

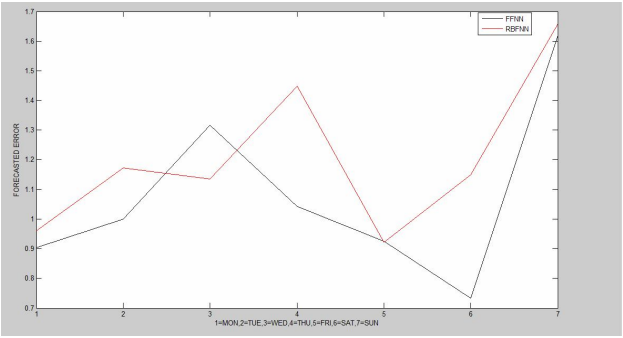

Fig 4.3 : Comparative Plot Showing Forecast Errors (MAPE) for a Selected Week in Summer (June 19-25)

4.2 Case: Short Term Load Forecasting on a Weekday (Tuesday Nov.22) and Holiday (Sunday Nov.27) for a Gujrat Utility:

Table - 4.4: Forecast Results with FFNN (Tuesday Nov.22)

Hour	Actual Load(MW)	Forecasted load(MW)	Absolute Error(MW)	Absolute % Error
1	2479	2572	93	3.75
2	2470	2563	93	3.76
3	2404	2530	126	5.24
4	2491	2534	43	1.73
5	2660	2509	151	5.68
6	2988	2738	250	8.37
7	3094	2835	251	8.37
8	3144	3007	137	4.36
9	3115	3140	25	0.8
10	3060	3108	48	1.57
11	2937	2867	70	2.38
12	2750	2947	197	7.16
13	2759	2868	109	3.95
14	2838	2896	58	2.04
15	2901	2921	20	0.69
16	2918	2929	11	0.38
17	2936	2983	47	1.6
18	3128	3011	117	3.74
19	3185	3039	146	4.58
20	3059	3085	26	0.85
21	2922	2998	76	2.6
22	2836	2871	35	1.23
23	2733	2838	105	3.84

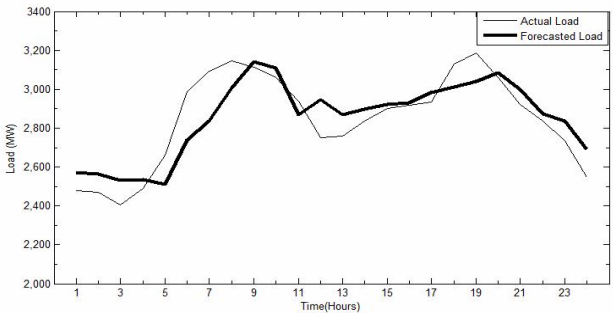

Fig 4.4: Forecast error with FFNN (Tuesday Nov.22)

Table - 4.5: Forecast Results with FFNN (Sunday Nov.27)

Hour	Actual Load(MW)	Forecasted load(MW)	Absolute Error(MW)	Absolute % Error
1	2479	2570	91	3.67
2	2470	2524	54	2.19
3	2404	2413	9	0.37
4	2491	2398	-93	3.73
5	2660	2783	123	4.62
6	2988	2884	104	3.48
7	3094	3002	92	2.97
8	3144	3095	49	1.56
9	3115	3086	29	0.93
10	3060	2916	144	4.7
11	2937	2898	39	1.33
12	2750	2688	62	2.25
13	2759	2690	69	2.5
14	2838	2808	30	1.06
15	2901	2798	103	3.55
16	2918	2886	32	1.1
17	2936	2886	50	1.7
18	3128	2998	130	0.96
19	3185	3086	99	3.11
20	3059	2973	86	2.81
21	2922	2878	44	1.5
22	2836	2754	82	2.89
23	2733	2671	62	2.27

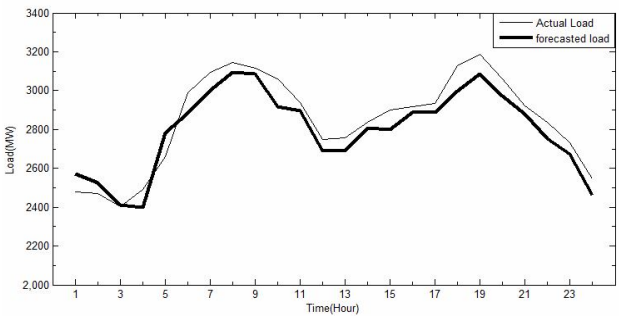

Fig 4.5: Forecast Error with FFNN (Sunday Nov.27)

Table - 4.6: Forecast results with RBFNN (Tuesday Nov.22)

Hour	Actual Load(MW)	Forecasted load(MW)	Absolute Error(MW)	Absolute % Error
1	2479	2536	57	2.3
2	2470	2549	79	3.2
3	2404	2534	120	5.41
4	2491	2520	29	1.16
5	2660	2518	142	5.34
6	2988	2650	338	11.31
7	3094	2884	210	6.79
8	3144	3036	108	3.44
9	3115	3041	74	2.38
10	3060	2997	63	2.06
11	2937	2975	38	1.29
12	2750	2925	175	6.36
13	2759	2874	115	4.17
14	2838	2848	10	0.35
15	2901	2915	14	0.48
16	2918	2935	17	0.58
17	2936	2949	13	0.44
18	3128	2986	142	4.54
19	3185	3067	118	3.7
20	3059	3064	5	0.16
21	2922	2977	55	1.88
22	2836	2892	56	1.97
23	2733	2801	68	2.49

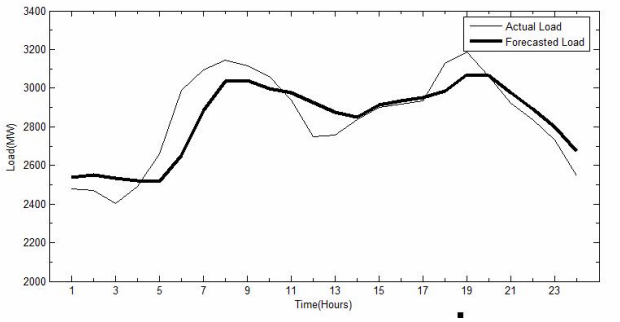

Fig 4.6: Forecast Error with RBFNN (Tuesday Nov.22)

Table - 4.7: Forecast Results with RBFNN (Sunday Nov.27)

Hour	Actual Load(MW)	Forecasted load(MW)	Absolute Error(MW)	Absolute % Error
1	2479	2531	52	2.11
2	2470	2542	72	2.9
3	2404	2481	77	3.17
4	2491	2520	29	1.17
5	2660	2549	111	4.15
6	2988	2678	310	10.37
7	3094	2922	172	5.36
8	3144	3038	106	3.37
9	3115	3042	73	2.33
10	3060	3061	1	0.05
11	2937	3039	102	3.47
12	2750	2843	93	3.37
13	2759	2797	38	1.36
14	2838	2878	40	1.42
15	2901	2916	15	0.53
16	2918	2902	16	0.55
17	2936	2954	18	0.61
18	3128	3057	71	2.27
19	3185	3096	89	2.81
20	3059	3072	13	0.42
21	2922	2951	29	1
22	2836	2877	41	1.43
23	2733	2755	22	0.81

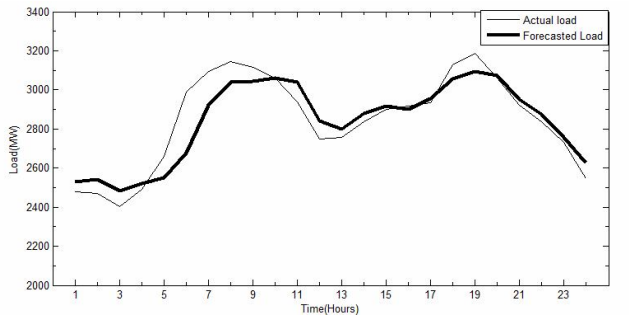

Figure - 4.7: Forecast Error with RBFNN (Sunday Nov.27)

5.CONCLUSION

The results of the system with these two networks reveal the following:

- RBF network predicts the hourly load more accurately than FFN network. For the given data, the average percentage error in forecasted hourly loads for RBF is lesser than FFN network.
- Both RBF & FFN networks are able to predict the load extremely fast.
- RBF takes considerably less CPU time for training time as compared to FFN network.

REFERENCES

[1] G. Gross and F.D. Galiana, "Short Term Load Forecasting" Proc. IEE, vol.75, no 12, Dec., 1987, pp.1558-1578

[2] I. Moghram and S. Rahman, "Analysis and evaluation of five short term load forecast techniques", IEEE Trans. On Power Systems, vol.4, no.4, 1989, pp. 1484-1491

[3] M.T. Hagen and S.M.Behr, "The time series approach to time series load forecasting", IEEE Trans, on Power Systems, PRWS-2(3), 1987, pp. 785-791

[4] A.D. Papalexopoulos , T.Hasterberg, "A Regression based Approach to Short Terni System Load Forecast", IEEE Trans. On Power Systems, vol.5, no.4, Nov. 1990, pp. 1535-1544

[5] S.R.Haung, "Short Term Load Forecasting using threshold autoregressive models", IEE Proc. C, vol. 144, no.5, Sept.1997, pp. 477-481

[6] W.R.Christiaanse, "Short Term Load Forecasting using general exponential smoothing", IEEE Trans. On Power Appar. Syst., PAS-3, 1988, pp. 900-911

[7] IEEE Committee report, "Load Forecasting Bibliography, Phase I", IEEE Trans, on Power App. Sys. vol. PAS-99, no.l , 1980, pp. 53-58.

[8] IEEE Committee report, "Load Forecasting Bibliography, Phase 11", IEEE trans, on Power App. Sys. vol. PAS-100, no.7, 1981, pp. 3217-3220

[9] Hong-Tzer Yang, Chao-Ming Huang, " A New Short-Term Load Forecasting Approach Self Organizing Fuzzy ARMA X Models " IEEE Trans, on Power Systems, vol. 13, no.l , Feb 1998, pp. 217-223

[10] Ajay Shekhar Pandey,D.Singh & S.K.Sinha, "Intelligent Hybrid Models For Short Term Load Forecasting", IEEE Transactions on Power System, vol. 25,Issue August 2010, pp. 1266-1273.

