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

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

FLEXIBLE SIGMODAL TYPE FUNCTIONS FOR NEURAL NETS USING GAME OF AUTOMATA

Mohammad B. Menhaj
Electrical Engineering department

Mohammad R. Meybodi
Computer Engineering department

Amirkabir University of Technology
Tehran-Iran

KEYWORDS:
NEURAL NETWORKS, SUPERVISED LEARNING, LEARNING AUTOMATA, GAME OF AUTOMATA

ABSTRACT

In this paper we present an application of the learning automaton approach to allow the sigmoidal nonlinearity to be more flexible in order to adopt itself to the undelying problem. By interconnection of automata to the feedforward neural networks, we apply the learning automaton scheme for adjusting the parameters of sigmoidal functions based on the observation of random response of the neural networks. Simulation results show that flexible sigmoidal function by using the learning automata approach enhances the ability of neural net learning algorithm.

1. INTRODUCTION

Artificial neural networks as non-linear model free adaptive dynamic systems offer a practical approach for real problems that sequentially computational devices (serial processors such as digital computers) can not solve. For these problems there are no computational algorithms, or if there is one, the algorithmic solution is too slow. Neural networks with their inherent parallelism need not to be explicitly programmed with any algorithm which would solve the problem; instead they will learn to solve the problem at hand. In fact, they learn data transformation. Although this remarkable capability of multi-layer feedforward neural networks made them very popular, they suffer from some drawbacks such as need for training data set and training period. The latter is the major drawback which is directly related to the speed of convergence of the training algorithms. This becomes a limiting factor specially for the real problems which require high degree of matching accuracy.

The standard on-line backward error propagation is the most useful learning algorithm for multi-layer feedforward neural networks. Since the backpropagation learning algorithm [1] was first popularized, there has been considerable research on methods to accelerate the convergence of the algorithm. This research falls roughly into two categories. The first category involves the development of ad hoc techniques (e.g., [2]-[5]). These techniques include such ideas as varying the learning rate, using momentum and re-scaling variables. Another category of research has focused on standard numerical optimization techniques such as conjugate gradient or quasi-Newton methods.(e.g., [6]-[8]).

The computation of the sensitivity for each neuron of the multilayer feedforward net requires that the derivative of the activation function associated with that neuron exists. For this to be so, the activation function has to be continuous. A popular example of differentiable nonlinear function which is commonly used in MLP nets is sigmoidal type functions. In all of the methods mentioned above, it was assumed that the sigmoidal nonlinearity and the saturation limits are inflexible; it means that the coefficient of the exponent of the exponential term and the scaling factor are fixed.

Knowing that the sigmoidal type functions, such as log-sigmoid and hyperbolic tangent, are commonly used as activation functions in MLP nets, one may consider their coefficients to determine the steep of function's linearity. By doing so, we gain much flexibility in order to move the net inputs of the sigmoidal functions near to their active regions where the associated gradients are not very close to zero. This makes the BP algorithm not be trapped to some points in the network parameters space where the BP algorithm would effectively stop, even though it is not close to a local minimum point.

As mentioned earlier, one problem in convergence of standard BP algorithm occurs because of the shape of the sigmoid function. The derivative of the sigmoid becomes very small out on the tails. This will cause the gradient of the error function to be small if the sigmoid is shifted far outside the active region of the input to the function. Therefore, we want the center of each sigmoid to be inside the active region of the input function. In addition, for the same reason, we do not want the slope of the sigmoids to be too high. We want each sigmoide to be approximately linear over some subset of the active region of the net input. By considering the coefficients of sigmoidal type functions as variables, we may be able to enhance the range of linearity of the sigmoidal function while it is needed.

Theory of Learning Automaton (LA) has made significant progress in the last decades and has attracted considerable interest due to their potential applicability in many different real-time engineering problems[8]. The main idea behind the learning automaton approach is to provide a general framework for the design of large stochastic distributed systems. Games of automta deals with the behavior of a collection of automata operating in a random environment. Any game of automata consists of more than one automata and results in some outcome depends on the behavior of these automata. Since, in general, each automata as a different performance criterion, the outcome is valued differently by each automata. The extent to which the automata can communicate with one another, the information available to each automata, the causal relations between the actions of the automata and the outcome of the game along with any binding agreements that the automata can enter into, all determines the rules of the game.

In this paper we present an application of the learning automaton approach to allow the sigmoidal nonlinearity to be more flexible in order to adopt itself to the underlying problem. This has been done for the following two different cases:

Case I. In this case, by interconnection of automata to the feedforward neural network, we apply the learning automaton scheme for adjusting the parameters of sigmoidal functions . The parameter adaptation is accomplished based on the observation of random response of the neural networks.

Case II. In case I, we only considered a single automata to adjust the flexible coefficients of the sigmidal functions of neurons. Whereas, in this case we use a collection of automata to simultaneously adjust both the learning rate and flexible coefficient of sigmidal nonlinearites in order to increase the rate of convergence of neural net learning algorithm.

In order to show the strength of our approach, some simulations have been done. The results of simulations are very promising indicating that the flexible sigmoidal function by using the learning automata approach enhances the ability of neural net learning algorithm. The paper is organized as follows. Section 2 briefly presents the basic backpropagation algorithm. The learning automaton is then introduced in section 3. Section 4 introduces the theory of game of automata. Section 5 presents simulation results and discussions. Section 6 concludes the paper.

2. BACKPROPAGATION ALGORITHM

Given the multilayer feedforward network, such as the three-layer network of Figure 1 with the basic building block diagrammed in Figure 2 and assuming that the nonlinear functions in figure 1 are of sigmoidal shape functions either log-sigmoid or hyperbolic tangent which are respectively represented by the following two functions:

$$f^{l+1}\left(n_i^{l+1}\right)=\frac{1}{1+e^{-c_i^{l+1}\times n_i^{l+1}}},\text{ and } f^{l+1}\left(n_i^{l+1}\right)=\frac{e^{c_i^{l+1}\times n_i^{l+1}}-e^{-c_i^{l+1}\times n_i^{l+1}}}{e^{c_i^{l+1}\times n_i^{l+1}}+e^{-c_i^{l+1}\times n_i^{l+1}}},$$

where $c_i^{l+1}(k)$ is the steepness coefficient of the sigmoid nonlinear function of neuron i in layer l+1 at time k that determines the range of linearity of the sigmoidal function for a fixed n_i^{l+1}, for an M layer network the system equations become [9]

Forward Equations::

$$\underline{a}^{0}(k)=\underline{p}(q), \qquad q=1,2,...Q$$

$$\underline{a}^{l+1}(k)=f^{l+1}\left(W^{l+1}(k)\underline{a}^{l}(k)+\underline{b}^{l+1}(k);\underline{c}^{l+1}(k)\right), \qquad l=0,1,\ldots,M-1$$

$$\underline{a}(k)=\underline{a}^{M}(k).$$

Backward Equations:

$$\underline{S}^{M} = -\dot{F}^{M}(\underline{n}^{M})\left(\underline{t}_q-\underline{a}\left(k,\underline{p}(q)\right)\right).$$

$$\underline{S}^{l}(k) = \dot{F}^{l}(\underline{n}^{l})\left(W^{l+1}\right)^{T}\underline{S}^{l+1}(k).$$

Adjustment Equations:

$$\Delta w_{i,j}^{l}(k) = -\alpha(k)\, s_i^{l}(k)\, a_j^{l-1}(k),$$

$$\Delta b_i^{l}(k) = -\alpha(k)\, s_i^{l}(k),$$

where α(κ) is the learning rate at time instant k, $s_i^l(k)$ denotes the sensitivity of the performance index to changes in the net input of unit i in layer l at sampling time k, and

$$\dot{F}^{l}(\underline{n}^{l}) = \begin{bmatrix} \dot{f}^{l}(n_1^{l}) & 0 & . & . & 0 \\ 0 & \dot{f}^{l}(n_2^{l}) & 0 & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & . & . & \dot{f}^{l}(n_{S_l}^{l}) \end{bmatrix}, \quad \dot{f}^{l}(x) \equiv \frac{d f^{l}(x)}{d x}.$$

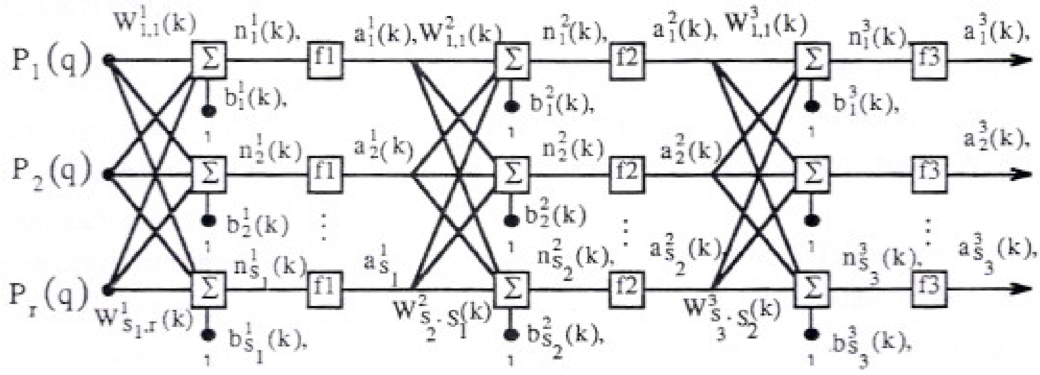
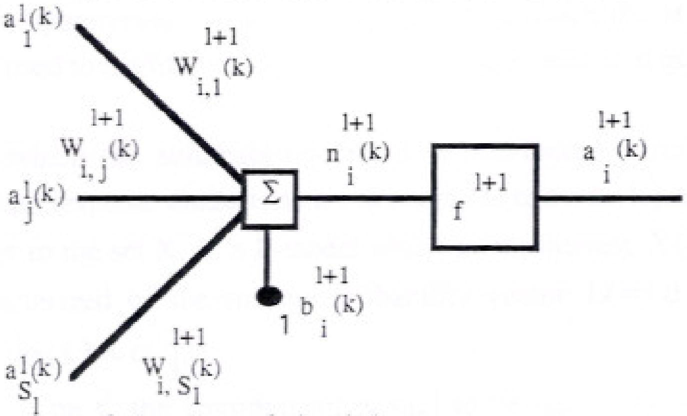


Figure 1. Three-Layer Feedforward Network



$$a_i^{l+1}(k) = f^{l+1}(n_i^{l+1}(k)).$$

$$n_i^{l+1}(k) = \sum_{j=1}^{S_l} w_{i,j}^{l+1}(k)a_j^l(k) + b_i^{l+1}(k)$$

Figure 2. Basic Network Computing Element

The overall learning algorithm now proceeds as follows: first, propagate the input forward using forward equations; next, propagate the sensitivities back using backward equations; and finally, update the weights and offsets using adjustment equations.

Knowing that the sigmoidal type functions, such as log-sigmoid and hyperbolic tangent, are commonly used as activation functions in MLP nets, we may consider their coefficients which determine the steep of function linearity. By doing so, we gain much flexibility in order to move the net inputs of the sigmoidal functions near to their active, better to say, linear regions where the associated gradients are not very close to zero. This makes the BP algorithm not be trapped to some points in the network parameters space where the BP algorithm would effectively stop, even though it is not close to a local minimum point.

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 [8].

Variable structure automata is represented by the sextuple < X,φ,α,P,G,T>, where x is the input set, φ is the set of states of the automata, α is the action set, 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 p(k+1) from p(k).

The environment in which the automata operate is represented by triple <α, x, D>. The environment has random response characteristics. It's input is α(k), $\alpha(k) \in \{\alpha_1, \alpha_2, \ldots.., \alpha_M\}$ 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 $D = (d_1, d_2, \ldots, d_M)$,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 $E_1, E_2, \ldots, E_n$ 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.

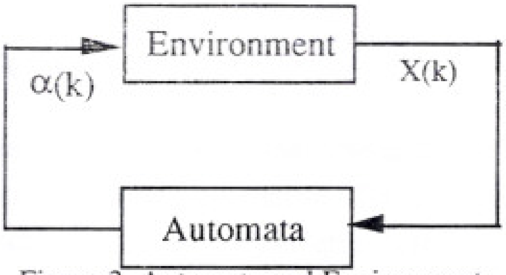


Figure 3. Automata and Environment

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[11-14]. The Linear Reward-Inaction (LR-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 (LR-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 $\alpha_i(\alpha_j)$ is performed and it results in a favorable response (X(k)=1). The probabilities are not changed if failure occurs. In an (LR-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. GAME OF AUTOMATA

Any game of automata consists of more than one automata and results in some outcome which depends on the behavior of these automata. The extent to which the automata can communicate with each other, the information available to each automata, the relation between the actions of the automata and the outcomes of the game along with any binding agreements that the automata can enter into, all determine the rules of the game. The complexity of the game can be classified in various ways depending on the number of players, the performance criteria (zero-sum or nonzero-sum) and the nature of communication used (cooperative or non-cooperative).

In this paper a cooperative game of two learning automata is used to find the best values for parameters α, learning rate, and γ , momentum factor. Two automata A_1 and A_2 involving in a cooperative game have the action sets

$$\left\{\alpha_i^1,\ i=1,2,\ldots,r_1\right\} \text{ and } \left\{\alpha_j^2,\ j=1,2,\ldots,r_2\right\},$$

respectively. The game can be represented by an $\left(r_1 \times r_2\right)$ payoff matrix D whose (i.j)-th element d_{ij}, represents the probability of success when the action pair $\left(\alpha_i^1, \alpha_j^2\right)$ is chosen.

For our application $d_{ij} = \text{prob}\left(e_t \leq T\right)$ when A1 chooses value of α_i^1 for the parameter α and value of α_j^2 for parameter of γ, where e_t is the error at time t and T is a threshold which is properly defined.

We assume $\underline{p}(n)=\left\{p_i(n),\ i=1,2,\ldots,r_1\right\}$and $\underline{q}(n)=\left\{q_i(n),\ i=1,2,\ldots,r_2\right\}$ be the action probability vectors of the two automata at stage n. Each automata operates independently and in a total ignorance of the other automata. Both automata receive identical payoff as the results of the game (selection of specific values of α and γ) at stage n. The goal of the game is to find the best values for α and γ in order to increase the likelihood of achieving global minima as much as possible. Each automata uses a learning algorithm of variable structure type for its choice resulting in the selection of an action at each play of the game. In matrix D, d_{ij} is called the local minimum if it is simultaneously the minimum of the i-th row and the j-th column. The global minimum is the minimum of all local minimum.

Studies have shown that under stationary environment (when matrix D is time independent) the convergence is to one of the local minima with probability of arbitrary close to one. In order to achieve global optimality with with probability of arbitrary close to one, the payoff matrix D must have a unique equilibrium point or restriction on the type of information available to the autamata must be relaxed. In fact, if actions chosen by the automata are made known to other automata, estimation algorithms can be applied and automata will converge the global minimum.

The payoff matrix for a two cooperative automata game operating in a neural net environment is time dependent and can be modeled as

$$D(t) = f(W(t-1), p(t-1), q(t-1)),$$

where D(t)is the payoff matrix at time t, W(t-1), P(t-1), and q(t-1) are the neural network adjustable parameters, the action probability vector for automata A_1 and the action probability for automata A_2, respectively.

The theory of two automata operating under such a complex time varying environment is under development by the authors. But simulation results have shown that the use of such a game helps the neural net to adaptively find good values for parameters α and γ in order to increase the probability of escaping the local minima.

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.

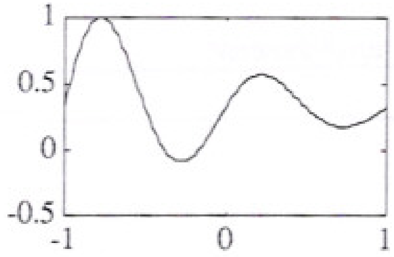


Figure 4. Function Approximation

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 curve for the standard BP with α=.01. After 5005 epochs the training took 13536475 flops with the sum of squared error of 0.0329.

Figure 6 shows the learning curve for the case in which variable learning rate BP has been applied to the same network. As we can see that after 2338 epochs the network could reach the error goal with total flops of 6331427. Figure 7 shows the response of the network when we applied LA for adaptive selection of the sigmoidal coefficients of neurons. Here the standard BP scheme has been used with fixed step size of α=0.01. As seen, the overall (LA+NN)-scheme performance is excellent. In this case, after 543 epochs the training took 2403100 flops with the total sum of squared error of .032.

Figure 8 represents the learning curve for the case in which we applied the cooperative game of automata scheme for adjustment of both learning rate, α, and the neuron's flexibility coefficients. In this case, the training took 681369 flops for 152 epochs with the sum of squared error of 0.0176. Figure 9 shows the case in which the sigmoidal coefficients are updated iteratively. Here, the network with standard BP scheme reaches the error goal after 2371 epochs with total 15880977 flops.

5. CONCLUSION

In this study another application of the learning automaton approach into the neural networks has been presented. This approach allows the sigmoidal nonlinearity to be more flexible in order to adopt itself to the underlying problem. By interconnection of automata to the feedforward neural networks, we apply the learning automaton scheme for adjusting the parameters of sigmoidal functions based on the observation of random response of the neural networks. Simulation results show that flexible sigmoidal function by using the learning automata approach enhances the ability of neural net learning algorithm.

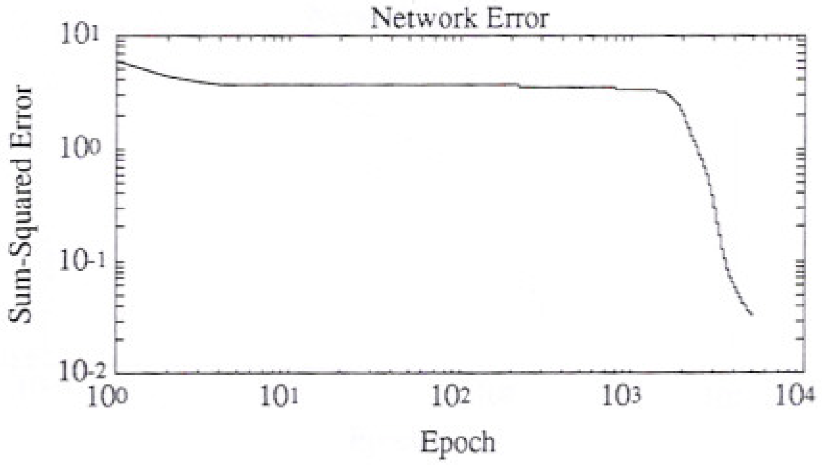


Figure 5. Learning Curve for Std. BP, α=0.01.

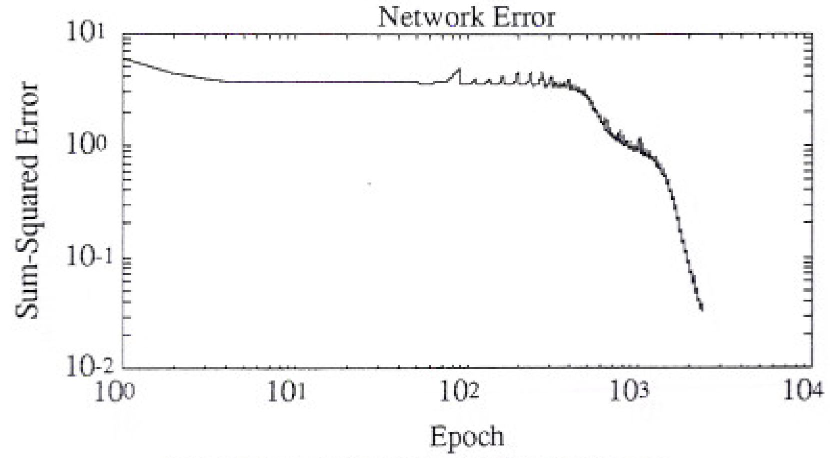


Figure 6. Learning Curve for variable α

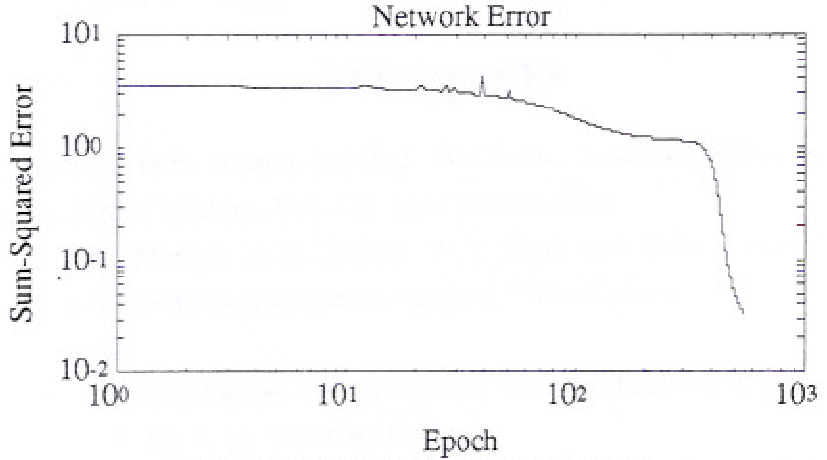


Figure 7. Learning Curve for single LA

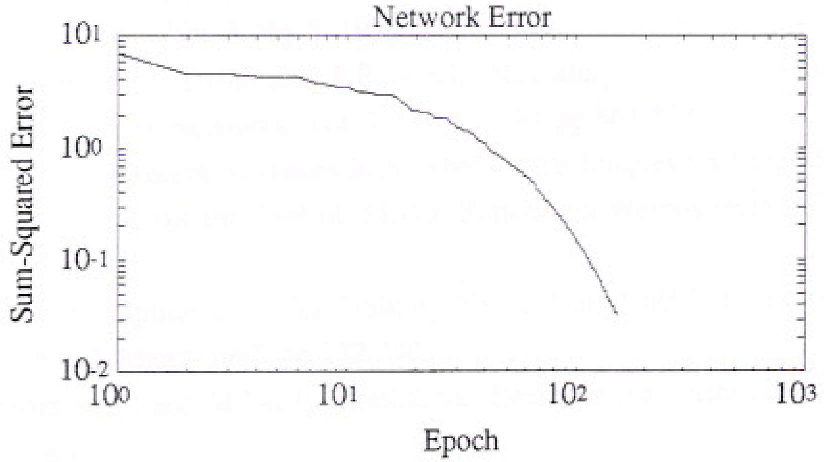


Figure 8. Learning Curve for Cooperative Game of Automata

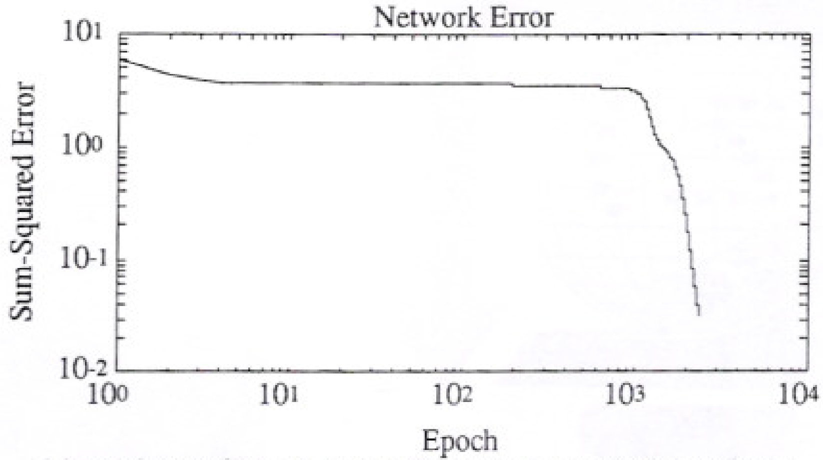


Figure 9. The Sigmoidal Coefficients are updated Iteratively

REFERENCES

[1] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning Representations by Back-Propagating Errors," Nature, Vol. 323, pp 533-536, 1986.

[2] T.P. Vogl, J.K. Mangis, A.K. Zigler, W.T. Zink and D.L. Alkon, "Accelerating the Convergence of the Backpropagation Method," Bio. Cybern., Vol. 59, pp 256-264, Sept. 1988.

[3] R.A. Jacobs, "Increased Rates of Convergence Through Learning Rate Adaptation," Neural Networks, Vol. 1, No. 4, pp 295-308, 1988.

[4] T. Tollenaere, "SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties," Neural Networks, Vol. 3, No. 5, 1990, pp 561-573.

[5] A.K. Rigler, J.M. Irvine and T.P. Vogl, "Rescaling of Variables in Back Propagation Learning," Neural Networks, Vol. 3, No. 5, 1990, pp 561-573.

[6] D.F. Shanno, "Recent Advances in Numberical Techniques for Large-Scale Optimization," in Neural Networks for Control, Miller, Sutton and Werbos (Eds.), Cambridge MA:MIT Press, 1990.

[7] E. Barnard, "Optimization for Training Neural Nets," IEEE Trans. on Neural Networks, Vol. 3, No. 2, March 1992, pp 232-240.

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

[9] M. B. Menhaj, Lecture Notes on Neural Networks, Amirkabir University of Tech., Dept. EE, Spring 1994, Tehran-Iran.

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