Computer Society of Iran
Second Annual Computer Conference
(CSICC '96)

Amirkabir University of Technology
Tehran, I. R. Iran
December 24 -26, 1996

APPLICATION OF LEARNING AUTOMATA TO NEURAL NETWORKS

Mohammad B. Menhaj — Electrical Engineering department
Mohammad R. Meybodi — Computer Engineering department
Amirkabir University of Technology
Tehran-Iran

KEYWORDS:
NEURAL NETWORKS, SUPERVISED LEARNING, AUTOMATA

ABSTRACT

This paper presents another application of Learning Automata (LA) to the training of multi-layer perceptrons. Here, in compared to the first application[1], several learning automata have been assigned to the neural net in order to remedy the local minima problem associated with the neural network learning scheme. The simulation results show that this scheme allows those neurons that may cause the local trapping to choose their own learning rates to bypass the local minima.

1. INTRODUCTION

As a novel idea, Neural Network (NN) did not suddenly arrive on the scientific scene. There has currently been a great deal of efforts by scientists (i.e., neurobiologists, psychologists, physicists and system theorists) researching neural network topologies actively over the last three decades. They continued to explore further aspects of neuron-like inspired models such as perceptron (Rosenblatt), ADALINE (Widrow), Backpropagation (Werbos, Rumelhart), self-organization (Kohonen), hopfield (Hopfield), ART (Grossberg),.....[2-10]. For a good survey of the historical events that leads to the development of the neural networks one may refer to [11].

As an assembly of several highly parallel interconnected processing elements, neural networks present a radically different approach to the computation. In neurobiologist's point of view neural networks exhibit characteristics analogous to human brain, but as a system theorist they act as highly nonlinear adaptive model-free function approximators. They are able to learn from their experience. This marvelous property made them very popular.

At the same time the theory of Learning Automaton (LA) has made significant progress in the last twenty years, but the application of the automata to other fields or to real-world problems has not grown in a similar manner [12-13]. The main idea behind the learning automaton approach is to provide a general framework for the design of large stochastic distributed systems. In this paper we present an application of the learning automaton approach to the artificial neural network.

By interconnection of automata to the feedforward neural network, we apply the learning automaton scheme to such networks for adjusting the synaptic weights. The parameter adaptation is accomplished based on the observation of random response of the neural network. Here the automaton, as a part of an augmented system, acts individually. It uses very little prior information.

This paper presents the application of learning automata to the training of multi-layer perceptrons. As a first phase of study, reported in [1], the automata approach incorporated into the backpropagation algorithm for training feedforward neural networks. A single learning automata was associated to the whole network for determining the best learning step size. The NN-learning algorithm consists of presenting at each iteration one input of the training set, computing the network's response and correcting the weights. The amount of the correction is proportional to the learning size. The single learning automata associated with the whole network gives the best step length used by the NN-Learning algorithm at each iteration.
In this paper the following improvements have been made to (LA+NN)-scheme.

1- In order to remedy the local minima problem associated with the NN-learning scheme, a separate LA will be assigned to each neuron of the network. This scheme allows those neurons which may cause the local trapping to choose their own learning rates in order to bypass the local minima. The local minima problem has been also solved by assigning a separate LA to each layer of the neural network. This approach also makes the incorrect saturation problem be avoided.

2- The single automata approach has been used but this time the interval from which the center of learning rates chosen will be changed as the automata operates. This makes the (LA+NN)-scheme to find a more accurate learning rate which leads to an increase of the speed of convergence of the overall net.

The paper is organized as follows. Section 2 briefly presents the standard backpropagation algorithm along with its notation that will be used in this paper. The learning automaton is then introduced in section 3. Section 4 describes the interconnection of the LAs into the neural networks. Section 5 presents simulation results and section 6 contains conclusions.

2. STANDARD BACKPROPAGATION

Given a set of training input/output data, $\{(\underline{p}^q, \underline{t}^q),\ q = 1, \ldots, Q\}$, the standard learning rule for multilayer feedforward neural net is the BackPropagation(BP) algorithm which can be summarized as follows. Consider a feedforward neural network with M layers (i.e. the M-th layer represents the output layer and there are M-1 hidden layers), depicted in figure 1, whose behavior is described by the following equations

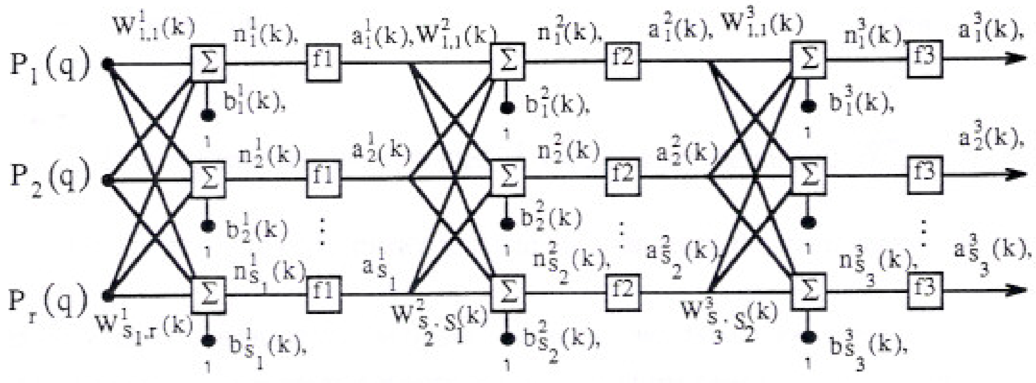

Figure 1. Three-Layer Feedforward Network

$$
\begin{aligned}
a^0_{i,q} &= p^q_i, \\
n^m_{i,q} &= \sum_j w^m_{i,j} \times a^{m-1}_{j,q} + b^m_i, \qquad m = 1, \ldots, M \qquad (1) \\
a^m_{i,q} &= f^m\left(n^m_{i,q}\right),
\end{aligned}
$$

where m is the layer index, q is the pattern number and $f^m(.)$ is the activation function in layer m.

Classically the objective to be minimized is the performance criterion

$$\underset{w^m_{i,j}, b^m_i}{\text{minimize}}\ \hat{F} = \frac{1}{Q}\sum_{q=1}^{Q} \hat{F}_q,$$

with, $$\hat{F}_q = 0.5\sum_{i=1}^{S_M}\left(t^q_i - a^m_{i,q}\right)^2, \qquad (2)$$

where $\underline{t}^q$ is the desired output vector corresponding to the q-th input pattern, $\hat{F}_q$ is the contribution of the q-th pattern to the objective function $\hat{F}$, and S_m denotes the number of neurons at layer m. The standard BP algorithm which is actually an approximate steepest descent rule is

$$\Delta w_{i,j}^m = \alpha \delta_{i,q}^m \times a_{i,q}^m, \tag{3-a}$$

$$\Delta b_i^m = \alpha \delta_{i,q}^m, \tag{3-b}$$

$$\delta_{i,q}^m = \left(\sum_{l=1}^{S_{m+1}} \delta_{l,q}^{m+1} \times w_{l,i}^{m+1} \right) \left(f^m \left(n_{i,q}^m \right) \right)', \quad m = 1, \ldots, M-1 \tag{3-c}$$

$$\delta_{i,q}^M = \left(t_i^q - a_{i,q}^m \right) \left(f^M \left(n_{i,q}^M \right) \right)', \tag{3-d}$$

$$w_{i,j}^m (k+1) = w_{i,j}^m (k) + \Delta w_{i,j}^m, \tag{4-a}$$

$$b_i^m (k+1) = b_i^m (k) + \Delta b_i^m, \tag{4-b}$$

where α is the learning rate and the so-called δ variables are defined as $\delta_{i,q}^m = \dfrac{\partial \hat{F}_q}{\partial n_{i,q}^m}$; it is known as the sensitivity of the performance index to the changes in the net input neuron i at layer m for pattern q.

The term backpropagation can be figured out as follows. Given the current interconnection weights of the neural network, the outputs of the neurons at the different layers are first calculated. Then the error is evaluated at the output layer. This error is backpropagated from the output layer towards the input layer by computing the sensitivity variables at each layer. Finally, because the δ variables are available, the neural network parameters are then adapted through equations (3). This process of parameter adaptation is repeated till some stopping criterion is fulfilled. The backpropagation algorithm is considered to have converged when the absolute rate of change in the mean squared error per epoch is good enough small. The process of backpropagation can be done as an on-line(pattern-by-pattern) mode as well as off-line (Batching) mode. In latter learning mode the weights are adjusted after running once through the whole patterns of the training set. Each iteration of the algorithm in this mode is called an epoch.

3. LEARNING AUTOMATA (LA)

This section introduces the basic idea of learning automata. The theory of learning automata is concerned with the analysis and synthesis of fixed or variable structure automata in a random environment. In this section we briefly describe the variable structure LA. For more information including fixed structure LA one may refer to [12].

Variable structure automata is represented by the sextuple $< X,\varphi,\alpha,P,G,T>$, where x is the input set, φ is the set of states of the automata, α is the action set, $\underline{p}(k)$ is the state probability vector governing the choice of the state at each stage k, and G is the output mapping. In this paper G is taken to be deterministic and one-to-one (i.e., the number of actions is equal to that of states and they are regarded to be synonymous). The learning algorithm, which is a recurrence relation, is denoted by T and it is used to modify the state probability vector, that is, it generates $\underline{p}(k+1)$ from $\underline{p}(k)$.

The environment in which the automata operate is represented by triple $<\alpha, x, D>$. The environment has random response characteristics. It's input is $\alpha(k)$, $\alpha(k)\in\{\alpha_1,\alpha_2,....,\alpha_M\}$ and it's output belongs to the set X. In a P-model which is our interest $X(k)\in\{0,1\}$ and the environment is characterized by the success probability vector $D=(d_1,d_2,...,d_M)$,with $d_i = \Pr ob[X(k)=1|\alpha(k)=\alpha_i]$.

If the $d_i^{,s}$ don't depend on k, the environment is said to be stationary, otherwise it is non-stationary. If $E_1,E_2,....E_n$ are stationary environments which can themselves be considered as states of a Markov process, we will have the Markovian switching Environment. If the $d_i(k)^{,s}$ vary periodically with time, the environment is called periodic environment.

In the P-model, the random variable X(k) takes only two values X(k)=1, known as the success input with probability $d_i(k)$ and X(k)=0 known as the penalty input with probability $c_i(k)=1-d_i(k)$.

Figure 3 represents a feedback connection of an automata and an environment. The environment models the system that communicates with the automata and supplied it with information. In the context of neural networks the random environment represents the realization of an optimization function to be minimized, at each time. The automata collects data from the environment and process it in order to achieve the desired performance.

The actions of the automata form the inputs to the environment and the responses of the environment in turns are the inputs to the automata. In automata the probability distribution is adjusted using an adjusted mechanism to reach the desired goal. This is the heat of the automata. The automata take one of a set of actions based on a set of corresponding probabilities and the environment responds to the automata by indicating success or failure. The automata then adjust its behavior based on this feedback by altering the set of probabilities. This is repeated until optimal performance of the automata is achieved.

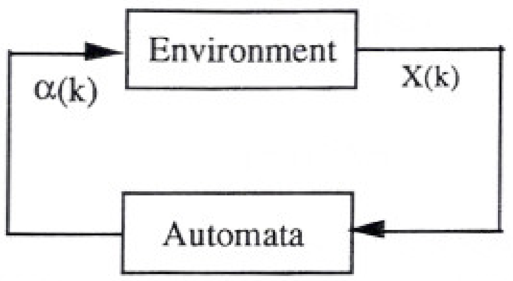

Figure 2. Automata and Environment

It is evident from the description of the LA that the crucial factor affecting the performance is the learning algorithm for updating the action probabilities. Various learning algorithms have been reported in the literature[12-13]. The Linear Reward-Inaction ($L_{R\text{-}I}$) scheme is one of the earliest schemes. Let α_i be the action chosen at time k as a sample realization from distribution p(k). In an ($L_{R\text{-}I}$) scheme the recurrence equation for updating p is defined as:

For X(k)=1,

$$p_i(k+1) = p_i(k) + \theta(1 - p_i(k)),$$
$$p_j(k+1) = p_j(k) - \theta p_j(k), \quad j \neq i,$$

and for X(k)=0,

$$p_j(k+1) = p_j(k).$$

The parameter θ is called step length; it determines the amount of increase (decrease) of the action probabilities, i.e., the probability $p_i(k)$ is increased (decreased) if the action $\alpha_i\left(\alpha_j\right)$ is performed and it results in a favorable response (X(k)=1). The probabilities are not changed if failure occurs. In an ($L_{R\text{-}P}$) scheme the recurrence equation for updating p is defined as:

For X(k)=0,

$$p_i(k+1) = p_i(k) + \theta(1 - p_i(k)),$$
$$p_j(k+1) = p_j(k) - \theta p_j(k), \quad j \neq i,$$

and for X(k)=1,

$$p_j(k+1) = (1-\gamma)p_j(k)$$
$$p_i(k+1) = p_i(k) + \gamma(1 - p_i(k)), \; i \neq j$$

The parameters γ and θ represent step lengths; they determine the amount of increase (decrease) of the action probabilities, i.e., the probability $p_j(k)$ is decreased at stage k+1 if the action α_i is performed and it results in a favorable response (X(k)=1) and increased by an amount proportional to $\left(1-p_j(k)\right)$ for an unfavorable response.

4. INTERCONNECTION OF LA's INTO NN

After the brief reviews of both layered neural network topology and learning automata along with the notation and nomenclature presented in the last two sections, this section describes the interconnection of the LAs into the neural networks. Figure 3 represents the overall interconnected system. The top part represents the environment, which is the neural networks with backpropagation algorithm, and the rest represents the learning automata mechanism (compare with Figure 2).

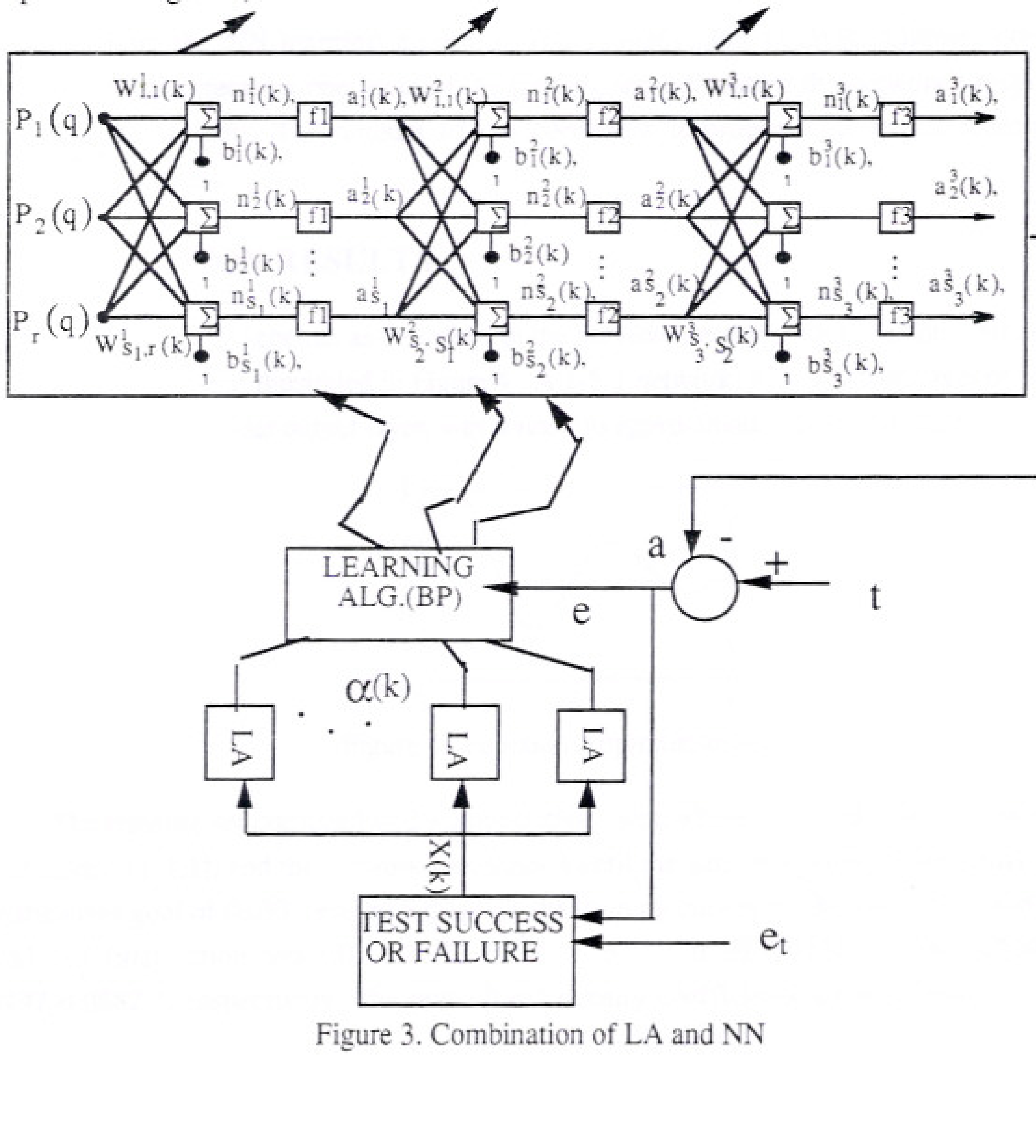

Figure 3. Combination of LA and NN

In the above figure **P**, **a** , **t** and **e** stand for the input, actual output, desired output, and the error, respectively. The $\mathbf{e_t}$ as a threshold is used to provide binary input for LAs and the $\alpha_{k's}$ are the step lengths for the NN-learning algorithm chosen the automata as their actions.

The following procedure describes the behavior of this augmented network.

1. Propagate the input forward using Eqns (1).
2 Propagate the sensitivities back using (3).
3. Use the LAs to choose the proper step lengths for NN-learning algorithm and update the weights and offsets using (4).
4. Steps 1-3 are repeated till some error goal is reached.

In short the NN-learning algorithm (backpropagation training scheme) consists of presenting at each iteration one input of the training set, computing the network's response and correcting the weights. The amount of the correction is proportional to the learning rates associated to neurons.

5. SIMULATION RESULTS

The (LA+ NN) scheme, as described in the previous section, was tested on a function approximation problem illustrated in Figure 4. A 1-5-1 network, with a hidden layer of sigmoid nonlinearities and a linear output layer, was trained to approximate this sinusoidal function.

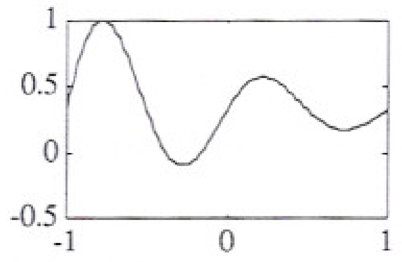

Figure 4. Function Approximation

The training set consisted of 20 input/output pairs, where the input values were scattered in the interval [-1,1]; and the network was trained until the sum of squares of the errors was less that the error goal of 0.033. Figure 5 displays the training curves for the variable action set with initial and final action sets {.01 .03 .05 .08 .1 .3 .5}, {0.0189 0.0424 0.0388 0.0329 0.0378 0.0497 0.0882 }, respectively. The reward and penalty coefficients are both equal to 0.1. The final action probabilities become {0.0384 0.4304 0.3441 0.0041 0.0050 0.1725 0.0054}. After 137 epochs the training took 385584 flops with the sum of squared error of 0.0324.

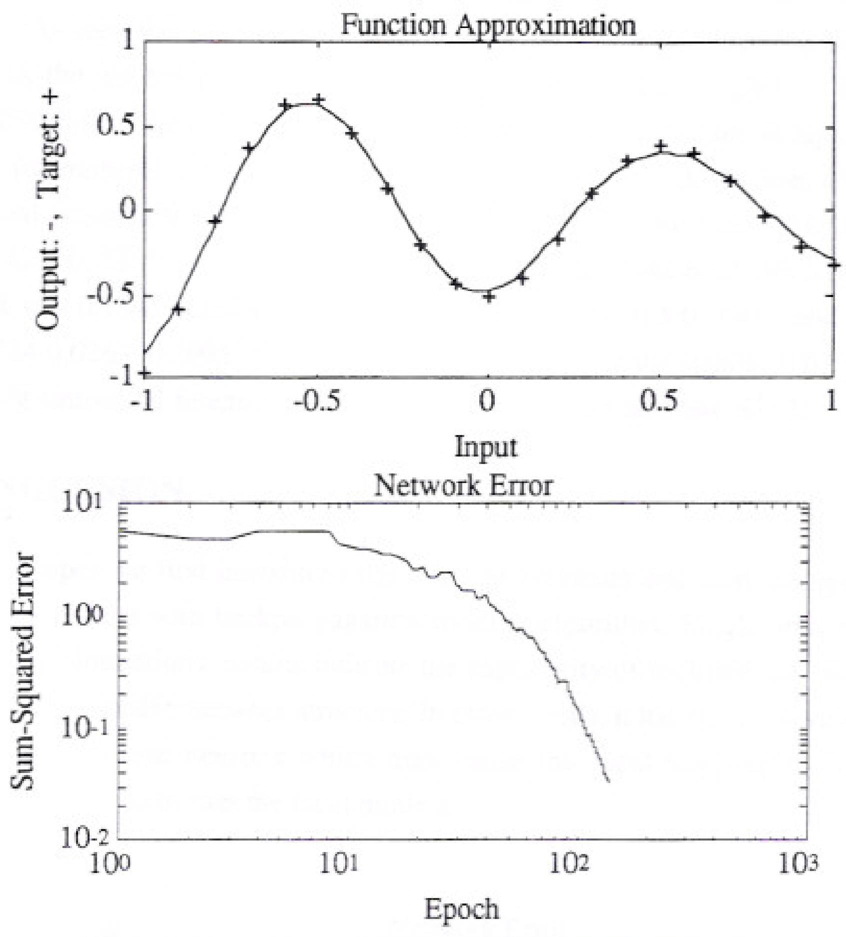

Figure 5. Learning Curve for Variable Center of Actions

Figure 6 shows the learning curve for the case in which multiple LA have been applied to the same network. Here, for each layer we assigned a separate LA to achieve the best step size. The initial actions for layers were selected as before. The final actions for layer one and layer two are {0.0177 0.0351 0.0282 0.0496 0.0266 0.0366 0.0502} and {0.0280 0.0393 0.0390 0.0336 0.0323 0.0288 0.0319} with the final action probabilities of {0.1052 0.1811 0.1184 0.2200 0.1755 0.1006 0.0992} and {0.4438 0.1569 0.0462 0.0237 0.1233 0.1070 0.0992}, respectively. The reward and penalty coefficients are both equal to 0.01 for first layer and 0.1 for the second layer. After 134 epochs the training took 392242 flops with the sum of squared error of 0.0329.

To highlight the effect of multiple LA on the performance of the augmented system, the number of hidden neurons has been reduced from 5 to 3 and all the other parameters are held fixed. Figures 7 and 8 show the learning curves for both single LA and multiple LA cases, respectively. As seen the multiple LA makes the neural networks escape the local minima while for single LA the net has been trapped into local minimum. For single LA, after 2000 epochs about 3495030 flops , the network operates inadequately with final sum of squared error of 2.35. In contrast, for multiple LA the network operates adequately with the sum of squared error of .033. The final actions for the first and the second Layer the network are {0.0112 0.0111 0.0115 0.0190 0.0142 0.0153 0.0142} and {0.0238 0.0131 0.0086 0.0056 0.0193 0.0234 0.0125} with probabilities of {0.1845 0.1324 0.1083 0.0834 0.1461 0.2413 0.1040} and {0.1890 0.2153 0.2094 0.1234 0.0267 0.2095 0.0266}, receptively. Note that the standard BP never could learn the underlying sinusoidal function with this number of hidden neurons, $S^1=3$.

6. CONCLUSION

In this paper we first introduced the learning automata and then incorporated it into the multi-layer neural nets with backpropagation training algorithm. Single and multiple LA have been tested. The simulations results indicate the superiority of multiple LA to the single LA in the sense of having smaller network structure. In other words, it has been shown that the multiple LA scheme allows those neurons which may cause the local trapping to choose their own learning rates in order to bypass the local minima.

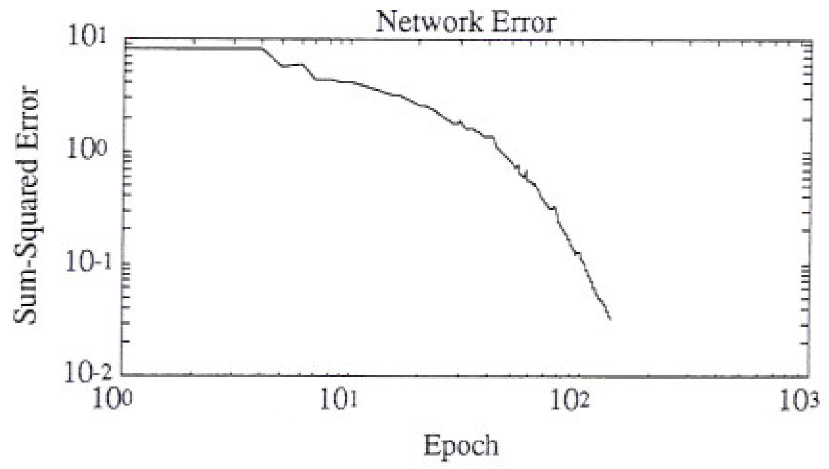

Figure 6. Learning Curve for Multiple LA's

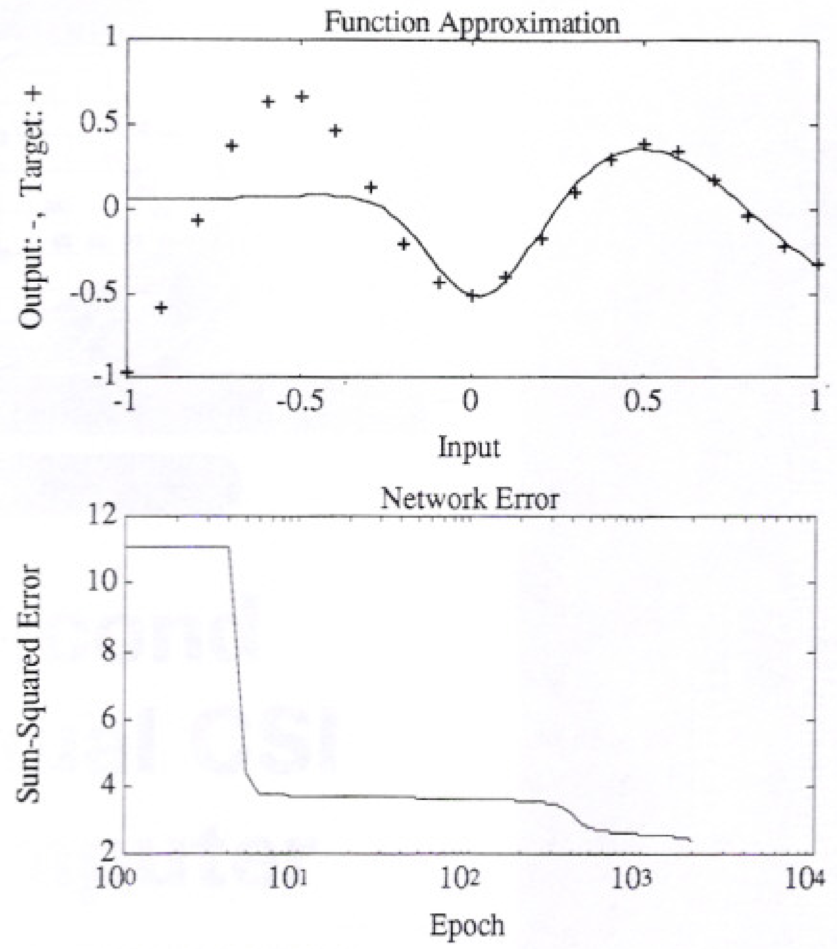

Figure 7. Learning Curve (Single LA), Trapped in Local Minimum

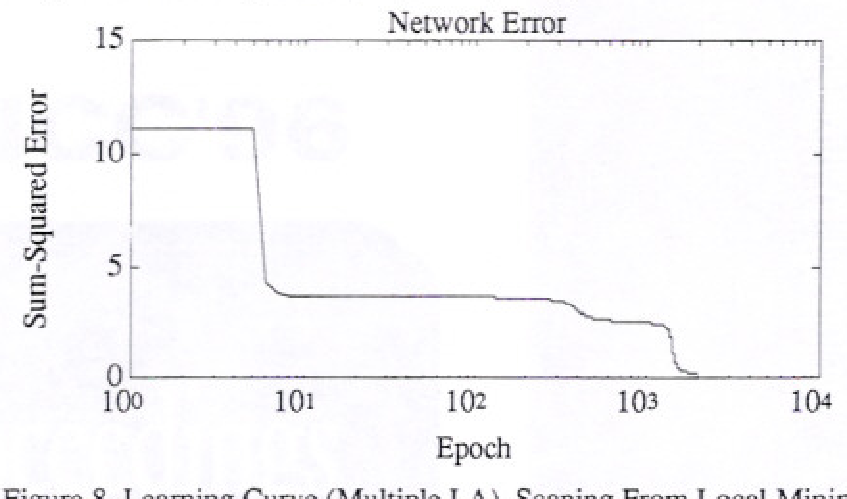

Figure 8. Learning Curve (Multiple LA), Scaping From Local Minima

REFERENCES

[1] Menhaj M. B. and M. R. Meybodi," A novel learning Scheme for Feedforward Neural Nets", ICEE-95., University of Science and Tech., Tehran-Iran

[2] Rosenblatt, F, "The Perceptron : A Probabilistic Model for Information Storage & Organization in the Brain," Psychological Review 65, PP. 386-408.

[3] Rumelhartmt, D. G. Hintonm R. Williams,"Learning Representations by Back-propagation Errors,"Nature 323:533-536, 1986.

[4] Gail A. Carpenter & S. Grossberg: Associative Learning, Adaptive Pattern Recognition & Cooperative Decision Making by Neural Networks, Hybrid & Optical Computing Spie, 1986.

[5] Kohonen T., Self-organization & Associative Memory, Berlin-springer-verlag, 1987.

[6] Widrow & R. Winter,"Neural Nets for Adaptive Filtering & Adaptive Patern Recognition," IEEE Control magazine, vol 21, no.3, PP. 25-39, March 1988.

[7] Grossberg, S. ,"Adaptive Pattern Classification and Universal recoding: I. Parallel Development & Coding of Neural Feature Detectors," Biological Cybernetics 23: PP. 121-134, 1976.

[8] Hopfield J.J, "Neural Networks & Physical Systems with Emergent Collective Computational Abilities," Proc. Natl. Acad. Sci. , vol. 79, PP. 2554-2558, 1982.

[9] Hagan, M. H., M. B. Menhaj, "Training Feedforward Networks with Marquardt Algorithm ."To appear in IEEE Trans. on Neural Networks , 1994.

[10] Menhaj, M.B, and M. H. Hagan, "Rapid Learning via Modified Backpropagation Algorithms for Multi-layer Feedforward Neural Nets." Submitted to IEEE Trans. on Neural Networks , 1994.

[11] Hush, D. R, and B. G. Horne, "Progress in Supervised Neural Networks: What's New Since Lippmann?" IEEE Signal Processing Magazine, January 1993.

[12] Narendra K. S. and M. A. L. Thathachar, Learning Automata an Introduction., Prentice Hall, 1989.

[13] Meybodi, M.R. and S. Lakshmivarahan, "ε -Optimality of a Class of Absorbing Barrier Learning Algorithms. " Information Science, 28, 1982, PP, 1-20.

Second Annual CSI Computer Conference

CSICC'96

Proceedings

Editor:
Reza Safabakhsh

دانشگاه صنعتی امیرکبیر

December 24- 26, 1996
Computer Engineering Department Amirkabir University of Technology
Tehran, I. R. Iran

انجمن کامپیوتر ایران
Computer Society of Iran