01 Jan 1994

A Hierarchial Neural Network Implementation for Forecasting

B. Fulkerson

M. A. Ozbayoglu

Cihan H. Dagli
Missouri University of Science and Technology, dagli@mst.edu

A HIERARCHIAL NEURAL NETWORK IMPLEMENTATION FOR FORECASTING

Murat A. Ozbayoglu, Cihan H. Dagli
Engineering Management Department
University Of Missouri-Rolla
Rolla, MO 65401

Bill Fulkerson
Deere & Company
Moline, IL 61625

ABSTRACT

In this paper, a hierarchial neural network architecture for forecasting time series is presented. The architecture is composed of two hierarchial levels using a maximum likelihood competitive learning algorithm. The first level of the system has three experts each using backpropagation and a gating network to partition the input space in order to map the input vectors to the output vectors. The second level of the hiearchial network has an expert using Fuzzy ART for producing the correct trend coming from the first level. The experiments show that the resulting network is capable of forecasting the changes in the input and identifying the trends correctly.

INTRODUCTION

The Problems in Forecasting with Time Series

Forecasting, an industrial activity and an inherent part of business has been done mainly by intuitive methods until the past few decades. An attempt to use scientific methods as an aid to make business decisions is an essential part of the present revolution in this field. The four steps involved in applying scientific methods to forecasting problems, data collection, data reduction, model construction and model extrapolation[1] are the building blocks for statistical forecasting.

There are many problems associated with the construction of models to be used in forecasting. Most of the time, it is not easy, if not impossible, to find a model which can fit to the situation with a minimum error. Generally, the model will have a form of

$$x_t = \alpha + \beta t + \epsilon_t \qquad (1)$$

where α and β are constants, β being the slope of the line and α the intercept with the x-axis, and ϵ_t is a random variable generally having a normal probability distribution. This is a standard regression model with time series data. The unpredicted behavior of the input data might bring the problem of model fitting, forcing the model to contain nonlinearity. Also in the extrapolation step, it is assumed that the situation is stable. The chances of a structure remaining stable decrease with the consideration of longer periods. Another problem is that even there is a very good model for the data which has given excellent forecasts, there is no guarantee that this model will continue to give good forecasts in the future.

The model described above is *essentially deterministic* in the sense that the extrapolation follows a linear trend for a given input. It has a deterministic part which describes the linear relation without variance and a stochastic part having some random variation. In a more realistic case, we will have

$$x_t = \alpha + \beta x_{t-1} + \epsilon_t \qquad (2)$$

Such a model is an example of a *stochastic* model. This is still a linear model. Problems of estimating parameters and forecasting tend to be simple for linear models. When nonlinear models are used, there will be extrapolation problems, so it will be difficult to make estimation and forecasting.

Another problem is the occurance of the unpredictable changes in the input pattern, which makes it impossible for a simple model to follow. In this case, a local model, rather than a global model, can be used for the system. There is no difference in the mathematical or statistical formulation of these two types of model, global or local. But often it is the case that, using models as local approximations leads to better forecasts than are obtained from their global use.

Previous research in the area of neural networks for curve fitting[2], simulating a stochastic function for multiple linear regression[3], forecasting power system peak loads[4] and approximation of polynomials[5] indicate the uses of artificial neural networks in this field.

In this paper, situations with one time series variable to be forecasted (denoted Y_t), and one or more time series variables (denoted $X_{1,t}, X_{2,t}, \ldots$) that might help to forecast the future movements or explain the past movements of Y_t are considered. So Y_t will be called the output, and $X_{i,t}$'s will be called the inputs. Inputs can be stochastic or deterministic. It is also assumed that observations of the various series occur at equally spaced time intervals, and while the output may be affected by the inputs, the inputs are not affected by the output. This type of structure is defined by *single-equation* models[6].

Due to the potential problems in estimating a simple regression model with time series data, a new approach using artificial neural networks is presented. In this case, instead of using a simple model between input and output, a hierarchial neural network architecture is used for the forecasting and the trend analysis. The details of the proposed architecture and the algorithms used are explained in the following sections.

Hierarchial Neural Networks

Hierarchial Neural Networks (presented by Jacobs, Jordan, Nowlan & Hinton, 1991)[7] is a modular neural network architecture in which a number of "*expert networks*" compete to learn a set of training data. As a result of this competition, the architecture adaptively divides the input space into regions, and learns separate associate mappings within each region[8]. The appropriate internal structure of the expert modules consist of a neural network architecture such as backpropagation and a gating network to classify the experts which model the input-output functions within their respective regions. The overall architecture with two experts for one hierarchial level is shown in Figure 1.

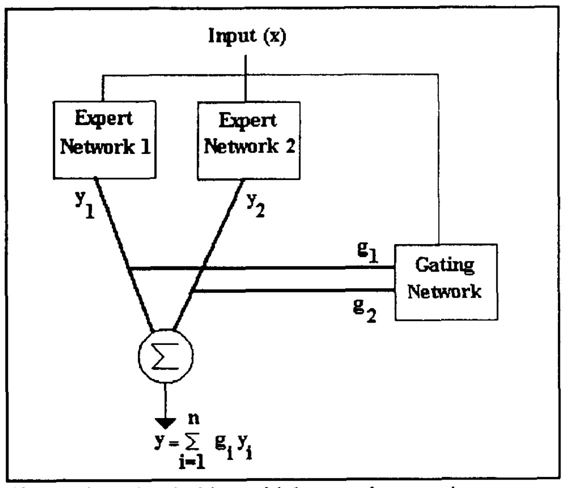


Figure 1 A simple hierarchial neural network

As seen from Figure 1, the inputs for the expert networks, and the gating network are the same. Inside the experts, the inputs are passed through a backpropagation network, thereby producing an output vector y_i, where i denotes the expert number. Then these outputs are crossed with the gating network outputs, and summed forming the final network output. The scenario can be described by the following equation.

where

y = output vector of the architecture

y_i = output vector of ith expert network

g_i = activation of ith output unit of gating network

n = number of expert networks

$$y = \sum_{i=1}^{n} g_i y_i \tag{3}$$

Expert and gating networks are trained simultaneously using the backpropagation algorithm to maximize the cost function

$$J = \ln \sum_{i=1}^{n} g_i e^{-\frac{1}{2}(y^* - y_i)^T (y^* - y_i)} \tag{4}$$

where

y^* = target output vector

y_i = output vector of ith expert network

g_i = activation of ith output unit of gating network
n = number of expert networks

The gating network, itself is composed of a simple three layer backpropagation algorithm, but in the output layer a *softmax* function is used in the nodes (Bridle, 1989)[9].

$$g_i = \frac{e^{s_i}}{\sum_j e^{s_j}} \tag{5}$$

The softmax function is normalized in the sense that

$$\sum_{i=1}^{n} g_i = 1 \tag{6}$$

In the hidden layer of the gating network, and in the hidden and output layer of expert networks, sigmoidal output function is used. The choice of the functions and the number of layers in the system depends on the particular application used.

Training in Hierarchial Networks

The cost function J in equation (4) is a log mixture density. Therefore maximization of J is a maximum likelihood estimation. When the training patterns are generated by different regressive processes, the output is selected with probability g_i, where g_i is the conditional priori probability that the ith expert network generated the current training pattern.

The posterior probability associated with the ith expert network is defined as

$$h_i = \frac{g_i e^{-\frac{1}{2}(y^* - y_i)^T (y^* - y_i)}}{\sum_j g_j e^{-\frac{1}{2}(y^* - y_j)^T (y^* - y_j)}} \tag{7}$$

Differentiating the cost function J equation (4) with respect to y_i gives

$$\frac{\partial J}{\partial y_i} = h_i (y^* - y_i) \tag{8}$$

This is used inside the error term in the expert network outputs, so the resulting error is backpropagated to the hidden layers, and the weights are updated according to that. Equation (8) implies that expert network's weights is updated in proportion to the posterior probability that it generated the current training pattern.

Differentiating the cost with respect to S_i, where S_i is the weighted sum of the inputs to the gating network's ith output unit, gives

$$\frac{\partial J}{\partial S_i} = h_i - g_i \tag{9}$$

This implies that, during training the prior g_i moves toward the posterior probability that the ith expert network generated the current training pattern.

This type of training results in a maximum likelihood competitive learning[10]. All output nodes are competing to win the pattern, but in addition to the winner, all of the losers are also updated in this type of learning. This will be a "*soft*" competition, all competitors are updated but the amount of update is proportional to each competitor's performance. The training finishes when the sum of the errors in the output of the networks is reduced to an acceptable level.

Fuzzy ART

Fuzzy ART model (introduced by Carpenter, Grossberg and Rosen)[11] is a modification of ART neural network families which can categorize input patterns. ART1 (Carpenter and Grossberg, 1987) categorizes binary inputs, whereas ART2 (Carpenter and Grossberg, 1987) can categorize both binary and analog input patterns.

Fuzzy ART can classify both analog and binary input patterns, and it is different from ART 2 in the sense that it replaces of the intersection operator ($\cap$) in ART 2 by the *min* operator (Λ) used in the fuzzy set theory. Due to the fuzziness in the systems, the output categories will overlap, and the amount of overlapping can be controlled by a dimensionless parameter called *vigilance* (ρ). The general algorithm for a Fuzzy ART network is as follows.

Input for the Fuzzy ART is an M-dimensional vector ($I_1,\ldots,I_M$), where each component I_i is in the interval [0,1]. At a preprocessing stage the input vectors are normalized. A normalization procedure called *complement coding* is generally used in Fuzzy ART networks. Complement coding creates a complement of the input vector, and doubles the input vector size by adding the complement vector right after the input vector. It is complemented in such a way that the summation of any element in the input vector and it's complement is M.

If there are N categories and each category j corresponds to a vector $\underline{W}_j = (W_{j1},\ldots,W_{jM})$ of adaptive weights, the Fuzzy ART Long Term Memory (LTM) weight matrix will be an (NxM) matrix, which is initially

$$W_{j1} = \ldots = W_{jM} = 1 \qquad (10)$$

When a category j is selected, the corresponding row in the LTM matrix (W_j) will be updated according to the input vector.

The category j is selected according to the choice function T_j which is defined by

$$T_j = \frac{|I \wedge W_j|}{\alpha + |W_j|} \qquad (11)$$

where
I is the complement coded input vector of size 2M
W_j is the j_{th} row of the LTM weight matrix
α is the choice parameter
Λ is the fuzzy AND operator which is

$$(x \wedge y)_i = \min(x_i, y_i) \qquad (12)$$

and $|x|$ is defined by

$$|x| = \sum_{i=1}^{M} |x_i| \qquad (13)$$

When choice function is applied for all the categories, the T_j which has the maximum value is chosen. If more than one T_j is maximal, the category j with the smallest index is chosen.

The chosen category is passed through the match function, and if the output is greater than the vigilance parameter ρ, the *resonance* occurs. This is shown below :

$$\frac{|I \wedge W_j|}{|I|} \geq \rho \qquad (14)$$

In this case the weight matrix is updated. If the above criteria does not hold *mismatch reset* occurs, and the selected T_j is set to -1 and stays until there is an update in the weight matrix. A new maximum is chosen using eqn (1.13). The search continues until the chosen j satisfies equation (14).

When resonance occurs, the jth row of the weight matrix is updated as follows :

$$W_j^{(new)} = \beta(I \wedge W_j^{(old)}) + (1 - \beta)W_j^{(old)} \qquad (15)$$

where
β is the learning rate.

For fast learning β is set to 1.

According to the choices of vigilance parameter and the number of categories, the ouput category space will cover the fuzzy unit cube.

IMPLEMENTATION

The system used to forecast input trend is composed of a two level hierarchial network. In the first level, a three expert and a gating network system is used. All of the four networks use backpropagation. The outputs of the experts are crossed with the gating network outputs, then summed and the resulting vector is the output for the first layer.

Each expert and the gating network has 5 input neurons, 50 hidden layer neurons, and 3 output neurons. The input to the system is coming from a file in which the trend graph is stored. The input is obtained by putting a 5x5 character size picture on the graph and converting the picture into normalized input points. 5 input points are used for cathching the trend for the particular time (window). The input points are normalized before being entered to the system. The second set of data points are obtained by moving the window one character size to the right (increase in time), and applying the process again. By this method, it is possible to convert any graph into input data for the system.

In the expert networks sigmoidal function is used in the neurons. However, in the output layer of the gating network, the softmax function is used. There are 3 ouput neurons in each expert. So each expert will produce an output vector having 3 elements. After crossed with gating network outputs and summed, the final output (y) for the first hierarchial level is found. It is a 3-element vector. The first element in the output will be high when there is a decreasing trend, the second will be high when the trend is stable and the third element will be high when there is an increasing trend. 45 data sets are used for training the first level (backpropagation). The training is continued until the error is dropped below 0.02.

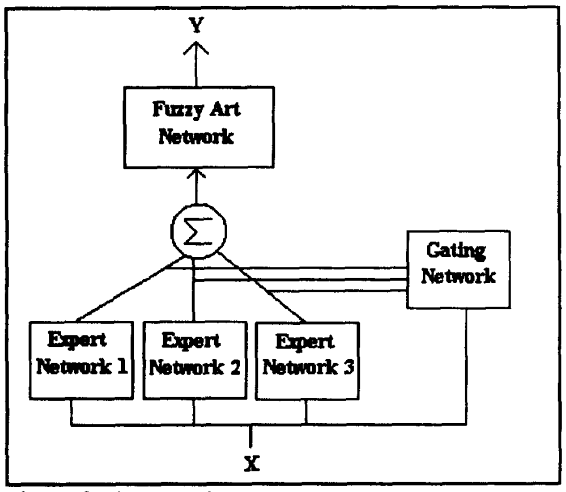


Figure 2 The overall system

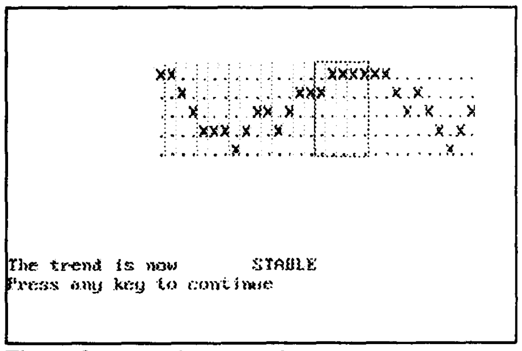
After backpropagation is trained, the fuzzy ART categorization starts. The output vectors of the 45 data training sets are first fuzzified and then inserted into the fuzzy ART network. The system generates 3 categories, and puts the inputs into these categories. With trial and error, it is found that choosing a vigilance parameter of 0.5 results the correct choice for categories. The output is defuzzified again. The overall two layer system is shown in Figure 2. (The fuzzification and defuzzification sections are not shown explicitly in the figure. They are inside the Fuzzy Art network.

When the training is finished, any test data can be sent to the system. A test data is created for this purpose. A particular moment during the test is shown in Figure 3. The input graph, the data window for that particular time and the final output can be seen in the same figure.

The results indicate that the implementation is capable of catching the trend at that moment, so it can guess the next data. When the input size gets bigger, or there are more than one time series, it will be necessary to use more hidden neurons, another hidden layer or more experts.

The reason for using fuzzy Art in connection with the backpropagation is to make it easier for the system to differentiate between similar inputs. Since backpropagation is a good generalizer, it is quite possible that when two similar input patterns are presented, they will generate the same output even if the desired outputs are different. However, when the first level output is feeded to the fuzzy Art, it will categorize them correctly.

Most of the time, the gating network outputs choose one of the experts for producing the output. But when the trend is fluctuating or unpredictable at that moment, the gating network can choose two experts to produce the outputs, but still there is a difference in the ratios. This is because of the fact that the trend is generally closed to one end, so the expert producing this kind of trend will be given more chance by the gating network.



```
The trend is now        STABLE
Press any key to continue
```

Figure 3 The testing example

CONCLUSION

In this paper, a hierarchial neural network for time series forecasting is presented. The network is capable of recognizing different trends in the input data. The existing system can be modified to predict more than one series at a time, or more complex trends, or larger input data size. The architecture used can be extended for higher hierarchial levels.

Hierarchial neural networks can be used in these type of areas where the input patterns are not certain and the input-output relation is not very easy to implement. By using both backpropagation and fuzzy Art in this implementation, specialization problem due to the similar inputs is solved without losing the generalization.

REFERENCES

1. Gilchrist, W., (1976) Statistical Forecasting, John Wiley & Sons

2. Wu, F.Y., (1993) Neural Networks for curve fitting data to second order differential equations, Intelligent Engineering Systems Through Artificial Neural Networks, Volume 3, pp 619-624, ASME Press, New York

3. Kilmer, R.A., Smith, A.E., (1993) Using Artificial Neural Networks to approximate a discrete event stochastic simulation model, Intelligent Enginnering SYstems Through Artificial Neural Networks, Volume 3, pp 631-636, AMSE Press, New York

4. Khotanzad A., Abaye, A. (1993) Forecasting power system peak loads by an adaptive neural network, Intelligent Engineering Systems Through Artificial Neural Networks, Volume 3, pp 891-896, ASME Press, New York

5. Carpenter, W.C., (1993) Are neural nets the best approximators ?, Intelligent Engineering Systems Through Artificial Neural Networks, Volume 3, pp 909-914, ASME Press New York

6. Pankratz, A., (1991) Forecasting with Dynamic Regression Models, John Wiley & Sons

7. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., & Hinton, G.E., (1991) Adaptive mixtures of local experts. Neural Computation, 3, pp 79-87.

8. Jordan, M.I., Jacobs, R.A., (1992) Hierarchies of Adaptive Experts, Advances in neural information systems 4, Morgan Kaufmann Publishers, pp 985-992.

9. Bridle, J. (1989) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Fougelman-Soulie, F. and Herault, J., editors, Neuro-computing: algorithms, architectures and applications. Springer-Verlag

10. Nowlan, S.N., (1989) Maximum Likelihood Competitive Learning, Advances in neural information processing systems 2, M. Kauffman Publishers, pp 574-582.

11. Carpenter, G.A., Grossberg S., Rosen, D.B., (1991) Fuzzy ART : Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System, Neural Networks, Vol. 4, pp 759-771, Pergamon Press.