Short-Term Load Forecasting Using Artificial Intelligence Techniques

V.Kumarathasan A.Sivabala L.Perera A.Aravinthan

Supervised by

Dr. H. Jahan C. Peiris Dr. Prientha D.P Wijeyathunga.

ABSTRACT

The primary objective of our project was to use artificial intelligence for forecasting the following half an hour Electricity power load in Sri Lanka using Neural Network and Fuzzy Logic and to evaluate their relative performance. We have also provided a user interface to interact with the system.

For predicting the following half an hour we have used historical data of Electricity Power load. We are using the current data to predict the following half hour load. In implementing our system we used Matlab 5.3 to implement both the Artificial Neural Network and Fuzzy Logic and for building GUI we used Matlab 5.3.

1.0 INTRODUCTION

Electricity demand estimates are the cornerstone of electricity planning. Without an adequate knowledge of the past and present electricity consumption patterns and the likely future development, electricity planning would be impossible. The necessity of accurate electricity demand forecasting is three-fold: generation expansion planning, transmission expansion planning, and financial planning. In addition, load forecasting helps the utility to determine the system's spinning reserve and fuel requirement, and plan their unit maintenance scheduling. The timely availability of sufficiently reliable electricity supply is vital to the proper functioning of the whole economy. Forced interruptions of electricity supply generally lead to substantial losses in output. Accurate demand estimates provide the decision-makers to undertake necessary measures to improve the reliability.

2.0 DATA PREPARATION

Holidays pose a special problem in load forecasting. But in our analysis we found there is no significant difference between normal days and special days. The second most important factor to affect the load profile is weather. In our analysis because of non-availability of temperature data in that detail we did not analyzed the relationship between load and temperature.

We assumed the load profile in particular day is depend mainly on the same day in the previous week and day in the two and three week ahead. We selected following variables as appropriate to build our model.

Feature variable x_3 implies information on half hourly load changes. x_1, Power load at time t reflects the features of a day of the week as well as weather conditions. The feature variable x_2 provides the weekly trend on a day. For our model implementation we have used first four months half hourly data to train the network and we used remaining months data to test our models. The exceptional data in a day are eliminated by taking average of the previous day and the following day.

$$x_{t+1} = f(x_1, x_2, \dots\dots\dots x_k)$$

Where,

x_{T+1}: Power load at time t+1

x_1, x_2, x_3: Feature variables at time t

$f(.)$: Non-linear mapping

x_1 : Power at time t, $x_1 = x(t)$

x_2 : Different between power load at time t and the average of power loads

$$x_2 = x(t) - x(t)_{mean}$$

x_3 : Difference of the average between time t and t+1

$$x_3 = x(t+1)_{mean} - x(t)_{mean}$$

Where,

$x(t)_{mean}$:

Average of three past days at time t on the same day of the week as the day to be predicted.

$$x(t)_{mean} = \frac{x(t-7\times48)+x(t-14\times48)+x(t-21\times48)}{3}$$

3.0 Artificial Neural Network Model Implementation

4.1 Choice Of Network Type

The first decision made about the implementation of Artificial Neural Network (ANN) was the choice of network type. The Feed Forward Neural Networks (FFNN) was chosen as the suitable one, because of its ability to come up with solutions for the ill-defined irregularities, through the use of hidden layers, as well as the control over the output of the network. But they do not have a way to achieve the proper weights to its connectors when the output is incorrect. However, this problem can be overcome by using back-propagation learning algorithm, as it is a basic mechanism to adjust the weights when the output is not matched with the target output.

3.2 Selection Of ANN Model

The selection of the ANN model architecture depends on the problem to be solved. The architecture includes the number of inputs, number of outputs, number of layers, number of neurons in each layer, etc. The number of layers and the number of neurons within each layer depends on the efficiency of the learning rule. The trial and error process is carried out until the error (the difference between the desired output and actual output) is reduced to an acceptable level.

3.3 Inputs And Output Of ANN Models

The input consists of the current load value and two feature variables corresponding to that. The output will be the load after half an hour. Learning and testing data are prepared for January to April and rest of the year respectively.

Date & Time	x_1	x_2	x_3	Target
22-Jan-1997, 0:00	350.1	-35	-24	342.1
22-Jan-1997, 0:30	342.1	-19	-17	346.1
22-Jan-1997, 1:00	346.1	2	17	339.1
22-Jan-1997, 1:30	339.1	12	-6	330.6
22-Jan-1997, 2:00	330.6	16	4	342.1
22-Jan-1997, 2:30	342.1	23	12	358.6
22-Jan-1997, 3:00	358.6	27	26	395.1

Table 1 *Training Data*

The sampling is half hour. One-step ahead prediction is made with the proposed method. The data prepared for training is shown in table1 and the data prepared for testing is shown in table2.

Date & Time	x_1	x_2	x_3
1-May-1997, 0:00	343.9	-32.9	- 36.5
1-May-1997, 0:30	316.4	-23.9	- 17.7
1-May-1997, 1:00	303.5	-19.0	- 9.4
1-May-1997, 1:30	290.5	-22.6	- 4.1
1-May-1997, 2:00	287.0	-22.0	- 2.3
1-May-1997, 2:30	282.1	-24.5	- 1.2
1-May-1997, 3:00	280.0	-24.6	- 0.03

Table 1 *Testing Data*

We achieved a model with three hidden layer, having twelve nodes in layer-1 and layer-2 and six nodes in layer-3 followed by an output layer. It satisfied the maximum performance level that we required. Fig 1 shows the architecture of the model selected.

3.4 Learning Rule and Learning Parameters

The process of changing the weights of the connections to achieve some desired results is referred to as learning. The simplest implementation of back-propagation learning updates the network weights and biases in the direction in which the performance function (which describes the error of the updates) decreases most rapidly - the negative of the gradient. The performance function is the mean square error MSE - the average squared error between the network outputs (a) and the target outputs (t).

$$F = \frac{1}{N}\sum_{i=1}^{N} e_i^2 = \frac{1}{N}\sum_{i=1}^{N}(a_i - t_i)^2$$

The RMSE is the root value of *F,* which was used to measure the performance of the networks we tried. Our network was trained until the RMSE to reduce up to 0.001.

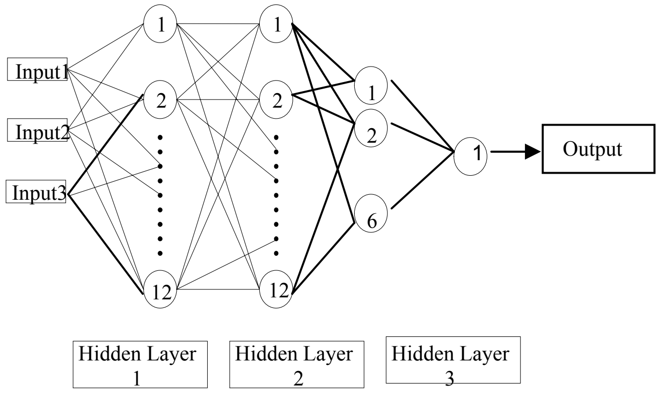


Fig 1 *Fully Connected 12:12:6:1 ANN*

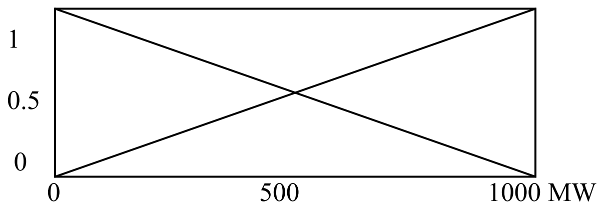
3.5 Transfer Function

The choice of transfer function is an important design issue in achieving fast convergence and good generalization of the network. The back-propagation requires that the transfer function, which determines the input-output relation of each node, should be differentiable at all points. Hence our network architecture used sigmoid functions such as ***tansig*** is used in the first layer, ***logsig*** is used in second hidden layer and ***purelin*** transfer function is used in the output layer to get the full range of output.

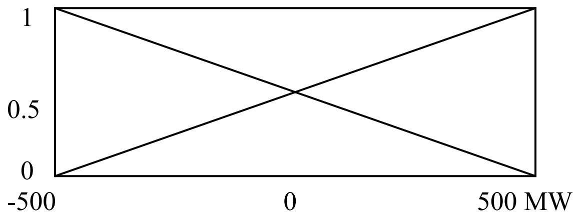
4. 0 Fuzzy Logic Model Implementation

Learning and testing data are prepared for January to April and rest of the year respectively. The sampling is half hour. One-step ahead prediction is made with the proposed method. Figure 2 shows the membership functions of four variables x_1, x_2, x_3 and **Time of Day.**

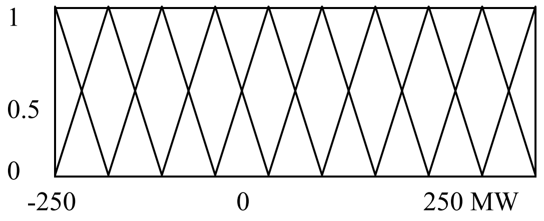
$x_1 = x(t)$



$x_2 = x(t) - x(t)_{mean}$



$x_3 = x(t+1)_{mean} - x(t)_{mean}$



Time of day

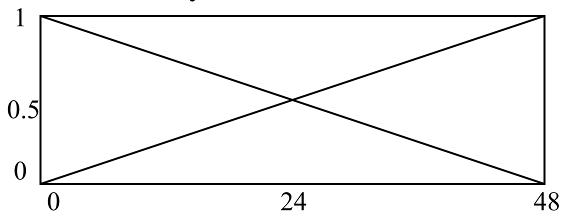


Fig 2 *Obtained Membership Functions*

There are two types of fuzzy inference systems that can be implemented in the Fuzzy Logic Toolbox,

Mamdani-type inference

A type of fuzzy inference in which the fuzzy sets from the consequent of each rule are combined through the aggregation operator and the resulting fuzzy set is defuzzified to yield the output of the system.

Sugeno-type inference

A type of fuzzy inference in which, the consequence of each rule is a linear combination of the inputs. The output is a weighted linear combination of the consequents.

These two types of inference systems vary somewhat in the way outputs are determined. The main difference between Mamdani-type of fuzzy inference and Sugeno-type is that the output membership functions are only linear or constant for Sugeno-type fuzzy inference. Because of the following reasons we have selected the Sugeno type inference system to implement our Fuzzy Logic Model,

- It's computationally efficient.
- It works well with linear techniques.
- It works well with optimization and adaptive techniques.
- It's well-suited to mathematical analysis.

We have used Fuzzy Logic Toolbox, which is available in MatLab 5.3 to build, train and to test. The Fuzzy Toolbox GUI editor provides the following components

- Fuzzy Inference System Editor
- Membership Function Editor
- Rule Editor
- Rule Viewer
- Surface Viewer

4.1 Fuzzify the Inputs

There are many Membership Functions (MF). We have selected the Triangular function as MF for all inputs, since it gave better performance. In our project, we have selected the output membership function as linear. Table1 shows summary of MF and input levels.

Triangular Function

The triangular curve is a function of a vector, x, and depends on three scalar parameters a, b, and c, as given by the parameters a and b locate the "feet" of the triangle and the parameter b locates the peak as shown in Fig 3.

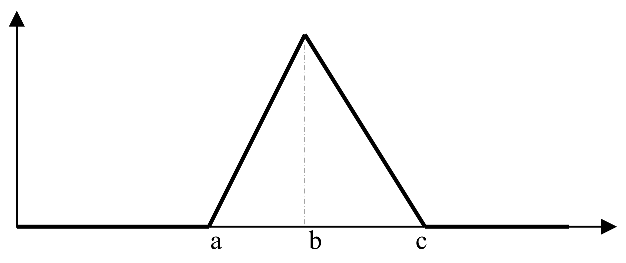


Fig 3 *Triangular Function*

Type	**Member Function**	**No. of levels**
Inputs		
x_1	Triangular	2
x_2	Triangular	2
x_3	Triangular	10
Time of Day	Triangular	2

Table 3 *summary of MF and input levels*

5. 0 Analysis of Results

$$\text{Root Mean Squared Error} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Actual_i - Predicted_i)^2}$$

$$\text{Average Error Percentage} = \frac{1}{N}\sum_{i=1}^{N}\left\{\frac{|Actual_i - Predicted_i|}{Actual_i} * 100\right\}$$

The error calculated here is for a particular day for compare the results between models

Root Mean Squared Errror of ANN Model = 19.43 MW
Root Mean Squared Errror of Fuzzy Model = 15.89 MW

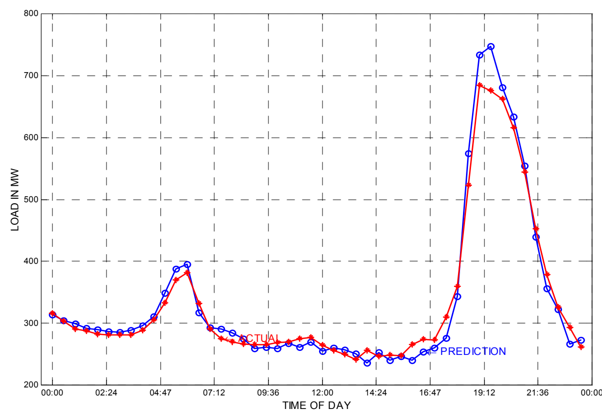


Fig 4 Prediction of a Sample day (1-May-1997)

6. 0 Conclusion

We have designed and successfully developed fuzzy logic model and neural network model for short term power load forecasting. The performances of the models are in acceptable level of accuracy

Reference

1) Eric W. Tyree & J. A. Long, 1995
City University, Presented at the Third International Conference on Artificial Intelligence Applications on Wall Street New York, 1995

2) Winston, P. H., 1992
Artificial Intelligence: 3rd edition (USA: Addition-Wesley).

3) Optimum Fuzzy Inference for Short-Term Load Forecasting
IEEE Transactions on Power Systems, Vol.11, No.1, February 1996.