ANN Based Techniques for Short Term Load Forecasting

P. B. Duttagupta and Debajyoti Roy

Abstract—Neural network techniques have been suggested for short-term load forecasting by a large number of researchers. This paper studies the applicability of this kind of a technique using various training algorithms. The work is intended to be a basis for a real forecasting application. First, a literature survey was conducted on the subject. Most of the reported algorithms are based on the so-called Multi-Layer Perceptron (MLP) network. There are numerous training suggestions, but the large variation and lack of comparisons make it difficult to directly apply proposed methods. It was concluded that a comparative study of various training algorithms seems necessary. Two training rules were developed and tested on the real load data of a French electric utility. Most of them use a MLP network to identify the assumed relation between the future load and the earlier load data. The models were divided into two classes. First, forecasting the load for a whole day at once was studied. Then hourly models, which are able to update the forecasts as new data arrives, were considered. The test results showed that the hourly models are more suitable for a forecasting application due to their speed and reliability. The paper suggests that this kind of an hourly neural network model should be implemented for a thorough on-line testing in order to get a final opinion on its applicability.

KeyWords—Short Load Forecasting (STLF), Neural Networks, Multi-Layer Perceptron (MLP) networks, Artificial Neural Network (ANN).

I. INTRODUCTION

Load forecasting is one of the central functions in power systems operations. The motivation for accurate forecasts lies in the nature of electricity as a commodity and trading article; electricity can not be stored, which means that for an electric utility, the estimate of the future demand is necessary in managing the production and purchasing in an economically reasonable way. Load forecasting methods can be divided into very short, short, mid and long term models according to the time span. In very-short term forecasting the prediction time can be as short as a few minutes, while in long-term forecasting it is from a few years up to several decades. This work concentrates on short term forecasting, where the prediction time varies between a few hours and about one week. Short-term load forecasting has been lately a very commonly addressed problem in power systems. One reason is that recent scientific innovations have brought in new approaches to solve the problem. The development in computer technology has broadened possibilities for these and other methods working in a real time environment. A majority of the recently reported approaches are based on neural network techniques. The attraction of the methods lies in the assumption that neural networks are able to learn properties of the load, which would otherwise require careful analysis to discover. However, the development of the methods is not finished, and the lack of comparative results on different model variations is a problem. Therefore, to make use of the techniques in a real application, a comparative analysis of the properties of different model types seems necessary.

This paper studies the applicability of different neural network training algorithms on short-term load forecasting. The approach is comparative. The models are divided into two classes: models forecasting the load for one whole day at a time, and models forecasting ahead hour by hour. Testing is carried out on the real load data of a French electric utility. The objective is to accomplish suggestions on choosing the most appropriate training algorithm.

The various parameters which are believed to have an impact on the load demand can be enumerated as follows

- In the short run, the meteorological conditions cause large variation in this aggregated load. In addition to the temperature, also wind speed, cloud cover, and humidity have an influence.
- In the long run, the economic and demographic factors play the most important role in determining the evolution of the electricity demand.
- From the point of view of forecasting, the time factors are essential. By these, various seasonal effects and cyclical behaviors (daily and weekly rhythms) as well as occurrences of legal and religious holidays are meant.

The other factors causing disturbances can be classified as random factors. These are usually small in the case of individual consumers, although large social events and popular TV–programs add uncertainty in the forecasts. Industrial units, on the other hand, can cause relatively large disturbances. Only short-term forecasting will be dealt in this work, and the time span of the forecasts will not range further than about one week head. Therefore, the economic and demographic factors will not be discussed.

II. ANN MODELS

As an N-dimensional input vector is fed to the network, an
M-dimensional output vector is produced. The network can be understood as a function from the N-dimensional input space to the M-dimensional output space. This function can be written in the form:

\[ y = f(x; W) = \sigma (W_n \sigma (W_{n-1} \sigma (\ldots \sigma (W_1)x))) \]

where,
- \( y \) is the output vector
- \( x \) is the input vector
- \( W_i \) is the matrix containing the weights of the \( i \)th hidden layer.

The neuron weights are considered as free parameters. The most often used MLP-network consists of three layers: an input layer, one hidden layer, and an output layer. The activation function used in the hidden layer is usually nonlinear (sigmoid or hyperbolic tangent) and the activation function in the output layer can be either nonlinear (a nonlinear-nonlinear network) or linear (a nonlinear-linear network). For the present work feed forward and back propagation network was adopted for ANN modeling. Using heuristics the input layer nodes were designed to contain pertinent data that were considered to have a major relation to the desired forecast value. Historical data from an electric utility was used in deriving the models based on the following procedure.

Input layer nodes were made up of the following components which affect the hourly load viz.
- Bias
- Hour of the day
- Day of the week
- Past load data

Moving time window was used to select the data to train the network. Historical data were divided into 12 learning and recalling sets each learning set containing one month's data, and recalling set containing one week data. Data in the learning and recalling sets had a record format defined by the network structure under study. Different networks were then trained and tested. During the training of a network, different combinations of the following network training methods were chosen and tested to be sure that model was continuously refined. Through a series of tests and modifications, the network architecture shown in Table 1 for the two different forecasting procedures was found to be best for the particular case. The bias node is used as feed forward term and feeds into all hidden layer nodes and the output node. A Linear transfer function is used for the input layer and everywhere else the sigmoid transfer function is used. In case of the next hour load forecasting model the input layer comprises of one bias node, 5 hour of the day nodes, 7 day of the week nodes and finally 4 nodes for the historic data which in this case was the past four hour load data. Similarly in case of the 24 hour load data nodes of the input layer were chosen except the fact that over here the hour of the day consideration did not matter a great deal. The idea behind using a day of the week information as input is to take into account the fact that load profiles of week days and weekends are totally different from each other.

<table>
<thead>
<tr>
<th>MNL Architecture</th>
<th>Next Hour Load Forecast</th>
<th>Next 24 Hour Load Forecast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>17 (1:bias, 5:hour, 7:day, 4:load)</td>
<td>32 (1:bias, 7:day, 24:hour)</td>
</tr>
<tr>
<td>Hidden Layer</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Input Layer</td>
<td>1</td>
<td>24</td>
</tr>
</tbody>
</table>

### III. ANN Training

The network weights are adjusted by training the network. It is said that the network learns through examples. The idea is to give the network input signals and desired outputs. To each input signal the network produces an output signal, and the learning aims at minimizing the sum of squares of the differences between desired and actual outputs. From here on, we call this function the sum of squared errors. The learning is carried out by repeatedly feeding the input-output patterns to the network. One complete presentation of the entire training set is called an epoch. The learning process is usually performed on an epoch-by-epoch basis until the weights stabilize and the sum of squared errors converges to some minimum value. The most often used learning algorithm for the MLP-networks is the back propagation algorithm. This is a specific technique for implementing gradient descent method in the weight space, where the gradient of the sum of squared errors with respective to the weights is approximated by propagating the error signals backwards in the network. The recursive nature of this learning mechanism is shown below.

\[ \Delta W_{ij}(n+1) = \varepsilon \delta_j O_i + \Delta W_{ii}(n) \alpha \]

Where,
- \( O_i \): Output
- \( W_{ij} \): Weight of link joining output of neuron i to input of neuron j
- \( \varepsilon \): Adaptation Gain

The derivation of the algorithm is given, for example, in Haykin (1994). Also some specific methods to accelerate the convergence are explained there. A more powerful algorithm is obtained by using an approximation of Newton's method called Levenberg-Marquardt. In applying the algorithm to the network training, the derivatives of each sum of squared error (i.e. with each training case) to each network weight are approximated and collected in a matrix. This matrix represents the Jacobian of the minimized function. The Levenberg-Marquardt approximation is used in this work to train the MLP networks. Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feed forward networks), then the Hessian matrix can be approximated as \( H = J^T J \)
and the gradient can be computed as 
\[
g = \nabla e
\]
where, \( J \) is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and \( e \) is a vector of network errors.

The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:
\[
X_{k+1} = X_k - (J^T J + \mu I)^{-1} J^T e
\]
When the scalar \( \mu \) is zero, this is just Newton's method, using the approximate Hessian matrix. When \( \mu \) is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, \( \mu \) is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm. In essence, the learning of the network is nothing but estimating the model parameters. In the case of the MLP model, the dependency of the output on the model parameters is however very complicated as opposed to the most commonly used mathematical models (for example regression models). This is the reason why the iterative learning is required on the training set in order to find suitable parameter Neural networks in load forecasting values. There is no way to be sure of finding the global minimum of the sum of squared error. On the other hand, the complicated nonlinear nature of the input-output dependency makes it possible for a single network to adapt to a much larger scale of different relations than for example regression models. That is why the term learning is used in connection with neural network models of this kind.

IV. SIMULATION RESULTS AND DISCUSSION

Four arbitrary test weeks were chosen in different seasons of the year. The load of each day of the data set was forecast without using the actual load data of that day. Thereby, the hour by hour forecasting model was applied for each test day recursively 24 times. After forecasting the load patterns for each test day, these forecasts were compared with the real load data, and the average error percentages were calculated. In comparing different models, the average percentage forecasting error is used as a measure of the performance. This is defined:
\[
E_{\text{ave}} = \frac{1}{N} \sum_{i=1}^{N} \left[ \frac{\hat{L}_i - L_i}{L_i} \right] \times 100\%
\]
where,
- \( N \) = the number of cases to be forecast (number of hours in the case of hourly load forecasting).
- \( L_i \) = the \( i \)th load value
- \( \hat{L}_i \) = the \( i \)th load forecast

The reason for using the average percentage error is the fact, that its meaning can be easily understood. It is also the most often used error measure in the load forecasting literature used as reference of this work, and therefore allows for some comparative considerations (although the results are not directly comparable in different situations). Another possibility as a performance measure would be the root-mean-square (RMS) percent error. This penalizes the square of the error as opposed to the average forecasting error. Therefore, the RMS takes the deviation of the errors into account more effectively. However, when both measures were calculated on some test models with relatively small errors, the orders of preference were in practice the same with both measures. Therefore, the average forecasting error will be used throughout this work. In case of the hourly forecast the training algorithms of Gradient back propagation and Levenberg Marquardt were compared whereas in case of the 24 hour forecast the simulation was done only using the later. Although Levenberg Marquardt is a very fast training algorithm, it often has given fairly inaccurate results due to the large approximations it makes while calculating the Hessian matrix. The following are the hourly forecast results of a certain week.
The following bar graph compares the average percentage errors in case of the hourly forecasting using two different training algorithms viz. the Gradient back propagation and Levenberg Marquardt. It also gives an idea of the errors occurring in case of the 24 hour forecast using Levenberg Marquardt algorithm.

Fig 7. Power in MW.

V. CONCLUSION

Artificial Neural Network models provide a very useful tool for short term load forecasting. Radically different from statistical methods, these models have shown promising results in load forecasting. Based on the test results, it can be concluded that there is uniquely determined network structure that is best for a set of hourly load. Models are not unique, and systems with different load characteristics require different structure. However, once a model is identified for a given system, the training algorithm has to be chosen to optimize the speed and reliability of the forecasting procedure. The study indicates that, hourly forecasting is more fast and reliable but one has to make a choice between the two training algorithms. On one hand the Gradient back propagation rule is best for its precision whereas Levenberg Marquardt rule is found out to be extremely fast as far as the simulation goes. Neural network models are very much sensitive to bad data. Intelligent data filtering techniques need to be designed in order to maintain an acceptable accuracy in the ANN models based load forecasts. Since the ANN simply interpolates among the training data, it will give high error with the test data that is not close enough to any of the training data. In general, the neural network requires training data well spread in the feature space in order to provide highly accurate results. The training times required in our experiments vary, depending on the case studies, from 3 to 5 hours of CPU time using a PIII 550 MHz Windows® machine. Due to lack of availability of reliable weather information their effects could not be taken into consideration.

VI. ACKNOWLEDGEMENT

The authors would like to thank the department of Electrical Engineering IIT Kharagpur for all the support and affection.

VII. REFERENCES