International Journal of Soft Computing 2 (4): 549-554, 2007
© Medwell Journals, 2007

# A Novel Method of Escape from Local Minima for Elman Neural Network

Zhiqiang Zhang and Zheng Tang
Faculty of Engineering, University of Toyama,
Gofuku 3190, Toyama Shi, 930-8555, Japan

**Abstract:** Eleman Neural Network (ENN) have been efficient identification tool in many areas (classification and prediction fields) since they have dynamic memories. However, one of the problems often associated with this type of network is the local minima problem which usually occurs in the process of the learning. To solve this problem and speed up the learning process, we propose a method to add a term in error function which related to the neuron saturation of the hidden layer for Elman Neural Network. The activation functions are adapted to prevent neurons in the hidden layer from stucking into saturation area. We apply the new method to the Boolean Series Prediction Questions to demonstrate its validity. Simulation results show that the proposed algorithm has a better ability to find the global minimum than back propagation ENN algorithms within reasonable time.

**Key words:** Elman Neural Network (ENN), modified error function, local minima problem, Boolean Series Prediction Question (BSPQ)

## INTRODUCTION

Elman Neural Network (Elman, 1990) is a type of partial recurrent neural network, which consists of two-layer back propagation networks with an additional feedback connection from the output of the hidden layer to its input layer. The advantage of this feedback path is that it allows ENN to recognize and generate temporal patterns and spatial patterns. This means that after training, interrelations between the current input and internal states are processed to produce the output and to represent the relevant past information in the internal states (Stagge and Sendhoff, 1999). As a result, the ENN has been widely used in various fields from a temporal version of the Exclusive-OR function to the discovery of syntactic or semantic categories in natural language data.

However, since ENN uses Back Propagation (BP) to deal with the various signals, it has proven to be suffering from a sub-optimal solution problem (Pham and Liu, 1993; Shi *et al*., 2003). At the same time, for the ENN, it is less able to find the most appropriate weights for hidden neurons and often get into the sub-optimal areas because the error gradient is approximated (Smith, 2004). Furthermore, The efficiency of the ENN is limited to low order system due to the insufficient memory capacity (Adem and Seref, 2006). Therefore, several approaches have been suggested in the literature to increase the performance of the ENN with simple modifications (Gao *et al*., 1996; Chagra *et al*., 1998; Huang *et al*., 2004). Also these improved modifications attempt to add other feedback connections factors to the model which can increase the capacity of the memory in order to enhance the memroy of the network, but it can not avoid the local minima problem essentially. So the local minimum problem still is a serious problem and usually occurs in various applications.

In this study, we explain the neuron saturation in the hidden layer as the update disharmony between weights connected to the hidden layer and output layer from a new angle of view. Then we propose an improved ENN algorithm to help the network to avoid the local minima problem caused by such disharmony. The new proposed method is to increase the memory capacity of the network by finding the essential question and resolving it rather than adding other connection among layers like the references listed above. Besides, the modified error function did not require much additional computation and did not change the network topology. Finally, simulation results are presented to substantiate the validity of the proposed algorithm by comparing with original ENN (Elman, 1990) algorithm and improved ENN algorithm (Pham and Liu, 1993). Since a three-layered network is capable of forming arbitrarily close approximation to any continuous nonlinear mapping (Schreiber *et al*., 1990; Hornik *et al*., 1989) we use three layers for all training networks.

**Corresponding Author:** Zheng Tang, Faculty of Engineering, University of Toyama, Gofuku 3190, Toyama Shi, 930-8555, Japan

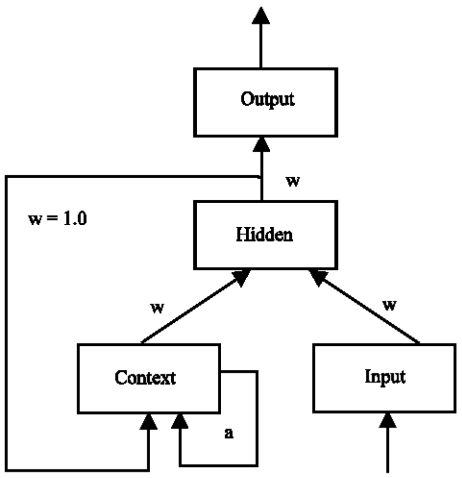

Fig. 1: The structure of the ENN

## ENN'S STRUCTURE

Figure 1 shows the structure of a simple ENN. In Fig. 1, after the hidden units are calculated, their values are used to compute the output of the network and are also all are stored as "extra inputs" (called context unit) to be used when the next time the network is operated. Thus, the recurrent contexts provide a weighted sum of the previous values of the hidden units as input to the hidden units. As shown in Fig. 1, the activations are copied from hidden layer to context layer on a one for one basis, with fixed weight of 1.0 (w = 1.0). The forward connection weight is trained between hidden units and context units as well as other weights. If self-connections are introduced to the context unit when the values of the self-connections weights (*a*) are fixed between 0.0 and 1.0 (usually 0.5) before the training process, it is an improved ENN as proposed by Pham and Liu (1993). When weights (a) are 0, the network is the original ENN.

Figure 2 is the internal learning process of the ENN by the error back-propagation algorithm. From Fig. 2 we can see that training such a network is not straightforward since the output of the network depends on the inputs and also all previous inputs to the network. So, it should trace the previous values according to the recurrent connections (Fig. 3).

Figure 3 shows that a three layers ENN where a back propagation is used to calculate the derivatives of the error (at each output unit) by unrolling the network to the beginning. In Fig. 3, when the next input (time t) is represented, the input source of the hidden layer inclues two parts which are input part from outside and recurrent part from context unit. Moreover, the context units contain values which are exactly the hidden unit values at

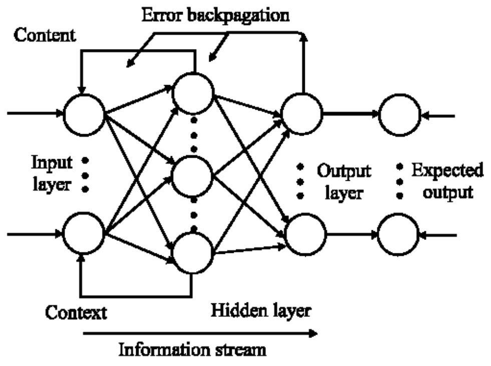

Fig. 2: Internal process analysis of ENN

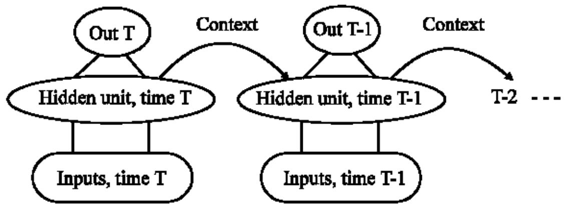

Fig. 3: Unroll the ENN through time

time t-1 and t-2... untill the initial start state. Thus these context units provide the network with memory through tracing all the process of the learning by continuous iteration. Therefore, the ENN network is converted into a dynamical network that is efficient in the use of temporal information of the input sequence, both for classification as well as for prediction (Lawrence *et al*., 2000; Conner *et al*., 1994). However, the calculation of the functional derivatives is not straightforward and it leads the network easily to a sub-optimal situation.

## THE PROPOSED ALGORITHM

In the original ENN, usually the sigmoid function is used to process the network. Our proposed method is based on the same understanding of the current sigmoid function shown as Eq. 1.

$$f(x) = \frac{1}{1+e^{-x}} \tag{1}$$

The derivative of the sigmoid function is shown as Eq. 2.

$$f'(x) = g(1-f(x)) * f(x) \tag{2}$$

Since we use the Sigmoid function, saturation problem is inevitable. Such a phenomenon is caused by the property of the activation function (Cybenko, 1989).

The shape curve of the sigmoid function is shown as Fig. 4. In Fig. 4, we can see there are two extreme areas A and B which are called saturation fields and the scope of the value for the sigmoid function is between 0 and 1.

From Fig. 4 we can see that once the activity level of all hidden layer approaches the two extreme areas A and B (the outputs f'(x) of all neurons are in the extreme value close to 1 or 0), f'(x) will almost be 0. For the ENN, the change in weights is determined by the sigmoid derivative (Eq. 2) which can even be as small as 0. So for some training patterns, weights connected to the hidden layer and the output layer are modified inharmoniously, that is all the hidden neuron's output are rapidly driven to the extreme areas before the output start to approximate to the desired value. Thus the hidden layer will lose their sense to the error. The local minimum problem may occur.

To overcome such a problem, the neuron output in the output layer and those in the hidden layer should be considered together during the iterative update procedure. Motivated by this, we add one term concerning the outputs in the hidden layer to the conventional error function for the BP algorithm. In such way, weights connected to the hidden layer and the output layer could be modified harmoniously. And we apply the algorithm to the ENN to avoid the local minima problem caused by this disharmony.

For the original ENN algorithm, the error function is given by

$$E_A = \sum_{t=0}^{T} E_P \tag{3}$$

Where p indexes over all the patterns for the training set in the time interval (0, T). In our study, the time element is updated by the next input of the pattern from training set. So we can get the follwing equation.

$$E_A = \sum_{t=0}^{T} E_P = \sum_{p=1}^{P} E_p \tag{4}$$

$E_p$ is defined by

$$E_p = \frac{1}{2}\sum_{j}^{J}(t_{pj}^{k} - o_{pj}^{k})^2 \tag{5}$$

Where $t^k_{pj}$ is the target value (desired output) of the j-th component of the output for pattern p and $o^k_{pj}$ is the j-th unit of the actual output pattern produced by the presentation of input pattern p in the time k and j indexes all the output units.

To minimize the error function $E_A$, the ENN algorithm uses the following delta rules as back propagation algorithm:

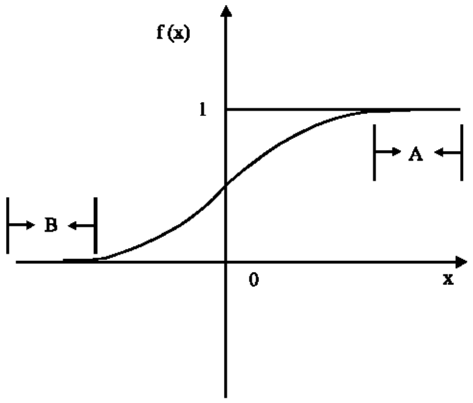

Fig. 4: The curve of sigmoid function

$$\Delta w_{ji} = -\eta_A \frac{\partial E_A}{\partial w_{ji}} \tag{6}$$

Where $w_{ji}$ is the weight connected between neurons i and j and $\cdot_A$ is the learning rate. For the proposed ENN algorithm, the modified error function is given by:

$$\begin{aligned} E_{new} &= E_A + E_B \\ &= \frac{1}{2}\sum_{p=1}^{P}(t_{pj} - o_{pj})^2 \\ &+ \frac{1}{2}\sum_{p=1}^{P}(\sum_{j=1}^{J}(t_{pj} - o_{pj})^2 \times \sum_{j}^{H}(y_{pj} - 0.5)^2) \end{aligned} \tag{7}$$

We can see that the new error function consists of two terms, $E_A$ is the conventional error function and $E_B$ is the added term. Where $y_{pj}$ is the output of the j-th neuron in the hidden layer and H is the number of neurons in the hidden layer.

$$\sum_{j}^{H}(y_{pj} - 0.5)^2 \tag{8}$$

Equation 6 can be defined as the degree of saturation in the hidden layer for pattern p. This added term is used to keep the degree of saturation of the hidden layer small while $E_A$ is large (the output layer have not approximate the desired signals). While the output layer approximates to the desired signals, the affect of term $E_B$ will be diminished and becomes zero eventually. Using the above error function as the objective function, we can rewrite the update rule of weight $w_{ji}$ as:

$$\Delta w_{ji} = -\eta_A \frac{\partial E_A}{\partial w_{ji}} - \eta_B \frac{\partial E_B}{\partial w_{ji}} \tag{9}$$

For pattern p, the derivative $.E_A/.w_{ji}$ can be computed as the same as the conventional error function does. Thus we can easily get $.E_B/.w_{ji}$ as following:

For weights connected to the output layer:

$$\frac{\partial E_B^p}{\partial w_{ji}} = \frac{\partial E_A^p}{\partial w_{ji}} \sum_{j=1}^{J} (y_{pj} - 0.5)^2 \tag{10}$$

For weights connected to the hidden layer:

$$\frac{\partial E_B^p}{\partial w_{ji}} = \frac{\partial E_A^p}{\partial w_{ji}} \sum_{j=1}^{J} (y_{pj} - 0.5)^2 + \sum_{j}^{J} (t_{pj} - o_{pj})^2 (y_{pj} - 0.5) \frac{\partial y_{pj}}{\partial w_{ji}} \tag{11}$$

Because $y_{pj} = f(net_{pj})$ and $net_{pj} = ,w_{ji}o_{pj}$, so

$$\frac{\partial y_{pj}}{\partial w_{ji}} = \frac{\partial y_{pj}}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} = f'(net)o_{pi} \tag{12}$$

Where $o_{pi}$ is the i-th input for pattern p and $net_{pj}$ is the net input to neuron j produced by the presentation of pattern p.

## SIMULATIONS

In order to verify the effectiveness of the modified error function for ENN, we compare its performance with those of the original ENN algorithm (Elman, 1990) and improved ENN algorithm (Pham and Liu, 1993) on a series of BSPQ problems including "11", "111" and "00" problems. In order to maintain the similarity for all algorithms, the learning rate of $,_A=,_B=0.9$ are used in all experiments where as the weights and thresholds are initialized randomly from (0.0, 1.0).

Two aspects of training algorithm performance ("success rate" and "iterative") are assessed for each algorithm. Simulations were implemented in Visual C++ 6.0 on a Pentium4 3.0GHz (1GB)). A training run was deemed to have been successful if the error precision E was smaller than the requested 0.1 or 0.01, For all the trials, 200 patterns were provided to satisfy the equilibrium of the training set and at the same time to ensure that there was enough and reasonable running time for all algorithms. The upper limit iterative were set to 2 0000 for three algorithms.

**The "11" questions:** Boolean Series Prediction Questions is one of the problems about time sequence prediction. First let us see the definition of the BSPQ. Suppose that we want to train a network with an P input and T targets as:

$$P = 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1$$

and

$$T = 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1$$

Here T is defined to be 0, except when two 1's occur in P in which case T is 1 and we called this problem as "11" problem (one kind of the BSPQ ). Also when "00"or "111" (two 0's or three 1's) occurs, it is named as the "00" or "111" problem.

Firstly, we deal with the "11" question and analyze the effect of the memory in the context layer of the network. In this paper we define the prediction set *P* randomly as stated in the 20's figures below.

$$P_1 = 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1$$

And we used the well trained network (error precison is attained) to do the final prediction about sequence $P_1$ to test the prediction capability of it. Table 1 compares the simulation results of the three methods, we can see that our proposed algorithm not only almost 100% succeeded but quickly got the convergence point. Of course, the original ENN was able to predict the requested input test, but the training success rate was slow. Furthermore, it succeeded only 73% when E was set to 0.1. And the improved ENN has increased the ability of the dynamic memorization of the network because of the self-connection weights element (a = 0.5). Although improved ENN could accelerate the convergent of the learning process (time was less than original ENN), it could not essentially avoid the local minima problems because of the characteristics of the gradient descent.

Figure 5 shows the learning characteristics for the three methods with the same initialization weight for the

Table 1: Experiment results for the "11" question with 5 neurons in the hidden layer

| Methods | Success rate(100 trials) | | Average iterative | |
|---|---|---|---|---|
| (1-5-1 network) | E = 0.1 | E = 0.01 | E = 0.1 | E = 0.01 |
| Original ENN | 73% | 63% | 260 | 331 |
| improved ENN | 83% | 82% | 198 | 251 |
| propose ENN | 100% | 99% | 89 | 135 |

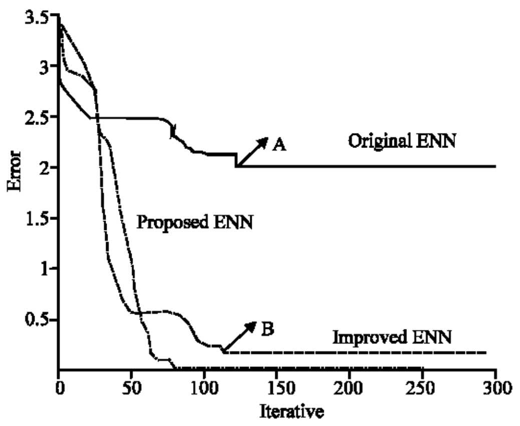

Fig. 5: Training error curve of the three ENN algorithm

network (1-5-1), when E was set to 0.01. From the simulation results we can see that the proposed ENN algorithm only needed about 80 iteration steps to be successful, but the original ENN and the improved ENN algorithm do not have goal value and gets trapped into local minima point A and B, respectively. The improved ENN have accelerated the learning process but it could not escape the local minima problem. However, with our proposed ENN algorithm it could get the convergent point by avoiding the local minima.

For the prediction set $P_1$, we can get its corresponding expected results (for “11” question) with below $T_1$.

$T_1$ = 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1

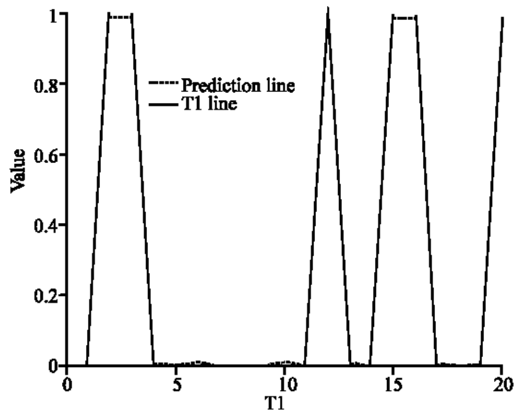

Fig. 6: Expected output $T_1$ and prediction result with proposed ENN

Figure 6 is the simulation prediction result of $P_1$ for the "11" question with our proposed ENN algorithm. In Fig. 6, the two lines represented the $T_1$ line and the prediction results line, respectively. From Fig. 6 we can see that the tolerance for every pattern was less than 0.05. So the network has enough ability to do the prediction of the given task as desired.

For the same problem, as we gradually increased the numbers of the neuron of the hidden layer, the original ENN and improved ENN have encreased the success rate of the learning, however, with our proposed ENN it was able to get more high success rate within less iteration steps by enhancing the search'capacity of the network. The results are shown in Table 2.

Table 2: Experiment results for the “11” question with 7 neurons in the hidden layer

| Methods (1-7-1 network) | Success rate(100 trials) E = 0.1 | E = 0.01 | Average iterative E = 0.1 | E = 0.01 |
|---|---|---|---|---|
| Original ENN | 82% | 70% | 355 | 509 |
| Improved ENN | 91% | 85% | 278 | 454 |
| Proposed ENN | 100% | 97% | 222 | 329 |

**The "111" and "00" questions:** As we change rules of the input sequence, we can continue to testify the validity of our proposed ENN algorithm. The specific parameter set of the network was same as the "11" questions.

Table 3 is the specific comparison results from the "111" question for the three algorithms. From Table 3 we can see that, for the network (1-7-1), the success rate of

Table 3: Experiment results for the “111” question with different neuron units in the hidden layer

| Structure of the network | Items/Methods | Success rate (100 trials) E = 0.1 | E = 0.01 | Average iterative E = 0.1 | E = 0.01 |
|---|---|---|---|---|---|
| 1-7-1 network | Original ENN | 50% | 45% | 789 | 1031 |
| | Improved ENN | 81% | 71% | 623 | 822 |
| | Proposed ENN | 90% | 85% | 370 | 584 |
| 1-10-1 network | Original ENN | 72% | 70% | 669 | 801 |
| | Improved ENN | 91% | 79% | 571 | 633 |
| | Proposed ENN | 100% | 98% | 421 | 530 |
| 1-12-1 network | Original ENN | 81% | 77% | 845 | 1002 |
| | Improved ENN | 93% | 86% | 609 | 799 |
| | Proposed ENN | 100% | 98% | 479 | 700 |

Table 4: Experiment results for the “00” question with different neuron units in the hidden layer

| Structure of the network | Items/Methods | Success rate (100 trials) E = 0.1 | E = 0.01 | Average iterative E = 0.1 | E = 0.01 |
|---|---|---|---|---|---|
| 1-5-1 network | Original ENN | 83% | 76% | 333 | 417 |
| | Improved ENN | 92% | 86% | 220 | 319 |
| | Proposed ENN | 100% | 97% | 93 | 152 |
| 1-7-1 network | Original ENN | 87% | 75% | 493 | 802 |
| | Improved ENN | 93% | 88% | 395 | 643 |
| | Proposed ENN | 99% | 97% | 201 | 305 |
| 1-10-1 network | Original ENN | 90% | 85% | 600 | 887 |
| | Improved ENN | 94% | 92% | 578 | 780 |
| | Proposed ENN | 100% | 97% | 500 | 662 |

the proposed ENN was lower (only 90% when the error criterion was set to 0.1) because of the insufficient capacity with less hidden unit nodes. As the unit nodes of the hidden was increased, our proposed ENN algorithm could almost attain 100% succeess than original ENN and improved ENN algorithms. The proposed ENN has successfully escaped from the local minima problems through adjusting the saturation area of the sigmoid function.

Table 4 is the specific comparison results from the "00" question for the three algorithms. From Table 4 we can see that as the complexity of the problem was increased, the training time also increased for three algorithms, but our proposed algorithm was much more effective on the complicated BSPQ problem for the requested error criterion.

Although the BSPQ problem is only a simple prediction task for the ENN, the other problems such as the detection of the wave amplitude can also be repeated with the proposed algorithgm to test the effectiveness. We feel there is a posibility for the proposed algorightmn to be applied as well as in the Jordan network or other partially modified recurrent neural networks since the structure of the ENN and the above mentioned networks are almost similar.

## CONCLUSION

In this study, we proposed a modified error function with two terms for ENN algorithm. This modified error function was used to harmonize the update of weights connected to the hidden layer and those connected to the output layer in order to avoid the localminima problem in the training learning process. Finally, the algorithm has been applied to the BSPQ problems including "11", "111" and "00" problems. Simulation results shows that the proposed algorithm is more effective at getting rid of local minima problem with less time and getting good prediction results than original ENN algorithm and improved ENN algorithm. Although the proposed algorithm is only been experimented with ENN, we somehow feel that there are posibility for it to be implemented in Jordan network or other partially modified recurrent neural networks due to the similarities of the structure. This area of research would be one of our foregoing efforts in expanding the effectiveness of local search algorithm in networks similar with ENN.

## REFERENCES

Adem Kalinli and Seref Sagiroglu, 2006. Eleman Network with Embedded Memory for System Identification. J. Inform. Sci. Eng., 22: 1555-1668.

Chagra, W., R.B. Abdennour and F. Bouani, 1998. A comparative study on the channel modeling using feedforward and recurrent neural network structures, Proc. IEEE. Int. Conf. Sys. Man and Cybernetics, 4: 3759-3763.

Conner, J.T., D. Martin and L.E. Atlas, 1994. Recurrent neural networks and robust time series prediction, IEEE. Trans. Neural Networks, 5: 240-253.

Cybenko, G., 1989. Approximation by superposition of a sigmoid function. Mathematics of Control, Signals and Sys., 2: 303-314.

Elman, J.L., 1990. Finding Structure in Time, Cognitive Sci., 14: 179-211.

GAO, X.Z., X.M. GAO and S.J. Ovaska, 1996. A modified Eleman neural network model with application to dynamical systems identification, Proc. IEEE. Int. Conf. Sys. Man and Cybernetics, 2: 1376-1381.

Hornik, K., M. Stinchcombe and H. White, 1989. Multilayer forward networks are nuniversal approximators, Neural Networks, pp: 359-366.

http://www.mathworks.com.

Huang. B.Q., T. Rashid and T. Kechadi, 2004. A New Modified Network based on the Elman Network, Proceeding of Artificial Intelligence and Applications.

Lawrence, S., C.L. Giles and S. Fong 2000. Natural language grammatical inference with recurrent neural network, IEEE. Trans. Knowledge Data Eng., 12: 126-140.

Pham, D.T. and X. Liu, 1993. Identification of linear and nonlinear dynamic systems using recurrent neural networks, Artific. Intelligence Eng., 8: 90-97.

Schreiber, C.S., H. Printz and J.D. Cohen, 1990. A network model of neuromodulatory effects: Gain, Signal-to-noise ratio and behavior, Science, 249: 892-895.

Shi, X.H., Y.C. Liang and X. Xu, 2003. An improved Elman model and recurrent back-propagation control neural network. J. Software, 14: 1110-1119.

Smith Andrew, 2004. Branch prediction with Neural Networks: Hidden layers and Recurrent Connections, Department of Computer Science University of California, San Diego La Jolla, CA 92307.

Stagge, P. and B. Sendhoff, 1999. Organisation of past States in Recurrent Neural Networks: Implicit Embedding, In: M. Mohammadian (Ed.), Computational Intelligence for Modelling, Control and Automation, IOS Press, Amsterda, pp: 21-27.