VOL. 3, NO. 2, APRIL 2008 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

LOAD FORECASTING BY A NOVEL TECHNIQUE USING ANN

T. Gowri Manohar and V. C. Veera Reddy
Department of Electrical and Electronics Engineering, S.V.U. College of Engineering, Tirupati, A.P., India
E-mail: tgmanohar1973@rediffmail.com

ABSTRACT

Basically the active power demands at various load buses need to be estimated ahead of time in order to plan the generation and distribution schedules, for contingency analysis and also for monitoring the system security. A great deal of attention has been given in recent years to the question of setting up the demand models for the individual appliances and their impact on the aggregated demand. For the allocation of spinning reserve it would be necessary to predict the load demands at least an hour ahead.

A method using ANN based technique is developed for short-term load forecast. The technique is tested on real time data collected from a 220 KV / 132 KV / 33 KV / 11 KV Renigunta Sub-Station, A.P, India. Calculations are done based on the hourly data of active power variations obtained over a period of one month. The active powers were used as input quantities for training the ANN and obtained the respective output active powers for the corresponding day.

Keywords: short-term, load, forecasting, ANN.

1.0 INTRODUCTION

For the purpose of optimal planning and operation of large-scale power systems, modern control theory and optimization techniques are being applied with the expectation of considerable reduction in cost. In achieving this goal, the knowledge of future load on power system is the first prerequisite, therefore, long and short-term load forecast are very important.

Short-term [1] load forecast plays a key role in the formulation of economic, reliable and secure operating strategies for the power system. The principal objective of the STLF function was to provide load predictions as basic inputs for the following:

- The basic generation scheduling functions.
- Assessing the security of the power system at any point of time.
- Timely dispatcher information.

Load forecasting with lead times, from a few minutes to several days, helps the system operator to schedule spinning reserved allocation effectively. The short-term load forecast (STLF) is needed for control and scheduling of power system, and also as inputs to load flow or contingency analysis. The STLF can also provide wide information regarding vulnerable situations that may take place, in advance.

The manually entered data may include weather updates, old forecasting parameter data. The significant effect of STLF on power system operations and production costs depends upon its accuracy. System dispatches must anticipate the system load patterns so as to have sufficient generation. The errors in load forecasts could affect in planning reserve requirement. Underproduction of load results in a failure to provide the necessary reserves, which in turn, translates to higher costs due to the use of the expensive peaking units. Over production of load, on the other hand, involves the start-up of too many units resulting in unnecessary increase in reserves and hence operating costs. Hence it is necessary to see that forecasted data has least errors.

Various algorithms [2, 3] have been suggested in the literature for STLF. They are mainly classified into:

- Time series models
- Regression models

Time-Series models employ the historical load data for extrapolation to predict future load. These models assure that the load term in stationary and treat any abnormal data point as bad data. General problems with the time series approach include the inaccuracy in predictions and numerical instability. Since these models do not utilize weather information, they often give inaccurate results as there is a strong correlation between the behavior of power consumption and weather variables such as temperature, humidity, cloud cover and wind speed.

Regression models analyze the relationship between weather variables and loads. The conventional regression approaches use linear or pise-wise linear representations. The regression approach finds the functional relationships between selected weather variables and load demands.

G. Gross *et al.*, [4] discuss the state of the art in short-term load forecasting. The Paper reviews the important role of STLF in the on-line scheduling and security functions of an energy management system (EMS). It then discusses the nature of the load and the factors influencing its behavior.

H. Mori *et al.*, [5] present a clustering method for preprocessing input data of short-term load forecasting in power systems. Clustering the input data prior to forecasting with the artificial neural network (ANN) reduces the prediction errors observed. In this Paper, an ANN is used to deal with one-step ahead daily maximum load forecasting, and the deterministic annealing (DA). Clustering is employed to classify input data into clusters.

S.Vemuri *et al.,* [6] present a study to investigate whether the ANN model is system dependent, and/or case dependent. Data from two utilities are used in modeling and forecasting. The effectiveness of a next 24 hour ANN model in predicting 24 hour load profile at one time was compared with the traditional next 1 hour ANN model. The author mentions that weekend and holiday load forecasts deserve further investigation.

I.Drezga et al., [7] describe a method for the input variable selection for artificial neural network based short-term load forecasting. This method is based on the phase-space embedding of a load-time series. It is stated that the accuracy of this method depends on temperature and cycle variables.

A novel method is proposed in this paper that is devoid of the draw backs which are observed in the above mentioned papers.

2.0 BACK PROPAGATION TECHNIQUE

This back propagation [8] is useful for training multi layer artificial neural network. Back propagation is a systematic method for training multi layer artificial neural networks and it has mathematical foundation very strongly. Back propagation has dramatically expanded the range of problems to which artificial neural networks can be applied.

A set of inputs is applied either from the outside or from a previous layer. Each of these is multiplied by a weight and the products are summed. This summation is termed as "NET" and must be calculated for each neuron in the network. After "NET" is calculated the activation function f is applied to modify it, there by producing the signal OUT.

$$OUT = \frac{1}{(1-e^{-NET})} \quad \text{..... (1)}$$

$$NET=X_1W_1+X_2W_2+\text{----------}+X_NW_N \quad \text{..... (2)}$$

$$OUT=f(NET) \quad \text{.....(3)}$$

$$\frac{\partial\, OUT}{\partial\, NET} = OUT(1-OUT) \quad \text{..... (4)}$$

This function is called sigmoid.

The sigmoid compress the range of NET so that 'OUT' lies between zero and one. Multi layer networks have greater representational power than single layer network only if non-linearity is introduced. The back propagation algorithm requires only that the function be every where differentiable. The sigmoid satisfies these requirements.

A. The multilayer network

For training with back propagation a multi layer network may be considered. The first set of neurons, connecting to the inputs, serve only as distribution points, they perform no input summation. The input signal is simply passed on through the weights to their outputs. Each neuron in subsequent layer produces NET and OUT signals as described above.

B. An overview of training

The objective of training the network is to adjust the weight. So that application of a set of inputs produces the desired set of outputs. These input – output sets can be referred to as vectors. Training assumes that each input vector is paired with a target vector representing the desired output, and these are called a training pair. Usually, a network is trained over a number of training pairs. This group of training pairs is called a training set.

Before starting the training process, all the weights must be initialized to small random numbers. This ensures that the network is not saturated by large values of the weights, and prevents certain other training pathologies.

For example, if the weights all start at equal values and if the desired performance requires unequal values, the network will not learn.

Training the back propagation network requires the following steps:

1. Select the next training pair from the training set; apply the input vector to the network input.
2. Calculate the output of the network.
3. Calculate the error between the network output and the desired output (the target vector from the training pair).
4. Adjust the weights of the network in a way that minimizes the error.
5. Repeat steps 1 to 4 for each vector in the training set until the error for the entire set is acceptably low.

The operations required in steps 1 and 2 above are similar to the way in which the trained network will ultimately be used; that is, an input vector is applied and the resulting output is calculated. Calculations are performed on a layer-by-layer basis. First the output of the neurons in layer j is calculated; these are then used as inputs to layer k; the layer k neuron outputs are calculated and these constitute the network output vector.

In step 3, each of the network outputs, labeled OUT is subtracted from its corresponding component of the target vector to produce an error. This error is used in step 4 to adjust the weights of the network, where the changes in polarity and magnitude of the weight are determined by the training algorithm.

After enough repetitions of these four steps the error between actual outputs and target outputs should be reduced to an acceptable value and the network is said to be trained. At this point, the network is used for recognition and weights are not changed.

Steps 1 and 2 can be expressed in vector from as follows; an input vector X is applied and an output vector Y is produced. The input target vector pair X and T comes from the training set. The calculations are performed on X to produce the output vector Y.

As we have seen calculation in multi layer networks is done layer by layer, starting at the layer nearest to the inputs. The NET value of each neuron in the first layer is calculated as the weighted sum of its neuron's inputs. The activation function 'f' then squashes NET to produce the OUT value for each neuron in that layer. Once the set of outputs for a layer is found, it serves as input to the next layer. The process is repeated layer by layer, until the final set of network outputs is produced.

C. Adjusting the weights of the output layers

Because a target value is available for each neuron in the output layer, adjusting the associated weights is easily accomplished using a modification of the delta 8 rule. Interior layers are referred to as "hidden layers" as their outputs have no target values for comparison.

The output of neuron in layer k is subtracted from its target value to produce an error signal. This is multiplied by the derivative of the squashing function (OUT * (1-OUT)) calculated for that layer's neuron k, thereby producing the 'δ' value.

δ = OUT (1-out) (Target – OUT) ……..…….... (5)

Then 'δ' is multiplied by OUT from neuron in layer j, the source neuron for that weight in question. This product is in turn multiplied by a training rate coefficient η (typically 0.01 to 1.0) and the result is added to the weight. An identical process is performed for each weight proceeding from a neuron in the hidden layer to a neuron in the output layer.

D. Adjusting the weights of the hidden layer

Hidden layers have no target vector, so the training process described above can't be used. Back propagation trains the hidden layers by propagating the output error back through the network layer-by-layer, adjusting the weights at each layer.

For hidden layer, 'δ' must be generated without benefit of a target vector. First, 'δ' is calculated for each neuron in the output layer. It is used to adjust the weights feeding into the output layer, then it is propagated back through the same weights to generate a value of 'δ' for neuron in the first hidden layer. This value of 'δ' is used in turn to adjust the weights of this hidden layer and in a similar way, are propagated back to all preceding layers.

Consider a single neuron in the hidden layer just before the output layer. In the forward pass, this neuron propagates its output value to neurons in the output layer through the interconnecting weights. During training these weights operate in reverse, passing the value of 'δ' from the output layer back to the hidden layer. Each of these weights is multiplied by the 'δ' value of the neuron to which is connects in the output layer. The value of 'δ' needed for the hidden layer neuron is produced by summing all such products and multiplying by the derivative of the squashing function.

$$\delta_{pj} = OUT_{pj}\left(1-OUT\right)\left(\sum_{q=1}^{n}\delta_{qk}.w_{pq}\right) \quad \ldots\ldots\ldots\ldots \quad (6)$$

E. Back Propagation Algorithm

The procedural steps considering 3 layer network are as follows:

Step 1: Initialize weights and offsets

Set all weights and offer (Bias) to small random values.

Step 2: Present input and desired outputs

Present a continuous valued input vector x_1, x_2 …… x_{n1} where N is the number of input nodes. Specify the desired (target) output T_1, T_2 ….. T_N where N_j is number of output nodes. The input could be new on each trial and be presented cyclically until weights stabilize.

Step 3: Calculate outputs at hidden layers.

Use the sigmoidal nonlinerity to calculate output, Y_1, Y_2….. Y_{NH} at hidden layer by using the following relations.

Step 4: Calculate the output at the output layer

Calculate the output at the output layer using the sigmoidal nonlinearly and output at the previous stage.

Step 5: Calculate Errors.

Compare the output with the target or desired outputs. Find the error, k which is the difference between the desired and calculated outputs.

Step 6: Back propagate the error.

Back propagates the error at the output layer to the hidden layer.

Step 7: Adapt weights using Error Back propagation rule. Adjust the weights (hidden-output) by using Error Back Propagation Rule.

Step 8: Repeat by going to step 2.

F. Implementation strategies of back propagation algorithm

The back propagation algorithm can be successfully implemented by using two types of strategies. They are using true gradient descent rule and approximate gradient descent rule.

The first method is also known as Batch Processing. This method follows the generalized delta rule. Weights will be adapted after placing all the training patterns in to the network. This is known as cycle or iteration. After each cycle, the weights will be adjusted using the error over all the patterns as given by the following equation.

$$w_{ji} = \sum P_j * Y_{pi} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7)$$

Where $P_j = \left(T_{pj} - Y_{pj}\right) f\left(NET_{pj}\right)$ is neuron j in output layer.

Momentum factor is not included in the above equations for the sake of simplicity.

The second method, also known as sequential processing adapts weights in the network after placing every training pattern to the network.

$$W_{pji} = W_{pji} * O_{pj} * Y_{pj} \quad \text{.....} \quad (8)$$

Where W_{pji} is the weight changed during p^{th} pattern for weight connecting j-th neuron of the layer under consideration to the i^{th} neuron of the previous layer. The batch processing i.e., true gradient descent rule, converges very slowly as the weights are adjusted only once in a cycle and it takes more memory to accommodate the error values for all patterns.

The sequential processing i.e., approximate gradient descent rule is considered to converge fast in many practical cases as the weights are adjusted after presenting every training pattern. This method suffers from the problem of over correcting the weights which gives rise to oscillations in the error. The deviation from the gradient descent rule makes it difficult for the network to reach the minimum error position. The amount of deviation mainly depends on the learning rate used (η) and momentum factor (α) employed. As long as the learning rate is small, departure from the true gradient descent rule will be negligible and the delta rule will implement a close approximation to gradient descent rule.

3.0 NETWORK CONFIGURATION AND ARCHITECTURE

The network structure also affects the accuracy of the forecast. The relationship between the input and output determines the network structures, which can be approximated by

- A linear function of the input variables
- A non linear function of the input variables.
- A combined linear and nonlinear function of the input variables.

Network configuration mainly depends on the number of hidden layers, number of neurons in each hidden layer and the selection of non-linearity. No clear cut guide lines exists up to date for deciding the architecture of ANN, though it is problem dependent. However, too small ANN can cause problem in learning/ memorization as well as poor generalization. Whereas, too large ANN can memorize very well but it can over generalize the relationship among inputs and outputs. The overgeneralization tends to produce poor performance for unseen data. In general a three layer ANN is employed.

For a three layer ANN, the number of hidden neurons can be selected by one of the following thumb rules:

1. (i-1) hidden neurons, where i is the number of input neurons.
2. (i+1) hidden neurons, where i is the number of input neurons.
3. For every 5 input neurons, 8 hidden neurons can be taken. This is developed seeing the performance of network within 5 inputs, 8 hidden neurons and 1 output.
4. (Number of input neurons, I) / (Number of output neurons, m), applicable for networks with m<i.
5. Half the sum of input and output neurons.
6. P/i neurons, where i is the input neurons and p represents number of training samples.

In the present work, only configuration 2, 3 are used for comparison, even though the other configurations can also be tried.

4.0 SYSTEM MODELING

The System model involves the identification of parameters that influence prediction of the future load. On the basis of experience of the load dispatchers, the features of the load curve and the existing practices of load management, the inputs could be selected.

A major part of the forecasting task is concerned with that of identifying the best possible model for the past load behavior. This is best achieved by decomposing the load model at any given point of time into a number of distinct components.

In the present work, five inputs are selected from the previous day and five each from the previous weeks on the same day. Thus load inertia, day of the week effect is considered as the major factors. In this model, the inputs considered for training the network to predict the load of day d and hour h, L(d-h), are L(d-1, h-2), L(d-1, h-1), L(d-7,h-1), L(d-7, h+1), L(d-7, h+2) of the previous week same day hours are considered to smoothen the curve to eliminate the edging effect, thus making total of 10 inputs.

A. Selection of network parameters

Most works on feed forward networks uses constant learning rate beta (η) and momentum factor alpha (α) values. Generally, η = 0.25 and α = 0.9 yield good results for most of the problems. In fact the optimal values of and n are problem dependent. In this project η= 0.25 and α = 0.75 are employed in training the ANN.

B. Selection of nonlinearity

Neural networks process the capability to generalize complex patterns and can represents non – linear function of any degree. The extent of non-linearity depends on activation function used. Hence, sigmoidal function which is universally preferred as an activation function is found to be satisfactory for many applications. In sigmoidal function abruptness factor (g) also affects its nonlinearity. When g is not mentioned it is assumed to be 1.0. In this work 'g' is taken as 1.0.

C. Normalization and denormalisation

The inputs and targets to the neural network are to be in the range of 0.0 and 1.0 if any of the variables, say output variable assumes a value close to unity or zero, it causes difficulty in training as one or zero are never realized practically by the activation or threshold function. A way to overcome this difficulty is to normalize the variables between a suitable range of 0.1 to 0.9.

Normalization of data should be done such that higher values should not suppress the influence of lower values sand symmetry of activation function is retained. Many methods for normalization are specified, but most widely used method using linear transformation is as follows:

The possible minimum and maximum values are to be specified considering both training and testing patterns. Normalization can be done by using same minimum and maximum values for different types of input and output variables treating input and / or output variables of same type as one group. The resolution will be high if later is used. By extending the same logic more resolution can be achieved when separate minimum and maximum values are used for every input and / or output variable.

Denormalization

Denormalization is the inverse process of normalization. As the output of the neural network will be in the range of 0.0 to 1.0 it is necessary to demoralize the output of the network in order to interpret their result and to assess the performance of the networks.

It is very important to see that the denormalization is an exact inverse process of normalization. The same minimum and maximum values are used in the normalization and denormalization.

D. Initializing weights

The aim of the training process is to arrive at a set of weights which satisfies the input and output relationship specified by the training examples. It is obvious that if the initial weights are close in values to the final weights then the number of iterations needed during training will be less, but there is no method or logic to arrive at initial weights which are close to the final weights, only leaving possibility of having arbitrary initial weights.

Rumelhart pointed out that if all the initial weights are same in value or zero there is a possibility of starting process at a global / local maxima which in most of the cases doesn't lead to reach global minima on error surface. The only one way to avoid this is to select all initial weights randomly. The range in which the initial weights will lie is considered to affect the number of iterations required. In the proposed work a small random weights I the range of 0.5 to 5.5 are used.

Once the trained networks are finalized, as the additional training patterns are included, the network can be further trained progressively for a fixed number of iterations, starting from the final converged weights, rather than starting from random weights.

E. Number of training patterns

The minimum numbers of patterns required are half the number of input neurons and the maximum is equal to the product of number of input neurons and number of hidden neurons or five times the number of input neurons, which ever is less. However, no justifications can be given to such thumb rules. In this work, performance of configurations 2-3 are studied with number of training patterns more than 7 for every network.

5.0 CASE STUDIES AND RESULTS

Load details are obtained from 220 KV / 132 KV/ 33 KV/11 KV Renigunta sub-station. Several case studies were conducted and the results obtained for one of the case studies are furnished as follows.

A. Forecast of Renigunta Sub-Station

Network Configuration	:	network - 1
No. of input nodes	:	10
No. of output nodes	:	1
No. of hidden nodes	:	21

This network is trained for 1 to 12 hours on 10th January. The 10 inputs presented for any pattern with target hour 't' are taken from the previous day i.e., 9th January, t-2,t-1, t, t+1, t+2 hours (5hours) and from the previous week of same day i.e., 3rd January, t-2,t-1, t, t+1, t+2 hours (5hours). All the input and output data are active powers in megawatts.

Table-1. Training Data.

Pattern No.											Out put data
1	80	88	88	92	100	76	74	76	80	90	80
2	84	80	88	88	92	75	76	74	76	80	80
3	88	88	92	100	100	74	76	80	80	104	90
4	88	92	100	100	120	76	80	90	104	108	100
5	92	100	100	120	128	80	90	104	108	110	108
6	100	100	120	128	132	90	104	108	110	112	120
7	100	120	128	132	130	104	108	110	112	104	128
8	120	128	132	130	118	108	110	112	104	104	132
9	128	132	130	118	126	110	112	104	104	108	138
10	132	130	118	126	106	112	104	104	108	100	128

Table-2. Training output in normalized form.

Pattern No.	Output Calculated	Target	Error
1.	0.177791	0.175	-0.002791
2	0.173286	0.175	0.001714
3.	0.298854	0.3	0.001146
4.	0.425416	0.425	-0.000416
5.	0.524335	0.525	0.000665
6.	0.676235	0.675	-0.001235
7.	0.773668	0.775	0.001322
8.	0.825921	0.825	-0.000921
9.	0.898349	0.9	-0.001651
10.	0.7778684	0.775	-0.000684

Table-3. Training output in denormalised form.

	Output Calculated	Target
1.	80.223315	80
2.	79.862884	80
3.	89.908375	90
4.	100.003328	100
5.	107.946762	108
6.	120.098782	120
7.	127.8934	128
8.	132.073681	132
9.	137.867953	138
10.	128.054724	128

The training network is tested for 4^{th}, 6^{th} and 8^{th} hours 11^{th} January. The 10 inputs presented for any pattern with target hour't' are taken from the previous day i.e., 10^{th} January t-2, t-1, t, t+1, t+2 (5 hours) add from the previous week of same day i.e., 4^{th} January t-2, t-1, t, t+1, t+2 (5 hours).

Table-4. Input data.

Patte-rn No.	Input Data									
1	80	90	100	108	120	80	76	80	92	108
2	100	108	120	128	132	80	92	108	112	122
3	120	128	132	138	128	108	112	122	108	108

Table-5. Testing output in denormalized form.

Pattern No.	**Actual Output** (MW)
1	83.90567
2	117.362076
3.	128.324297

6.0 CONCLUSIONS

In this work an attempt is made to develop a software package for obtaining and studying "SHORT TERM LOAD FORECASTING USING ARTIFIFICAL NEURAL NETWORKS". It is aimed to present the results of investigation carried out in this work with the real time data collected from RENIGUNTA substation, A.P, India.

The training of ANN for the prediction of the load of a particular day includes

No. of training patterns = 8

No. of input nodes = 10

No. of hidden nodes = 21

No. of output nodes = 1

Training of neural network is done with learning rate, $\eta = 0.25$ and momentum factor, $\alpha = 0.75$. The final converged weights are used to test on pattern sets the results of the actual value obtained from ANN are tabulated.

Finally, it is conclude that the neural network based forecasting of load provide solution much faster

than the conventional methods without degrading the accuracy the development and enhancement of training algorithms enables the application of ANN to still larger power systems thus making it suitable for real time applications.

ACKNOWLEDGEMENT

The authors express their deep sense of gratitude to the SPDCL staff for providing the required data to carryout the proposed work. The authors also express their heartfelt thanks to the Department staff members for giving their valuable suggestions in obtaining the results by the proposed method.

REFERENCES

[1] Mahalanabis A.K., Kothari P. and Ahson S.I. 1990. Computer Aided Power System Analysis and Control. Tata McGraw-Hill, India.

[2] Gross G., Galina F.D. 1987. Short Term Load Forecasting. Proceedings of IEEE. 75(12): 1558-1573.

[3] Khaparde S.A., Lohitha A., Desai. 1991. U.B. Load Forecasting using ANN. IEEE Tencon. 91: 208-212.

[4] G. Gross and F.D. Galina. 1987. Short Term Load Forecasting. Proceedings of IEEE. 75(12): 1558-1573.

[5] H. Mori and A. Yuihara. 2001. Deterministic annealing clustering for ANN-based short-term load forecasting. IEEE Transactions on Power Systems. 16: 545-551.

[6] S. Vemuri *et al.* 1993. Neural network based short term load forecasting. IEEE Transactions on Power Systems. 8: 336-342.

[7] I. Drezga and S. Rahman. 1998. Input variable selection for ANN-based short-term load forecasting. IEEE Transactions on Power Systems. 13: 1238-1244.

[8] Haykin S. 2000. Neural Networks: A comprehensive survey. Pearson Education, India.