15th Triennial World Congress, Barcelona, Spain

HYBRID NEURAL MODELS FOR TIME-SERIES FORECASTING

Sinha, M., Gupta, M.M. and Nikiforuk, P.N.

Intelligent Systems Research Laboratory
College of Engineering, University of Saskatchewan
57 Campus Drive, Saskatoon, Saskatchewan, S7N 5A9, CANADA
Email: guptam@sask.usask.ca

Abstract: Three new hybrid neural models which are based upon the basic neural model put forth by McCulloch and Pitts (Haykin, 1999) and the compensatory neural models by Sinha *et al.* (2000), (2001) are proposed in this paper. The basic neural and the compensatory neural models are modified to take into account any linear dependence of the outputs on the inputs. This makes the hybrid models suitable for the solution of some complex problems such as chaotic nonlinear time-series and more simple problems such as linear time-series. These models are verified using a simulation example. It is shown that the hybrid neural models are superior to the basic neural model and the compensatory neural models for time-series forecasting problems. *Copyright ©2002 IFAC.*

Keywords: neural models, neural network, time-series analysis, hybrid.

1. INTRODUCTION

Attempts have been made to solve problems associated with systems whose dynamics may involve both linear and nonlinear relations between the inputs and outputs using basic neuron (BN) based neural networks, and often these neural models require large numbers of connections and neurons. Moreover, the results of predictions on the validation set may not be very encouraging if the dynamical problem is chaotic. Wavelet neural networks have been suggested as being able to overcome these problems (Yamakawa *et al.*, 1994). However, wavelet neural networks may not be very efficient as they lead to significant increases in the computational burden (Sinha *et al.*, 2001), whereas their prediction accuracy is only comparable to that of the basic/compensatory neurons based neural networks. To overcome these deficiencies, it is proposed in this paper to include a linear term in the basic and the compensatory neural models to accomodate the possibility of any linear relationships between the inputs and outputs. These neural networks are named by prefixing the word "hybrid".

Three hybrid neural models are presented in this paper and are tested using a time-series forecasting problem. These three neural models are called hybrid basic neuron (HBN), hybrid compensatory neuron -1 (HCN-1), and hybrid compensatory neuron -2 (HCN-2).

2. NEURAL MODELS

A very brief review of the basic and the compensatory neural models will now be presented followed by the development of the three new hybrid neural models, which have been found to be suitable in dealing with time-series forecasting problems.

The hybrid basic neuron: Since the basic neuron (BN) described by McCulloch and Pitts is well known, only its modification to a hybrid basic neuron (HBN) will now be described. In HBN, the output y is a function of the weighted sum of the neural inputs and is described as

$$u = \sum_{i=0}^{N} w_i x_i \tag{1}$$

$$y = \phi(u) + u \tag{2}$$

$$\phi(u) = \gamma \tanh(\lambda u) = \gamma \frac{\left(\exp^{\lambda u} - \exp^{-\lambda u}\right)}{\left(\exp^{\lambda u} + \exp^{-\lambda u}\right)} \tag{3}$$

where λ is a steepness factor, γ is a gain constant, w_i is the i^{th} neural weight, x_i is the i^{th} neural input, x_0 is the bias, and N is the number of the neural inputs. This HBN is depicted in Fig. 1.

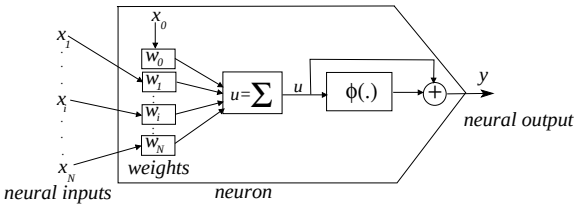

Fig. 1. Hybrid basic neuron (HBN)

The compensatory neurons -1 & 2 (CN -1 & 2): Sinha *et al.* (2000) have proposed a compensatory neuron -1 (CN-1) where each neuron incorporates two nonlinearities as shown in Fig. 2. This neural model

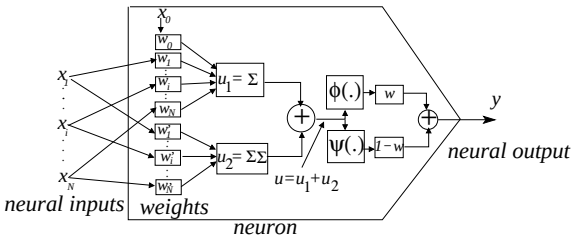

Fig. 2. Compensatory neuron -1 (CN-1)

was found to be efficient for solving functional mapping and classification problems, as well as time-series forecasting problems. The compensatory neuron -1 is defined as

$$u = \sum_{i=0}^{N} w_i x_i + \sum_{j=1\, j\neq i}^{N} \sum_{i=1}^{N} w_i' w_j' x_i x_j \tag{4}$$

$$y = w\phi(u) + (1-w)\psi(u) \tag{5}$$

$$\psi(u) = \gamma \arctan(\lambda u) \tag{6}$$

where $\phi(u)$ is defined in Eqn. (3). $\phi(u)$ is different from $\psi(u)$, and this allows classification into two classes using only one neuron, e.g., in the four-bit parity problem.

Later, Sinha *et al.* (2001) proposed a compensatory neuron -2 (CN-2) where each neuron incorporates one nonlinearity. The compensatory neuron -2 is achieved by setting $w = 1$ in Eqn. (5), and is depicted in Fig. 3.

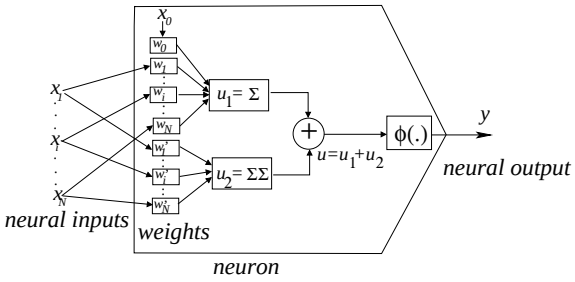

Fig. 3. Compensatory neuron -2 (CN-2)

The hybrid compensatory neurons -1 & 2 (HCN-1 & 2): In the hybrid compensatory neural models, which are modifications of the compensatory neural models, the weighted sum of the inputs $\left(\sum_{i=0}^{N} w_i x_i\right)$ is added to the outputs of the nonlinear functions.

HCN-1 is defined as

$$y = w\phi(u) + (1-w)\psi(u) + \sum_{i=0}^{N} w_i x_i \tag{7}$$

and is shown in Fig. 4, where u is defined in Eqn. (4), and $\phi(u)$ and $\psi(u)$ are given by Eqns. (3) and (6) respectively.

HCN-2, on the other hand, is defined as

$$y = w\phi(u) + \sum_{i=0}^{N} w_i x_i \tag{8}$$

where u and $\phi(u)$ are defined by Eqns. (4) and (3) respectively. This neuron is depicted in Fig. 5.

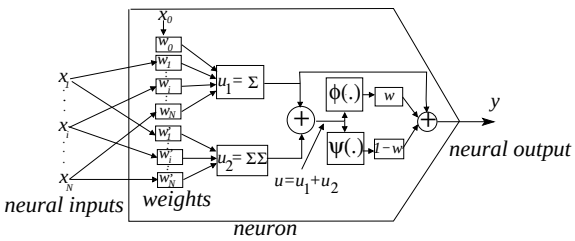

Fig. 4. Hybrid compensatory neuron -1 (HCN-1)

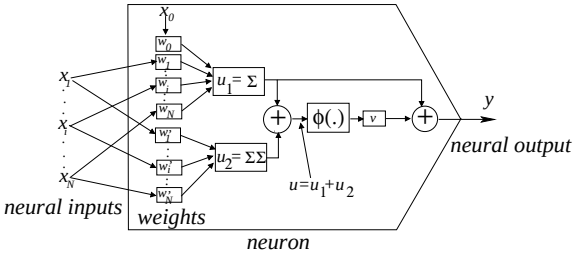

Fig. 5. Hybrid compensatory neuron -2 (HCN-2)

Various architectures: It is the intent of this paper to compare the performance of the three new hybrid neural models HBN, HCN-1 and HCN-2 with the performance of their respective original models. The neural network architectures used for this purpose have an

output layer that is composed of only a summation function and the nomenclatures used for these networks are as follows:

- Basic neural network architecture (BNNA) based on BN;
- Hybrid basic neural network architecture (HBNNA) based on HBN, Fig. 1;
- Compensatory neural network architecture-1 (CNNA-1) based on CN-1, Fig. 2;
- Compensatory neural network architecture-2 (CNNA-2) based on CN-2, Fig. 3;
- Hybrid compensatory neural network architecture-1 (HCNNA-1) based on HCN-1, Fig. 4;
- Hybrid compensatory neural network architecture-2 (HCNNA-2) based on HCN-2, Fig. 5.

3. LEARNING ALGORITHM

If in a neural network model the output layer has a summation function only, then the output layer neuron-weights can be updated using either a linear scheme such as matrix inversion, a singular value decomposition approach or the usual backpropagation algorithm. If the weights are updated using the backpropagation algorithm in conjunction with scaled conjugate gradient learning (SCG) (Moller, 1993), this then constitutes self-scaling scaled conjugate gradient learning (SSCG) (Sinha *et al.*, 2000) (for off-line learning). It has been shown (Sinha *et al.*, 2000) that this method gives better accuracy for functional mapping and classification problems. Here, all the computations are done in off-line mode and the error gradients for all the patterns are obtained by summing and averaging the error gradients for the individual patterns. The equations for calculating the gradient of the error function, described in Eqn. (9), for HCNNA-1 only are given below.

$$E(\tilde{w}(n)) = \frac{1}{2}\sum_{k=1}^{K}(y_d(k) - y(k))^2 \tag{9}$$

where $E(\tilde{w}(n))$ is the error at the n^{th} iteration, $y_d(k)$ is the desired output, $y(k)$ is the actual neural output, K is the number of neurons in the output layer, $\tilde{w}$ is a vector containing all the weights in the neural network, and n is the iteration number.

Gradient calculation: The error gradient calculation for the hybrid compensatory neural network architecture -1 (HCNNA-1) is as follows

*Gradient calculation for output layer of HCNNA-1***:**

$$\frac{\partial E(n)}{\partial w_{kj}} = -(y_d(k) - y(k))y_j \tag{10}$$

$$y_j = w_{jj}\phi_j(u_j) + (1.0 - w_{jj})\psi_j(u_j) + \sum_{i=0}^{N} w_{ji}x_i \tag{11}$$

$$u_j = \sum_{i=0}^{N} w_{ji}x_i + \sum_{i=1_{h\neq i}}^{N}\sum_{h=1}^{N} w'_{ji}w'_{jh}x_ix_h \tag{12}$$

where $\phi_j(u_j)$ is given by Eqn. (3) and $\psi_j(u_j)$ is given by Eqn. (6), the subscript j denotes the neuron number. w_{kj} is the weight from j^{th} neuron to the k^{th} neuron in the output layer, w_{ji}, w'_{ji} are the weights from the i^{th} input to the j^{th} neuron, w_{jj} is the weight used in the j^{th} compensatory/hybrid compensatory neuron, and y_j is the output of the j^{th} neuron.

Gradient calculation for input layer and neuron blocks of HCNNA-1*:*

$$\frac{\partial E(n)}{\partial w_{jj}} = \delta_j\left(\phi_j(u_j) - \psi_j(u_j)\right) \tag{13}$$

$$\frac{\partial E(n)}{\partial w_{ji}} = x_i\left(\delta^1_{y_j}w_{jj} + (1 - w_{jj})\delta^2_{y_j} + \delta_j\right) \tag{14}$$

$$\frac{\partial E(n)}{\partial w'_{ji}} = 2.0x_i\left(s_j - x_iw'_{ji}\right)\left(\delta^1_{y_j}w_{jj} + \delta^2_{y_j}(1 - w_{jj})\right) \tag{15}$$

$$\delta^1_{y_j} = \delta_j\frac{\partial\phi_j}{\partial u_j} \tag{16}$$

$$\delta^2_{y_j} = \delta_j\frac{\partial\psi_j}{\partial u_j} \tag{17}$$

$$\delta_j = \sum_{k=1}^{K} -(y_d(k) - y(k))w_{kj} \tag{18}$$

$$s_j = \sum_{i=1}^{N} x_iw'_{ji} \tag{19}$$

4. SIMULATION STUDIES AND DISCUSSION

While the essential components defining a neural network are its topology, size, functionality, learning algorithms, training/validation, and implementation, the main factors which will decide the superiority of neural models, in general, using supervised learning, are the computational burden for each iteration/epoch, number of epochs for convergence, NN size and its generalization. The performance measure involves the selection of these features and quantifying, in some form, the success of the selection. The result of the

performance evaluation of a neural network will depend significantly on its applications.

A chaotic nonlinear time-series problem: In this study the following nonlinear time-series equation was used to generate the training and the validation sets to test the efficiency of the proposed neural models.

$$x_{n+1} = \frac{5x_n}{1+x_n^2} - 0.5x_n - 0.5x_{n-1} + 0.5x_{n-2} \quad (20)$$

The initial values were taken as $x_0 = 0.2$, $x_1 = 0.3$, and $x_2 = 1.0$. The neural network consisted of three neural inputs and one neural output. The three neural inputs were comprised of two tapped delays $(x_{n-1},\ x_{n-2})$ and one present-value (x_n). The data set were constructed by deleting the past-past value and adding a new predicted value. A time-series of 101 points was used to construct the training data set, consisting of 99 training patterns.

To compare the performance of the proposed hybrid neural models, the following simulations results were recorded and are discussed:

- Mean square error (MS Error)
- Desired and predicted values
- Prediction error on validation set

These results are presented in Figs. 6 to 10. In Fig. 6, the mean square error for the problem defined in Eqn. (20) during training is presented. All the neural network models used for training consisted of three neurons in the input layer and a summation function only in the output layer.

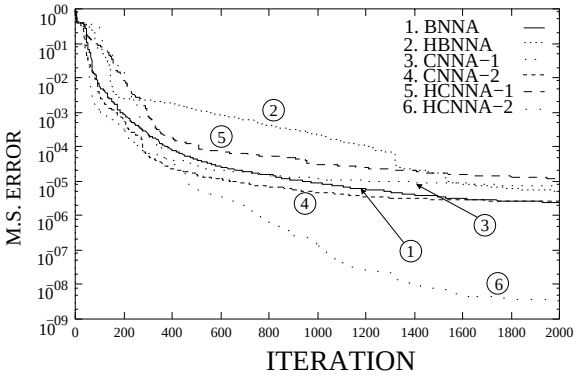

Fig. 6. Mean square error during training for different neural architectures

It can be observed that the convergence was best for HCNNA-2 while those for HBNNA and HCNNA-1 were sluggish as compared to the other models. The best convergence of HCNNA-2 is due to the fact that HCN-2 models higher order nonlinearity by using cross-correlation terms (double summation term in Eqn. (12)). Moreover, the weighted linear inputs added to the output of the neuron made it effective in modeling the linear terms involved in the time-series forecasting problem. The addition of the linearity to the output of the neuron may affect the initial convergence due to tradeoff with the nonlinearity. Unlike HCN-2, HCN-1 used two nonlinearities (usefull for classification problems) which did not help HCNNA-1 to converge more quickly. Similarly, in HBNNA the addition of the linear weighted sum of the inputs to the output of the neuron reduced the rate of convergence as explained above. It may also be noted that the amount of computation involved for the compensatory models was slightly more than that for BNNA and HBNNA.

All of the predicted outputs on the training set were observed to coincide very closely with the desired output for all of the neural models. This implies good learning on the training set and it is for this reason that the plots of these responses are not given in this paper. The prediction results for the validation set are presented in Fig. 7. It is observed that all the models predicted accurately up to the 105^{th} time-instant, however, after the 105^{th} time-instant the nonhybrid models (BNNA, CNNA-1, CNNA-2) diverged. This was because the nonhybrid neural models were not able to model the linear terms involved in Eqn. (20), therefore, could not produce the desired output which was very close to zero at the 106^{th} time-instant thereby leading to divergence.

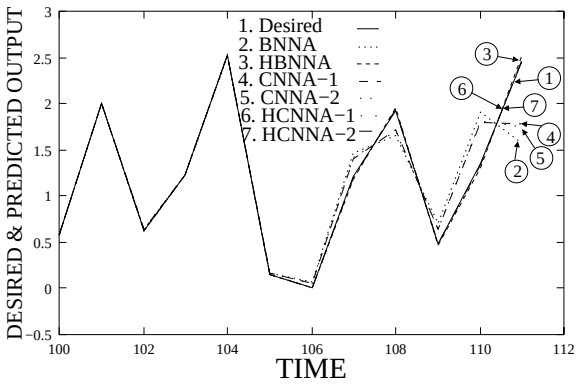

Fig. 7. Desired and predicted output for different neural network models on validation set

On the other hand, the hybrid models (HBNNA, HCNNA-1, HCNNA-2) predicted accurately beyond the 105^{th} time instant as they were able to model linear and nonlinear components of the given time-series equation adequately. This is evident from Fig. 7. It may be noted that HCNNA-2 performed best in terms of both the convergence and the prediction on the validation set. In Fig. 8, the difference between the desired and the predicted values on the validation set is presented for further clarification to show how the error has increased after the 105^{th} time-instant.

In the above discussion all of the models were composed of an equal number of neurons (three). The question which then arose was to what extent the performance of the nonhybrid models would improve if the number of neurons was increased for these. Based on the results already obtained it was decided to com-

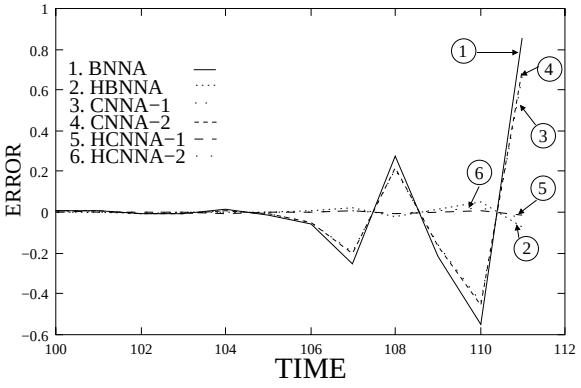

Fig. 8. Prediction error for different neural network models on validation set

pare the performance of HCNNA-2 with a BNNA now employing nine neurons in the input layer. As before, both had only a summation function in the output layer. A comparison of the mean square error decay during training is shown in Fig. 9, and the desired and predicted values in Fig. 10.

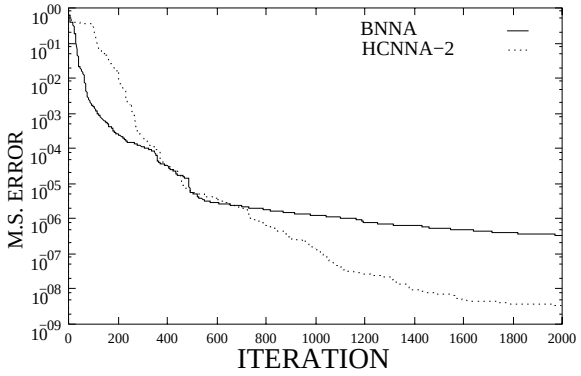

Fig. 9. Mean square error during training for BNNA and HCNNA2

It is evident from Fig. 9 that the increase in the number of neurons in BNNA resulted in some improvement in the convergence, but its convergence was poor compared to HCNNA-2. Moreover, the increase in the number of neurons in the case of BNNA made it more computing intensive than HCNNA-2. In Fig. 10, the prediction results for the validation set are presented. Again it is self evident that increasing the number of neurons in BNNA did not improve the prediction. A small error in prediction at the 106^{th} time-instant rapidly increased afterwards. This clearly indicates that the nonhybrid models were not able to model the time-series equation adequately.

5. CONCLUSIONS

Three new hybrid neural models were presented in this paper. It was shown that the hybrid compensatory neuron -2 which forms the basis of HCNNA-2 performed the best both in terms of convergence and the accuracy of prediction of the time-series. Moreover, all the hybrid models performed better than the

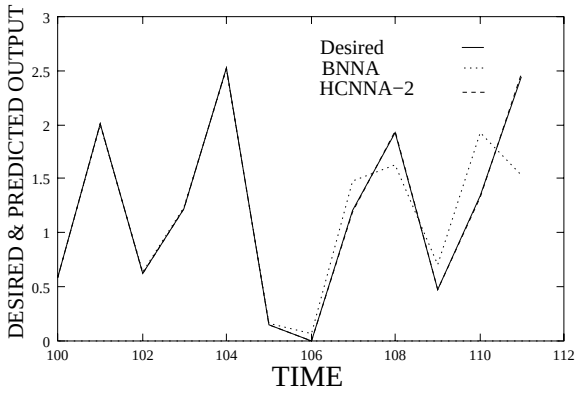

Fig. 10. Desired and predicted output for BNNA and HCNNA-2

nonhybrid models on the validation set. Another advantage of using the compensatory/hybrid compensatory models was that they required fewer neurons and, therefore, they could be trained on a smaller data set properly as compared to the basic neural model which generally required more neurons and thereby demanded a large data set. In fact, it was observed that as the nonlinearity of the system increased the compensatory/hybrid compensatory models became more effective. It is concluded, therefore, that the hybrid models are more suitable for the time-series forecasting problems which may involve both linear and the nonlinear terms.

6. REFERENCES

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall Inc.. Upper Saddle River, New Jersey.

Moller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks* **6**, 525–533.

Sinha, M., K. Kumar and P. K. Kalra (2000). Some new neural network architectures with improved learning schemes. *Softcomputing* **4**(4), 214–233.

Sinha, M., M. M. Gupta and P. N. Nikiforuk (2001). A compensatory wavelet neuron model. *Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver B.C. Canada, July 25- July 28, 2001, paper No. 243, pp. 1372-1376.*

Yamakawa, T., E. Uchino and T. Samatsu (1994). Wavelet neural network employing overcomplete number of compactly supported non-orthogonal wavelets and their applications. *in Proceeding of IEEE International Conference on Neural Networks, June 28 - July 2* pp. 1391–1396.