An Algorithm of Two-Phase Learning for Eleman Neural Network to Avoid the Local Minima Problem

***Zhiqiang Zhang*†, *Zheng Tang*†**

†*Faculty of Engineering, Toyama University, Toyama-shi, 930-8555 Japan*

Summary
Eleman Neural Network 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 process of the convergence, we propose an improved algorithm which includes two phases, a backpropagation phase and a gradient ascent phase. When network gets stuck in local minimum, the gradient ascent phase is performed in an attempt to fill up the local minimum valley by modifying parameter in a gradient ascent direction of the energy function. We apply this method to the Boolean Series Prediction Questions to demonstrate its validity. The simulation result shows that the proposed method can avoid the local minima problem and largely accelerate the speed of the convergence and get good results for the prediction tasks.

Key words:
Eleman Neural Network (ENN), Local Minima Problem, Gain parameter; Boolean Series Prediction Questions (BSPQ)

1. Introduction

Eleman Neural Network (ENN) is one type of the partial recurrent neural network which more includes Jordan neural networks [1-2]. ENN consists of two-layer back propagation networks with an additional feedback connection from the output of the hidden layer to its input. 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 [3-4]. 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 [1].

However, the ENN is a local recurrent network, so when learning a problem it needs more hidden neurons in its hidden layer and more training time than actually are required for a solution by other methods. Furthermore, the ENN 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 [5]. Therefore, having a fair number of neurons to begin with makes it more likely that the hidden neurons will start out dividing up the input space in useful ways. Since ENN uses back propagation (BP) to deal with the various signals, it has been approved that it suffers from a sub-optimal solution problem [6-10]. Therefore, the same problem would occur in the prediction task for the ENN.

The efficiency of the ENN is limited to low order system due to the insufficient memory capacity [11-12]. ENN had failed in identifying even second order linear systems as reported in the references [1, 6, 13-14], several approaches have been suggested in the literature to increase the performance of the ENN with simple modifications [15-19]. One of the modified methods is proposed by Pham and Liu on the idea which adds a self-connection weight (fixed between 0.0 and 1.0 before the training process) for the context layer. Furthermore, two methods about output-input feedback Eleman network (a context is added between output layer and input layer) and output-hidden feedback Eleman network (a context is added between output layer and hidden layer) are the classical improvement of the ENN [20]. The two methods have largely enhanced the memory capacity of the network and got broadly application by adding other contexts unit. The suggested modifications on the ENN in the literature mostly have been able to improve certain kinds of problems, but it is not clear yet which network architecture is best suited to dynamic system identification or prediction [21]. And at the same time, these methods usually change or add some other elements or connections in the network and enhance the complexity of the computation. However, these improved modifications attempt to add other feedback connections factors to the model that will increase the capacity of the memory in order to overcome the tendency to sink into local minima. The random perturbations of the search direction and various kinds of stochastic adjustment to the current set of weights are largely ineffective at enabling network to escape from local minima and make the network fail to converge to a global minimum within a reasonable number of iterations [22-23]. So the local minimum problem still is

Manuscript received April 5, 2007
Manuscript revised April 25, 2007

a serious problem and usually occurs in various applications.

In this paper, we propose a novel approach to supervised learning for multilayer artificial ENN. The learning model has two phases－a Backpropagation phase, and a gradient ascent phase. The Backpropagation phase performs steepest descent on a surface in weight space whose height at any point in weight space is equal to an error measure, and finds a set of weights minimizing the error measure. When the Backpropagation gets stuck in local minima, the gradient ascent phase attempts to fill up the valley by modifying gain parameters in a gradient ascent direction of the error measure. Thus, the two phases are repeated until the network gets out of local minima. The learning model has been applied into the Boolean Series Prediction Questions (BSPQ) problems including "11"," 111" and "00" problems. The proposed algorithm is shown to be capable of escaping from local minima and get better simulation results than the original ENN and improved ENN. Since a three-layered network is capable of forming arbitrarily close approximation to any continuous nonlinear mapping [24-25], we use three layers for all training network.

2. Structure of ENN

Fig. 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 stored as "extra inputs" (called context unit) to be used when the next time the network is operated, and the recurrent units of the connect layer is same as the units form hidden layer. 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 the 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 trainable between hidden units and context units as others weights [1]. 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 proposed by Pham and Liu [6]. When weights (a) are 0, the network is the original ENN.

Fig. 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. One approach used in the machine learning is to operate the process by time shown as Fig. 3.

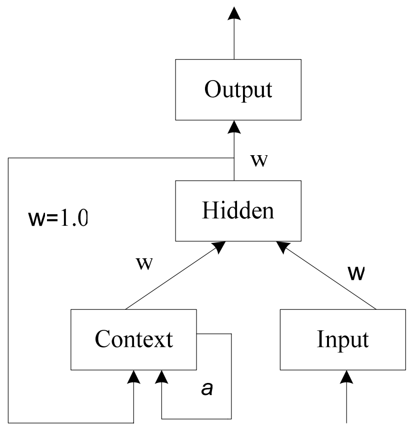

Fig. 1. The Structure of the ENN

Fig. 3 shows that a long feed forward network where by back propagation is able to calculate the derivatives of the error (at each output unit) by unrolling the network to the beginning. At the next time step t+1 input is represented , here the context units contain values which are exactly the hidden unit values at time t (and the time t-1 , t-2 …), thus, these context units provide the network with memory [28]. Therefore, the ENN network is converted into a dynamical network that is efficient use of temporal information in the input sequence, both for classification [29-30] as well as for prediction [31-32].

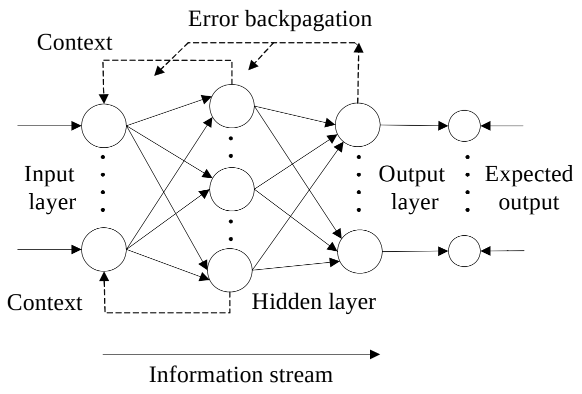

Fig. 2. Internal Process Analysis of ENN

Based on Fig. 2 and Fig.3 it is obvious that it is identical to the feed-forward network except for the hidden units taking inputs consisting of weighted sums of the previous values of the hidden units. At a word, ENN is not merely a function of its inputs but also computes some value of its past values, able to recall it when it is required and then it is then used again. However, these representations of temporal context need not be literal and they represent a memory which is highly task and stimulus-dependent [1]. So the algorithm of the network is almost same as the back-propagation algorithm except the context layers which bring the memory to the network.

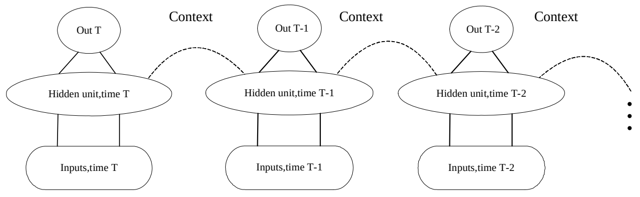

Fig. 3. Unroll the ENN Through Time

Fig. 4 is a flowchart of the proposed learning algorithm. In the flowchart, phase I is the backpropagation phase in weight space, and phase II is the gradient ascent phase in gain parameter space.

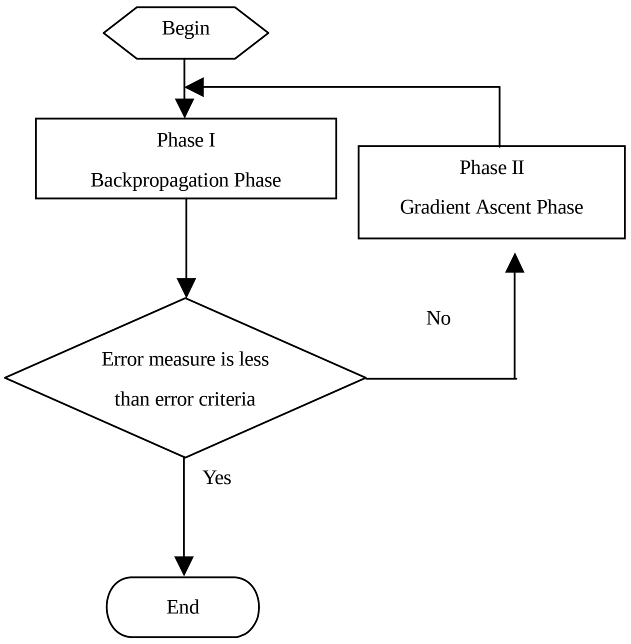

Fig.4 Flowchart of proposed algorithm.

3. Two−Phase Learning Algorithm

3.1 Backpropagation Phase for ENN

Backpropagation ENN learning algorithm and its variations are widely used as methods for the syntactic or semantic categories in natural language data. The Backpropagation algorithm for ENN tries to find a set of weights and thresholds that minimizes an overall error measure E

$$E = \sum_{t=0}^{T} E_p \tag{1}$$

where p indexes over all the patterns for the training set in the time interval [0, T]. In our paper, the time element is updated by the next input of the pattern training set. So we can get the following equation as following.

$$E = \sum_{t=0}^{T} E_p = \sum_{p=1}^{P} E_p \tag{2}$$

E_p is defined by

$$E_p = \frac{1}{2}\sum_j (t_{pj}^k - o_{pj}^k)^2 \tag{3}$$

where t_{pj}^k is target value (desired output) of the *j*-th component of the output for pattern *p* and o_{pj}^k 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 over the output units. To get the correct generalization of the delta rule, the BP algorithm for ENN set at

$$\Delta_p w_{ji} = -\eta_A \frac{\partial E_p}{\partial w_{ji}} \tag{4}$$

Where w_{ji} is the weight connected between unit *i* and unit *j*. The input of each unit in a layer (except input layer) is given by

$$net_{pj}^k = \theta_j + \sum_i w_{ji} o_{pi} + \sum_i r_{ji} u_{pi}^k \tag{5}$$

where net_{pj}^k is the net input to unit *j* in a layer produced by the presentation of pattern *p* at the time *k*, θ_j is a threshold of the unit *j* and o_{pi} is the output value of unit *i* for pattern p. let r_{ji} represent the forward weights from the context layer to the hidden layer, u_{pi}^k is the recurrent value from the hidden layer to the context layer, but only when net_{pj}^k is the net input for the neuron unit in the hidden layer, u_{pi}^k will be valid, otherwise, it will be 0. The number of the neuron unit in the hidden layer and context layer is same for the ENN.
The output of unit *j* for pattern *p* is specified by

$$o_{pj}^k = f_j'(net_{pj}^k) \tag{6}$$

where *f(x)* is a semilinear activation function which is differentiable and nondecreasing. According to the definition of the ENN, the output value of the hidden layer at time *k-1* is the input value for the context layer at time *k*, so

$$o_{pi}^k = o_{pj}^{k-1} \tag{7}$$

The rule for changing weight (threshold) typically used in Backpropagation is given by

$$\Delta_p w_{ji} = -\eta \frac{\partial E_p}{\partial w_{ji}} = -\eta \frac{\partial E_p}{\partial net_{pj}^k} \frac{\partial net_{pj}^k}{\partial w_{ji}} = \eta \delta_{pj}^k o_{pi}^k \tag{8}$$

where $\Delta_p w_{ji}$ is the change to be made to w_{ji} following presentation of pattern *p* and η is called the learning rate. δ_{pj}^k is defined as $-\partial E_p / \partial net_{pj}^k$ at time *k*. When unit *j* is an output unit, then δ_{pj}^k in the above equation is given by

$$\delta^k_{pj} = (t^k_{pj} - o^k_{pj}) f'_j(net^k_{pj}) \tag{9}$$

And when unit *j* is a hidden unit, then δ^k_{pj} is given by

$$\delta^k_{pj} = f'_j(net^k_{pj}) \sum_m o^k_{pm} w_{mj} \tag{10}$$

where $f'_j(net^k_{pj})$ is the derivative of the activation function of unit *j*, w_{mj} is the weights between output layer and hidden layer.

But for the weight r_{ji} between context layer and hidden layer, the adjustment of the weight is given by

$$\Delta_p r_{ji} = -\eta \frac{\partial E_p}{\partial r_{ji}} = -\eta \frac{\partial E_p}{\partial o^k_{pj}} \frac{\partial o^k_{pj}}{\partial r_{ji}} \tag{11}$$

where o^k_{pj} is the output value of the hidden layer for the unit j, and we can get the following equations, respectively.

$$\frac{\partial E_p}{\partial o^k_{pj}} = -\eta \frac{\partial E_p}{\partial net^k_{pm}} \frac{\partial net^k_{pm}}{\partial o_{pj}} = -\sum_m o^k_{pm} w_{mj} \tag{12}$$

where net^k_{pm} is the input value of the out layer for the unit *m*, w_{mj} is the weight between hidden layer and output layer for the unit m.

$$\frac{\partial o^k_{pj}}{\partial r_{ji}} = \frac{\partial f'(net^k_{pj})}{\partial r_{ji}} = f'(\theta_j + \sum_i w_{ji} o_{pi} + \sum_i r_{ji} u^k_{pi}) = u^k_{pi} f'(net^k_{pj}) o^{k-1}_{pj} \tag{13}$$

where *j* is the input value of the hidden layer for the unit *j*. So, we can get the final computation equation as following

$$\Delta_p r_{ji} = -\eta \frac{\partial E_p}{\partial r_{ji}} = -\eta \sum_m \delta^k_{pm} w_{mj} f'(net^k_{pj}) o^{k-1}_{pj} \tag{14}$$

Where w_{mj} are the weights between the output and the hidden layer for the unit *j*. Thus, the Backpropagation rule (referred to as the generalized delta rule) causes each iteration to modify weights in such way as to approximate the steepest descent. But, if the Backpropagation descends into a minimum that achieves insufficient accuracy in the functional approximation, it fails to learn.

To help the network escape from the local minima, we add a gradient ascent phase in the gain space. The details are described in the next section. First, we must consider when a gradient ascent phase should be started. in the backpropagation phase, the absolute value of the change of the error measure is accumulated for every 50 corrections. Here, one weight correction corresponded to backpropagation modification to weights for all patterns. If the accumulation of change of E for the 50 weight corrections was less than a very small pre-selected constant, in our paper, it is 0.001, and the current error measure E was larger than the error criteria (E=0.1 or E=0.01, where E was the sum of squares error function for the full training set.), the Backpropagation learning (Phase I) is considered to be trapped in a local minimum. Then the backpropagation phase stops and goes to the gradient ascent phase.

3.2Gradient Ascend Phase for ENN

It is known to all that the Backpropagation in weight space for ENN may lead to a convergence to either a local minimum or a global minimum. However, there is not an effective way for the network to reach to global minimum from a local minimum. We propose a gradient ascent learning method that attempts to fill up a local minimum valley by increasing the error measure of a local minimum in the best path. Usually, the activation function of unit *j*, f_j is given by a sigmoid function $f_j(x)$ with the "gain" parameter g_j.

$$f(x) = \frac{1}{1 + e^{-g_j x}} \tag{15}$$

The error measure is also a function of the gain parameters of neurons. Therefore we can increase intentionally the error measure by modifying the gain parameters in a gradient ascent direction of the error measure in gain parameter space and drive the Backpropagation out of the local minimum. We consider the gain parameters as variables that can be modified during the phase II as weights and thresholds in phase I. Here suppose that a vector g corresponds to the gain parameters of neurons. Since for the gain vector g, the modification requires the change of the gain to be in the positive gradient direction, we take:

$$\Delta_g = \varepsilon \nabla E(\vec{g}^{\,w}) \tag{16}$$

where ε is a positive learning rate constant and ∇E is the gradient of the error measure E with respect to the gain parameter *g*. If the constant ε is small enough, the change of the gain parameter *g* results in an increase of the error measure (Phase II). The derivative of the error measure with respect to the gain parameter g_j of neuron *j* can be easily calculated using the chain rule as following (here we do not consider the time gene):

$$\Delta_p g_j = \varepsilon \frac{\partial E_p}{\partial g_j} \tag{17}$$

And now let us define $net'_j = g_j \times net_j$, then

$$\Delta_p g_j = \varepsilon \frac{\partial E_p}{\partial net'_j} \frac{\partial net'_j}{\partial g_j} = \varepsilon \frac{\partial E_p}{\partial net'_j} net_j \tag{18}$$

because

$$\frac{\partial E_p}{\partial net_j} = -\delta_{pj} = \frac{\partial E_p}{\partial net'_j} g_j \tag{19}$$

$$\frac{\partial E_p}{\partial net'_j} = -\frac{\delta_{pj}}{g_j} \tag{20}$$

Finally

$$\Delta_p g_j = \varepsilon \frac{\partial E_p}{\partial g_j} = -\varepsilon \delta_{pj} \frac{net_j}{g_j} \tag{21}$$

Here, δ_{pj} can be computed as the same as conventional Backpropagation, for any output unit *j*,

$$\delta_{pj} = (t_{pj} - o_{pj}) f_j'(net_{pj}) \tag{22}$$

and for the unit that is not an output unit,

$$\delta_{pj} = f_j'(net_{pj}) \sum_m o_{pm} w_{mj} \tag{23}$$

We take $f_j(x) = (1)/(1 + e^{-g_j x})$ as activation function, in this case:

$$f_j'(net_{pj}) = g_j o_{pj}(1 - o_{pj}) \tag{24}$$

Hence for an output layer unit *j*

$$\delta_{pj} = g_j (t_{pj} - o_{pj}) o_{pj} (1 - o_{pj}) \tag{25}$$

and for a hidden unit j

$$\delta_{pj} = g_j o_{pj}(1 - o_{pj}) \sum_m \delta_{pm} w_{mj} \tag{26}$$

where w_{mj} represents the weights between hidden layer and output layer

We can calculate the $\Delta_p g_j$ through the above δ rules alike as the gradient descend phase.

In Phase II, the absolute value of the change of the error measure E is also accumulated for every gain correction. If the accumulated value of the error measure E is larger than a preselected small constant, for example, 0.02 in our simulation, go to phase II.

In order to explain simply how the proposed algorithm helps a network escape from a local minimum, we use a two-dimensional graph of an error measure of a neural network with a local minimum and a global minimum, as shown in Fig.5 (a). The horizontal axis represents the error produced by the network corresponding to some state in weight space on the vertical axis (to facilitate understanding, it is expressed in one dimension). The initial network defines a point (e.g., point A) on the slope of a specific "valley", the bottom of this valley (point B) is found by error minimization of the Backpropagation in the weight space (Fig.5 (b)). Then the error measure is evaluated. If the error measure is less than an error criterion, then stop. If connection weights cannot be corrected even when the network has larger error values than the error criteria, go to phase II. Phase II tends to increase the error measure in the gradient ascent direction of the gain space.

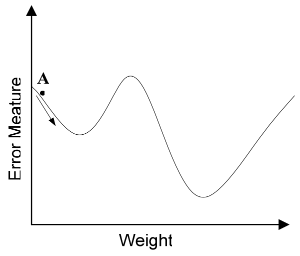

Fig .5(a)

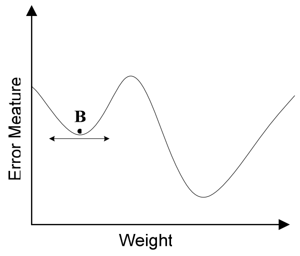

Fig .5(b)

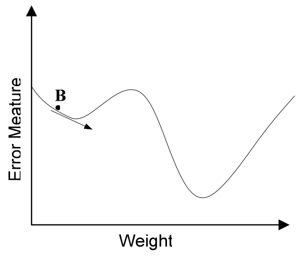

Fig .5(c)

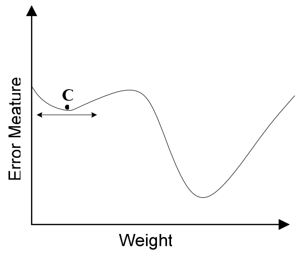

Fig .5(d)

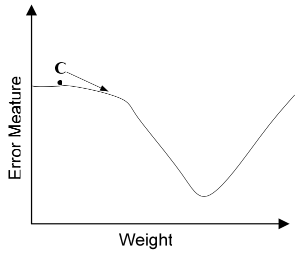

Fig .5(e)

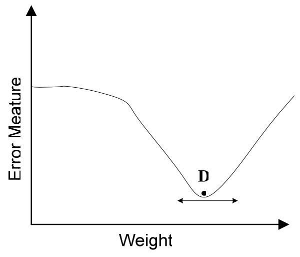

Fig .5(f)

Fig.5 The relationship between error measure and the weight space

After the gradient ascent phase (Phase II), the error measure at point B is raised, and then point B becomes a point at the slope of the "valley" in weight landscape again (Fig.5 (c)). Then Backpropagation phase seeks a new minimum (Point C) in weight landscape (Fig.5 (d)). Thus, the repeats of Backpropagation phase in weight space and the gradient ascent phase in gain space may result in a movement out of a local minimum as shown in Fig.5 (e)(f). In this way, phase I and II were repeated until the error measure was less than the error criteria, which is assumed to be a successful run, oppositely, when the iterative steps reach an upper limit of 30000, it was assumed to be an unsuccessful run.

4. Simulation

In order to test the effectiveness of the proposed method, we compared its performance with those of the original ENN algorithm (proposed by Eleman [1]) and improved ENN algorithm (proposed by Pham and Liu [6]) on a series of BSPQ problems including "11", "111" and

"00" problems. To compare them, each algorithm was used to train a network on a given problem with identical starting weight values and momentum values. In our proposed method, when the network was trapped into the local minimum, go to Phase II and the gain parameters of all neuron in hidden layer were modified to help the network escape from the local minimum, which is found to be a more effective scheme during the prediction tasks.

Three aspects of training algorithm performance－"success rate" ,"iterative" and "training time" were assessed for each algorithm. Every method will run 100 trials. And the iterative steps were two parts which were the backpropagation phase and the gradient ascent phase for the proposed ENN algorithm. All data of simulations were performed out on an PC (Pentium4 2.8GHz, 1G). We used the modified back propagation algorithm with momentum 0.9. And the learning rate η =1.0 for the backpropagation and ε =1.0 for the gradient ascent were selected. The weights and thresholds were initialized randomly from 0.0 to 1.0 and the gain parameters of all neurons were set to 1.0 initially. In this condition, the requested error criteria was very high, if the network reach to this error criteria point (E=0.1 or E=0.01), all patterns in the training set could get a tolerance of 0.05 for each target element. And we used the well trained network to do the final prediction about sequence P_1 to test the prediction capacity of it. For all the trials, 150 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.

4. 1. "11" questions

Boolean Series Prediction Questions is one of the problems about time sequence prediction. First let us see the definition of the BSPQ [10]. Now suppose that we want to train a network with an input P and targets T as defined below.

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 of the Context layer of the network. In this paper we define the prediction set P_1 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

Firstly, we deal with the "11" question and analyze the effect of the memory from the context layer for the network.

Table 1 compared the simulation results of the three algorithm, we can see that our proposed method could almost 100% succeeded to get the convergent criterion. Of course, the original ENN was able to predict the requested input test, but the training success rate was slow. Further, it succeeded only 75% 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 gene (a=0.5). Although improved ENN could accelerate the convergent of the learning process (iterative was less than original ENN), it could not essentially avoid the local minima problems, the success rate was only 68%, when error was set 0.01.

Fig.6 shows the learning characteristics for three methods with the same initialization weight for the network (1-5-1), when E was set to 0.1. From the simulation results we can see that the proposed ENN algorithm only needed about 170 iterations to be successful, but the original ENN and the improved ENN algorithm have not got the goal value and got into local minima point A and B, respectively. The improved ENN have only accelerated the process of the learning, but it could not escape the local minima problem. But our proposed ENN algorithm could get the convergent point by avoiding the local minima. From the "proposed ENN" line we can see that at time k_1, after the Backpropagation learning about 127 iterative steps, the error measure E decreased from 3.12 to 2.22 and got stuck in a local mimimum. Then the gradient ascent learning process (Phase II) was performed, and error is increased from 2.22 to 2.24 (k_2). Then the Backpropagation learning settled to a new local minimum (k_3). Backpropagation learning escaped from the local minimum and converged to the global minimum (k_4).

For the prediction set P_1, we can get its corresponding expected results 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. 7 is the simulation prediction result of P_1 for the "11" question with our proposed ENN algorithm. In the Fig. 7, the two lines represented the T_1 line and the prediction results line respectively. From the Fig. 7 we can see that the tolerance of every pattern was less than required standard 0.05. So the network has enough ability to do the prediction of the given task as desired.

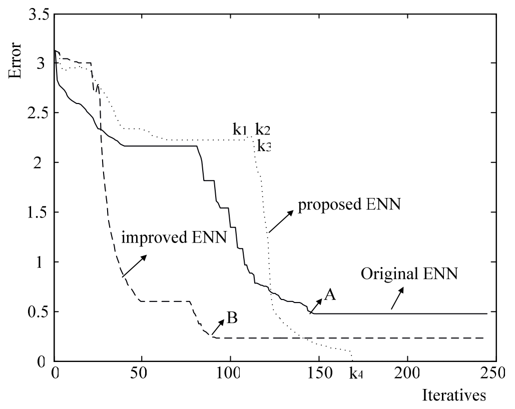

Fig.6. Training error curve of the three ENN algorithm

For the same problem, as we gradually increased the quantity of the neuron of the hidden layer, the original ENN and improved ENN were able to get to the convergence point with the finite iterative steps, however, our proposed ENN could 100% get to the error criterion by avoiding the local minima. The results are shown in Table 2.

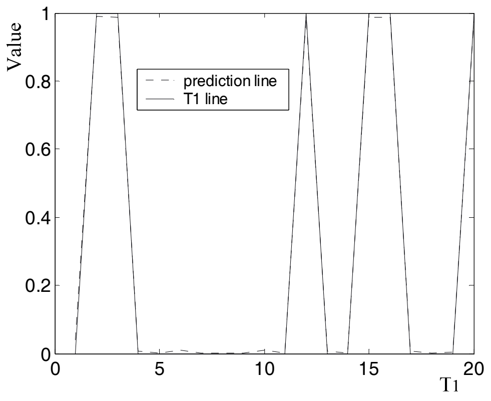

Fig.7. Expected Output T1 and prediction result with Proposed ENN

In order to better testify the effectiveness of the proposed algorithm, we continued to increase the quantity of neuron unit in the hidden layer.

From the Table 3, we can see, the proposed method could get almost 100% success rate than other two algorithms through avoiding the local minima problems, although the iterative steps was bigger than the improved ENN algorithm sometimes. This is because the network's iterative steps were increased because of the gradient ascent phase II.

4. 2. "111" and "00" questions

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

Table 4 is the specific comparison results from the "111" question for the three algorithms. From Table 4 we can see that, for the network (1-7-1), the success rate of the proposed ENN was lower (only 90% when the error criterion was set to 0.01) 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 100% succeed than original ENN and improved ENN algorithms although the iterative step was bigger than two methods. The proposed ENN has successfully escaped from the local minima problems through gradient ascent phase.

Table 5 is the specific comparison results from the "00" question for the three algorithms. From Table 5 we can see that, as the complexity of the problem was increased, the training time and iterative steps were also increased for three algorithms, but our proposed algorithm was much more effective on the complicated BSPQ problem to avoid the local minima problem.

Although the BSPQ problem is only a simple prediction task for the ENN, the other problems are also compatible with our proposed algorithm, such as the detection of the wave amplitude [10]. And our algorithm is also appropriate to the Jordan network or other partially modified recurrent neural networks, such as the network proposed by Shi [20].

5. Conclusion

In this paper, we proposed a supervised learning method for Eleman Neural Network to avoid the local minima problem. The approach was shown to be of higher convergence to global minimum than the original ENN learning algorithm and than the improved ENN algorithm in network training. The algorithm has been applied to the BSPQ problems including "11", "111"and "00" problems. Through the analysis of the result from various Boolean Series Predict problems we can see that the proposed algorithm is effective at avoiding the local minima problems and getting high success rate. Although we only applied the algorithm into the basic ENN, we make this improvement as a benchmark and take effort to research application of the algorithm in the recurrent neural network like Jordan network and fully recurrent network.

Acknowledgment

I would like to thank Professor Tang Zheng, my supervisor, for his many suggestions and constant support in the whole course of this research.

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

Methods(1-5-1 network)	Success rate(100 trials)		Iterative（average steps）		Average CPU time (second)	
	E=0.1	E=0.01	E=0.1	E=0.01	E=0.1	E=0.01
Original ENN	75%	66%	245	320	0.15	0.20
improved ENN	87%	65%	207	299	0.13	0.18
propose ENN	100%	100%	320	500	0.21	0.32

Table 2. Experiment results for the "11" question with 7 neurons in the hidden layer

Methods(1-7-1 network)	Success rate(100 trials)		Iterative（average）		Average CPU time (second)	
	E=0.1	E=0.01	E=0.1	E=0.01	E=0.1	E=0.01
Original ENN	89%	78%	375	568	0.22	0.31
Improved ENN	95%	90%	305	510	0.19	0.26
Proposed ENN	100%	100%	427	680	0.35	0.61

Table 3. Experiment results for the "11" question with 10 neurons in the hidden layer

Methods(1-10-1 network)	Success rate(100 trials)		Iterative（average）		Average CPU time (second)	
	E=0.1	E=0.01	E=0.1	E=0.01	E=0.1	E=0.01
Original ENN	91%	89%	455	789	0.25	0.40
Improved ENN	95%	91%	395	589	0.20	0.37
Proposed ENN	100%	98%	578	835	0.36	0.43

Table 4. Experiment results for the "111" question with different neuron units in the hidden layer

structure of the network	Items / Methods	Success rate(100 trials)		Iterative（average）		Average CPU time (second)	
		E=0.1	E=0.01	E=0.1	E=0.01	E=0.1	E=0.01
1-7-1 network	Original ENN	56%	43%	844	1009	0.52	0.70
	Improved ENN	75%	60%	560	783	0.40	0.50
	Proposed ENN	95%	90%	1009	1550	0.55	0.71
1-10-1 network	Original ENN	78%	74%	689	955	0.61	0.68
	Improved ENN	85%	79%	508	611	0.39	0.46
	Proposed ENN	100%	99%	1143	1690	0.62	0.88
1-12-1 network	Original ENN	85%	75%	811	1027	0.51	0.71
	Improved ENN	90%	81%	665	749	0.47	0.49
	Proposed ENN	100%	98%	1240	1432	0.66	0.73

Table 5. Experiment results for the "00" question with different neuron units in the hidden layer

structure of the network	Items / Methods	Success rate(100 trials)		Iterative（average）		Average CPU time (second)	
		E=0.1	E=0.01	E=0.1	E=0.01	E=0.1	E=0.01
1-5-1 network	Original ENN	78%	71%	300	455	0.21	0.35
	Improved ENN	90%	84%	148	250	0.09	0.19
	Proposed ENN	100%	97%	398	661	0.22	0.37
1-7-1 network	Original ENN	85%	81%	458	960	0.28	0.50
	Improved ENN	95%	90%	356	601	0.22	0.39
	Proposed ENN	99%	99%	600	1254	0.40	0.75
1-10-1 network	Original ENN	95%	89%	628	1095	0.43	0.6
	Improved ENN	98%	92%	588	901	0.37	0.51
	Proposed ENN	100%	100%	870	1138	0.69	0.85

References

[1] Jeffrey L.Eleman, "Finding Structure in Time", Cognitive Science, 14,179-211,1990.

[2] M. I. Jordan, "Attractor dynamics and parallelism in a connectionsist sequential machine" in Proceedings of the 8th Conference on Cognitive Science, pp.531-546,1986.

[3] C.W.Omlin, C.L. Giles, Extraction of rules from dicrete-time recurrent neural networks, Neural Networks 9(1),41-52,1996.

[4] P.Stagge, B. Sendhoff, Organisation of past states in recurrent neural networks: implicit embedding, in: M. Mohammadian (Ed.), Computational Intelligence for Modelling, Control ¥& Automation, IOS Press, Amsterda.pp.21-27,1999.

[5] G.Cybenko, "approximation by superposition of a sigmoid function", Mathematics of Control, Signals, and Systems, 2,303-314,1989.

[6] D.T.Pham and X. Liu, "Identification of linear and nonlinear dynamic systems using recurrent neural networks", Artificial Intelligence in Engineering, Vol.8,pp.90-97,1993.

[7] Andrew SMITH," Branch prediction with Neural Networks: Hidden layersand Recurrent Connections" , Department of Computer Science University of California, San Diego La Jolla, CA 92307, 2004.

[8] G.Cybenko, "approximation by superposition of a sigmoid function", Mathematics of Control, Signals, and Systems, 2,303-314, 1989.

[9] Multilayer Network Learning Algorithm Based on Pattern Search Method, IEICE Transaction on Fundamentals of Electronics, Communications and Computer Science, E86-A, 1869-1875, 2003.

[10] http://www.mathworks.com

[11] C.L. Giles, C.B. Miller, D, Chen, G.Z. Sun, H.H. Chen, Y.C. Lee, Extracting and learning an unknown grammar with recurrent neural networks, in: J.E. Moody, S.J. Hanson. R.P. Lippmann (Eds.), Advances in Neural Information Processing Systems 4, Morgan Kaufmann Publishers, San Mateo, CA, pp.317-324, 1992.

[12] R.I.Watrous, G. M. Kuhn, Induction fo finite-state automata using second-order recurrent neural networks, in: J. E. Moody, S. J. Hanson, R. P. Lippmann (Eds.), Advances in Neural Information Processing System 4, Morgan Kaufmann Publishers, San Mateo, CA, pp.317-324, 1992.

[13]A.Kalinli and D.Karaboga,"Training recurrent neural networks by using parallel tabu search algorithm based on crossover operation", Engineering Applications of Artificial Intelligence, Vol. 17, pp.529-542,2004.

[14] D.T.Pham and X.Liu, "Training of Eleman networks and dynamic system modeling", International Journal of Systems Science, Vol.27,pp.221-226,1996.

[15]J. W. Dong, J. X. Qian and Y.X.Sun, "Recurrent neural networks for recursive identification of nonlinear dynamic process", in Proceedings of the IEEE International Conference on IndustrialTechnology,pp.794-798,1994.

[16] D.P.Kwok, P. Wang, and K.Zhou,"Process identification using a modified Eleman neural network", International Symposium on Speech, Image Processing and Neural Networks,pp,499-502,1994,

[17] X.Z.Gao,X.M.Gao, and S.J.Ovaska, "A modified Eleman neural network model with application to dynamical systems identification", in Proceedings of the IEEE International Conference on System , Man and Cybernetics, Vol.2, pp.1376-1381,1996.

[18] W.Chagra, R. B. Abdennour, F.Bouani, M.Ksouri, and G.Favier, "A comparative study on the channel modeling using feedforward and recurrent neural network structures," in Proceedings of the IEEE International Conference on System , Man and Cybernetics, Vol.4, pp.3759-3763,1998.

[19] J.E.W. Holm and N.J.H.Kotze, "training recurrent neural networks with leapfrog," in Proceeding of the International Conference on Industrial Electronics, pp.99-104,1998.

[20] Shi XH, Liang YC, Xu x. An improved Eleman model and recurrent back-propagation control neural network. Journal of software , 14(6):1110－1119,2003.

[21] Adem Kalinli and Seref Sagiroglu, "Eleman Network with Embedded Memory for System Identification" in Journal of Informaiton Science and Engineering 22, 1555-1668,2006.

[22] Ingman and Y. Merlis, " Local minimization escape using thermodynamic properties of Neural Networks", Neural Networks, vol. 4,no.3, pp.395-404,1991.

[23] Wang Chuan, Principe Jose.C, "Training neural networks with additive noise in the desired signal", IEEE Transactions on Neural Networks, vol. 10, no. 6, pp. 1511-1517, 1999.

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

[25]G.Cybenko, "Approximation by superposition of a sigmoid function", Mathematics of Control, Signals, and System, vol.2, 303-314, 1989.

[26]K.Funahashi, "On the approximate realization of continuous mapping by neural networks", Neural Networks, vol.2, 183-192, 1989.

[27]K. Hornik, M.Stinchcombe and H.White, "Multilayer forward networks are nuniversal approximators", Neural Networks, vol.I, 359-366, 1989.

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

[29] C.L.Giles, C.B.Miller, D,Chen, G.Z.Sun, H.H. Chen, Y.C. Lee, Extracting and learning an unknown grammar with recurrent neural networks, in : J. E. Moody, S.J. Hanson. R.P. Lippmann (Eds.), Advances in Neural Information Processing Systems 4, Morgan Kaufmann Publishers, San Mateo, CA, pp.317-324, 1992.

[30] S. Lawrence, C.L.Giles, S.Fong, Natural language grammatical inference with recurrent neural network, IEEE Trans. Knowledge Data Eng. 12(1), 126-140, 2000.

[31]. J.T.Conner, D. Martin, L.E. Atlas, recurrent neural networks and robust time series prediction, IEEE Trans. Neural Networks 5(2) 240-253, 1994.

[32].E.W.Saad, D.V.Prokhorov, D.C.Wunsch, Comparative study of stock trend prediction using time delay recurrent and probabilistic neural networks, IEEE Trans, Neural Networks 9(6),1456-1470, 1998.

[33]C.Servan-Schreiber, H.Printz and J.D.Cohen, "A network model of neuromodulatory effects: Gain, signal-to-noise ratio, and behavior", Science, vol.249,pp.206-208,1976.

[34] T.L.Burrow, and M.Niranjan,"The use of feed forward and recurrent neural networks for system identification",

Technical report, Cambridge University Engineering Department, 1993.

ZhiQiang Zhang Received his B.S degree and M.S. degree from Shandong University, Jinan, Shandong, China, in 2002, 2005 and in computer science and management science, respectively. He is currently working for his Ph.D. degree at Toyama University, Japan. His main research interests are neural networks and optimizations.

Zheng Tang Received the B.S. degree from Zhejiang University, Zhejiang, China in 1982 and an M.S. degree and a D.E. degree from Tshinghua University, Beijing, China in 1984 and 1988, respectively. From 1988 to 1989, he was an Instructor in the Institute of Microelectronics at Tsinghua University. From 1990 to 1999, he was an associate professor in the Department of Electrical and Electronic Engineering, Miyazaki University, Miyazaki, Japan. In 2000, he joined Toyama University, Toyama, Japan, where he is currently a professor in the Department of Intellectual Information Systems. His current research interests include intellectual information technology, neural networks, and optimizations.