DAY AHEAD LOAD FORECASTING USING AN ARTIFICIAL NEURAL NETWORK & ELMAN RECURRENT NETWORK

Ellen Banda, Komla A. Folly
Department of Electrical Engineering
University of Cape Town
South Africa
Email: bandael@eskom.co.za, komla.folly@uct.ac.za

ABSTRACT
The forecast of load for a period of the next minute up to a week is defined as short term load forecasting. It is required for generation commitment and dispatching as well as assisting Power System Operations engineers with the analysis of network contingencies. Conventional methods such as Box Jenkins and Regression Based methods were previously applied to short term load forecasting. It was found that these methods were unable to adapt to the dynamics of a power system resulting in large forecasting errors. Computational Intelligence techniques were then developed to improve on the inefficiencies of the conventional methods. This paper presents two day ahead short term load forecasting models using an artificial neural network and an Elman recurrent neural network. Weather and non-weather models are developed for comparative purposes. The models are then applied to actual data obtained from a power utility in South Africa.

KEY WORDS
artificial neural network, computational intelligence, elman recurrent neural network

1. Introduction
A short term load forecast ranges from minutes up to a week. These forecasts are essential components of any energy management system. The forecasts are used by system dispatchers and operation analysts to control and to plan power system operations as well as for power system security studies such as contingency analysis and load management [1, 2, 3].

The accuracy of short term load forecasts can have significant effects on power system operations as the economy of operation and the control of the power system may be quite sensitive to forecasting errors [2]. Large forecasting errors may have an adverse effect on the power system as a forecast can either be overly conservative or overly-risky. Forecasts that exceed the amount of load demanded may result in the start-up of too many units and unnecessarily high levels of reserves whilst forecasts that are too low, may result in failure to provide the necessary spinning and operating reserves as well as not meet the demand that is required by the power system [2]. All these factors can induce heavy economic and operational costs.

Techniques used for short term load forecasting vary from the traditional Multiple Linear regression to the more recent Computational Intelligence (CI) techniques such as Expert Systems, Fuzzy logic and Artificial Neural Network. Traditional load forecasting tools utilised time series models which extrapolated historical load data to predict the future loads. These tools assumed a static load series and retained normal distribution characteristics. Due to their inability to adapt to changing environments and load characteristics, large forecasting errors would result when a deviation between historical load data and present conditions occurred [4].

Computational Intelligence techniques, however, learn and adapt to changing environments and forecast accordingly with less forecasting errors as compared with the traditional forecasting techniques [3].

This paper presents a models developed using artificial neural networks which are feed-forward and recurrent for a day ahead load forecast. Four models are developed (weather & non-weather). All models will be trained and thereafter applied to real data from a power utility. This paper first provides an overview of Artificial Neural Networks (ANN) as well as Recurrent Neural Networks.

2. Artificial Neural Networks
Artificial neural networks are non-linear models that are capable of doing non-linear curve fitting. It was inspired by the way the biological systems of humans such as the brain, process information. The human brain is made up of neurons which are interconnected by dendrites and collects information via this connection. ANN’s are made up of a number of simple and highly interconnected processing elements called neurons [5, 6]. ANN’s learn by example and are configured for particular classes of problems or applications through a learning system [6].
Figure 1: Artificial Neuron with a bias [7]

Figure 1 above illustrates the mathematical model of a neuron with the weight (w) depicting the strength of the connection between the input variable (p) and the output (a). The input (p) together with the adjustable weight is then taken through a transfer function (f) and thereafter an output (a) is produced. A neural network is made up of a number of these neurons which are then interconnected.

Figure 2 illustrates a general feed-forward neural network which is the most commonly applied neural network architecture to short term load forecasting. It comprises of an input vector which would generally contain the inputs made up of historical load data, historical and forecasted weather parameters, day types etc [7]. It contains a hidden layer and then an output layer, usually one output is sufficient; however this can be configured as required.

A training algorithm such as back propagation (BP) is used for ANNs. It is basically a training method which calculates the difference between what the output is and what it was supposed to be, i.e. the reference or target output. The weights are then adjusted according to the errors and then propagated back into the system until the error is minimised [5].

The back propagation algorithm is excellent in its ability to accommodate weather variables and other variables as deemed fit by the engineer. However, the main drawback with this training algorithm is that the training process can become very cumbersome and the method may not always converge [8, 9]. Different proposals of dealing with the convergence problem are presented in [8] where the back propagation algorithm with a momentum factor is said to lead the neural network to converge much faster as well as introducing a new modified total error function within the algorithm.

Another challenge with ANN models lays with the development of the network topology i.e. the number of hidden layers and neurons. These have a great effect on the learning capability of a neural network and the size of the network may also be dependent on the system that the model would be applied on [10].

3. Recurrent Neural Networks

Recurrent neural networks are feedback networks which are a function of both the current inputs as well as the previous output [11]. This paper proposes the use of an Elman Recurrent neural network (ERNN) model to forecast load as a recurrent neural network toolbox was readily available.

Several authors have indicated that the ERNN is more accurate than the various artificial neural networks available such as the multiple layer perceptron, radial basis networks etc [12]. An ERNN is a feed-forward network which has the outputs of the hidden layer connected back to the inputs and is trained using a back-propagation training algorithm [12, 13].

Recurrent ANNs are capacitated to internally encode temporal contexts from their feed-back connections. They evolve as a sequential system and, consequently, can describe a dynamical system evolution in a more efficient way than the feed-forward models [12].

Figure 3 illustrates an Elman RNN which is also known as a simple recurrent network (SRN). The following description of the way the ERNN operates is taken from [14]:

As depicted in the above figure, the outputs of the hidden layer are fed back into through a context layer. These are the only feedback connections in the network and the weights from the hidden layer to the context layer are constant values. All other connections are feed-forward with adjustable weights.
The Elman network has a large depth, low resolution memory, since the context units keep an exponentially decreasing trace of the past hidden neuron output values. In this network, signals are processed in two time steps. During the first step at time $t-1$, signals from the input and context layers, which are fully connected to the hidden layer, are distributed to the hidden layer units. The pattern of activation outputs from the hidden layer are then computed and passed onto the output layer for processing at time $t$. At the same time, the hidden layer outputs are copied back onto a set of context units.

Outputs from the context units then combine together with new input signals on the next cycle to feed the hidden units again at time $t+1$. Thus, the external inputs are being mixed with the previously computed inputs “in context” to give recurrent combinations of transformed inputs to the output layer. The weights on the feedback connections from the hidden to the context layer are fixed, typically as unit valued weights. All other weights learn to encode sequences of input patterns during the training process. The activation functions are non-linear differentiable functions, although the output activation function is normally linear [14].

### 4. Model Development

This section discusses how the forecasting models were developed and tested and thereafter applied to actual data from a South African Power Utility.

#### 4.1 Data Pre-Processing

The data set used for this analysis was obtained from Eskom Distribution. The area of study is a residential area in the province of Kwa-Zulu Natal (KZN). Weather data for the area was obtained from the South African Weather Services. The data sets for both weather and historical load were obtained for the period from 2009 – 2011.

The data was normalised as follows:

$$L_{zi} = \frac{L_{ai}}{L_{max}}$$

where:

$L_{si}$ = scaled load data for day $i$
$L_{ai}$ = actual load for day $i$
$L_{max}$ = maximum load

Equation 1 was used to normalize the load. The weather data was normalised by analysing the temperature data and determining the absolute maximum experienced by the area. It was found that the maximum temperature the area has received was ~39.7°C. Therefore, it was chosen to normalize by dividing by an absolute temperature of 40°C. The humidity index was normalised by dividing by 100.

### 4.2 ANN Architecture

Table 1 below describes the inputs and outputs for both models. The difference between the two is that one considers previous and forecast day temperature as well as humidity.

<table>
<thead>
<tr>
<th>Models</th>
<th>Input</th>
<th>Description</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>1-48</td>
<td>Previous day half-hourly load data</td>
<td>1-48</td>
<td>Forecasted half hourly data</td>
</tr>
<tr>
<td>ANN-w</td>
<td>1-48</td>
<td>Previous day half-hourly load data</td>
<td>49-50</td>
<td>Previous day min and max Temperature</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>51-52</td>
<td>Forecast day min and max Temperature</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>53-54</td>
<td>Previous day min and max Humidity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>55-56</td>
<td>Forecast day min and max Humidity</td>
</tr>
</tbody>
</table>

A Netlab toolbox developed by Ian Nabney was used to design, train and test the ANN models. The toolbox uses a training algorithm called the scaled conjugate gradient back propagation method. Load data

The number of hidden layer neurons was determined by trial and error by looking at the topology which provides a minimum training error. Table 2 shows the chosen networks which will be used to evaluate the performance of the models.
Table 2: Final ANN architectures

<table>
<thead>
<tr>
<th>Models</th>
<th>Inputs</th>
<th>Hidden Neurons</th>
<th>Output</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>48</td>
<td>10</td>
<td>48</td>
<td>0.351</td>
</tr>
<tr>
<td>ANN-w</td>
<td>56</td>
<td>30</td>
<td>48</td>
<td>0.235</td>
</tr>
</tbody>
</table>

4.3 ERNN Model

Table 3 below depicts the two models designed for the purpose of this forecast study. RNN is based on historical load and RNN-w takes in an extra 8 inputs which will account for the weather sensitive aspect of the data. The aim is to determine whether the addition of these variables will assist in improving forecasting accuracies.

<table>
<thead>
<tr>
<th>Models</th>
<th>Input</th>
<th>Description</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERNN</td>
<td>1-48</td>
<td>Previous day half-hourly load data</td>
<td>1-48</td>
<td>Forecasted half hourly data</td>
</tr>
<tr>
<td>ERNN-w</td>
<td>1-48</td>
<td>Previous day half-hourly load data</td>
<td>1-48</td>
<td>Forecasted half hourly data</td>
</tr>
<tr>
<td></td>
<td>49-50</td>
<td>Previous day min and max Temperature</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>51-52</td>
<td>Forecast day min and max Temperature</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>53-54</td>
<td>Previous day min and max Humidity</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>55-56</td>
<td>Forecast day min and max Humidity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.3.1 Training of ERNN Models

The number of hidden neurons for both models was chosen based on trial and error method. From the literature surveyed, most authors do not prescribe a method to choose the number of hidden neurons that can be used in a model however; the final numbers of hidden neurons are based on the time it takes to converge as well as the mean absolute percentage error (MAPE) or the mean square error (MSE).

The networks were trained by back-propagation algorithm. The following table lists the MSE for the various numbers of hidden neurons. Both models were trained using the same number of hidden layer neurons and the epoch was set to 2500.

It can be seen from table 4 that the ERNN-w model performed better with 5 hidden neurons. The best performing network for both models was then chosen for the simulation phase. In this instance, both models performed better with 5 hidden neurons.

Table 4: Performance Evaluation of the Models

<table>
<thead>
<tr>
<th>Models</th>
<th>Inputs</th>
<th>Hidden Neurons</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERNN</td>
<td>48</td>
<td>5</td>
<td>0.0138</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.0154</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.0336</td>
<td></td>
</tr>
<tr>
<td>ERNN-w</td>
<td>56</td>
<td>5</td>
<td>0.0120</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.0225</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.0437</td>
<td></td>
</tr>
</tbody>
</table>

5. Performance Evaluation

The performance of the models was evaluated using the Mean Absolute Percentage Error (MAPE) and is calculated as follows:

\[
\text{MAPE} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{L_{ai} - L_{fi}}{L_{ai}} \right| \times 100
\]

Where N is the total number of half hourly load points in a day, \( L_{ai} \) is the actual load at point \( i \) and \( L_{fi} \) is the forecasted load at point \( i \) in a day.

A forecast for Tuesday the 19\(^{th}\) of July as well as Saturday the 23\(^{rd}\) of July 2011 was conducted using the selected models. The following figures illustrate the performance of the models.

Figures 4 and 5 illustrate the forecasts obtained using the ANN models (including the weather sensitive model). It can be seen from these graphs that the model which produced a more accurate forecast was the ANN with the weather sensitive component.
A similar simulation was conducted for the recurrent networks. The following figures demonstrate the results obtained using the two models.

It can be seen that the ERNN-w performed considerably better compared to the network which only considered historical load data thus supporting the hypothesis that the inclusion of weather variables plays a significant role in improving accuracies.

It can be seen from figure 6 above that both networks try to track the curve quite closely however the ERNN-w performs better. Figure 7 also illustrates a forecast for a Tuesday and again, the ERNN-w produces a better result.

Table 5 illustrates the maximum MAPE obtained for each of the models on the selected forecast days. It can be deduced from the table that the model which exhibits the best performance is the ERNN model with a weather sensitive component.

### Table 5: Maximum MAPE for all models

<table>
<thead>
<tr>
<th>Day</th>
<th>ANN</th>
<th>ANN-w</th>
<th>ERNN</th>
<th>ERNN-w</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday</td>
<td>8.2%</td>
<td>6.7%</td>
<td>8.0%</td>
<td>6.1%</td>
</tr>
<tr>
<td>19 July</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td>8.34</td>
<td>7.44</td>
<td>12.2%</td>
<td>3.9%</td>
</tr>
<tr>
<td>23 July</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 6. Conclusion and Future Work

Based on the studies conducted using these models, it can be concluded that the addition of factors that have a great influence on a load is able to assist in improving forecasting accuracies. Weather data, particularly in areas that experience a significant change in weather, needs to be included in a short term load forecast.

However, there is room for more improvement as there are a number of variables in both models; ANN and ERNN, which need to be changed such as the learning rate, momentum factors, number of epochs etc. These elements have a great influence on the results and an optimal value needs to be obtained initially so that the final results can be of an acceptable accurate level.

The advantages of the ERNN model over the ANN model is that the computation time required is reduced because of the small number of hidden neurons whereas the ANN model used a significant number. It can thus be deduced that the ERNN is an architecture which requires more in-depth research regarding a practical approach in implementing it in a power system environment.

Further work regarding the optimization of weights in the models is required. This can be done by using other computational intelligence techniques such as genetic algorithms, particle swarming optimization fuzzy logic methods etc. It is the view of the author that the use of these methods in conjunction with the neural networks would improve forecasting accuracies.
these methods in conjunction with the neural networks would improve forecasting accuracies.

References


