Adaptation of Parameters of BP Algorithm Using Learning Automata

Hamid Beigy
Computer Eng. Department
Amirkabir University of Technology
Tehran, Iran
beigy@ce.aku.ac.ir

M. R. Meybodi
Computer Eng. Department
Amirkabir University of Technology
Tehran, Iran
meybodi@ce.aku.ac.ir

Abstract

Backpropagation (BP) algorithm is a systematic method for training multilayer neural networks. Despite of the many successful applications of backpropagation, it has many drawbacks. For complex problems it may require a long time to train the networks, and it may not train at all. Long training time can be the result of the non-optimal parameters. It is not easy to choose appropriate value of the parameters for a particular problem. In this paper, by interconnection of fixed structure learning automata (FSLA) to the feedforward neural networks, we apply learning automata scheme for adjusting these parameters based on the observation of random response of neural networks. The main motivation in using learning automata as an adaptation algorithm is to use its capability of global optimization when dealing with multi-modal surface. The feasibility of proposed method is shown through simulations on three learning problems: exclusive-or, encoding problem, and digit recognition. The simulation results show that the adaptation of these parameters using this method not only increases the convergence rate of learning but it increases the likelihood of escaping from the local minima.

Keywords: Neural Network, Backpropagation, Learning Automata, Momentum Factor, Steepness Parameter

1 Introduction

Error backpropagation training algorithm (BP) which is an iterative gradient descent algorithm is a simple way to train multilayer feedforward neural networks [12]. The backpropagation algorithm is based on the gradient descent rule:

$$\Delta w_{jk}(n) = -\alpha \frac{\partial E}{\partial w_{jk}} + \mu \times \Delta w_{jk}(n-1) \qquad (1)$$

where w_{jk} is the weight on the connection outgoing from the unit j and entering the unit k, α, μ, and n are learning rate, momentum factor, and time index, respectively. In the BP framework α and μ are constant and E is defined as:

$$E(n) = \frac{1}{2} \sum_{p=1}^{pattern} \sum_{j=1}^{outputs} (T_{p,j} - O_{p,j})^2 \qquad (2)$$

Where $T_{p,j}$ and $O_{p,j}$ are desired and actual outputs for pattern p at output node j and the index p varies on the training set. In the BP algorithm framework, each computational unit compute the same activation function. The computation of the sensitivity for each neuron requires the derivative of activation function, therefore this function must be continuous. The activation function is normally a sigmoid function chosen between the two functions $f(x) = \frac{1}{1+e^{-\lambda x}}$ and $f(x) = \frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$. The steepness parameter λ determines the active region (region in which the derivative of sigmoid function is not very small) of activation function. As the steepness parameter decreases from positive infinity to zero the sigmoid function changes from a unit step function to constant value 0.5.

Several researchers have investigated the effect of adaptation of momentum factor and steepness parameter on the performance of BP algorithm. In [7] the steepness of every neuron is adjusted such that the average distance of the two closest data points from the dividing hyper-plane is attained and in [13] the steepness of every neuron is adjusted by gradient descent algorithm. In [6] the momentum factor is adjusted in order to cancel the introduced noise (which is as the result of misadjustment of momentum factor) and to retain the speed up as well as convergence. In [3] the momentum factor is considered as a function of gradient and in [5] the error function is divided to five regions and in every region the momentum factor and learning rate are adjusted differently. In [14] the mean square error (MSE) is considered as a function of learning rate and momentum factor and these parameters are adjusted to minimize the MSE.

Often the mean-square error surfaces for backpropagation algorithm are multi-modal. The learning automata is

known to have well established mathematical foundation and global optimization capability [11]. This latter capability of learning automata can be used fruitfully to search a multi-modal mean-square error surface. Variable structure learning automata (VSLA) have been used to find the appropriate value for different parameters of BP learning algorithm including learning rate [2] , steepness parameter [8], and momentum factor [9] [1]. In this paper, we present the application of the fixed structure learning automata (FSLA) for appropriate selection of the momentum factor and steepness parameter of BP algorithm in order to achieve higher rate of convergence and also increase the probability of escaping from the local minima. The feasibility of proposed method is shown through simulations on three learning problems: exclusive-or, encoding problem, and digit recognition. These problem are chosen because they posses different surfaces and collectively present an environment that is suitable to determine the effect of proposed method. Simulation on these problems show that the adaptation of momentum factor and steepness parameter using this method not only increases the convergence rate but it increases the likelihood of bypassing the local minima. Also simulations show that FSLA approach for adaptation of BP parameters performs much better than VSLA approach reported in [9] [8].

The rest of the paper is organized as follows: The learning automata is introduced in section 2. Section 3 presents the proposed method. Simulation results and discussion are given in section 4. Section 5 concludes the paper.

2 Learning Automata

Learning automata can be classified into two main families, fixed and variable structure learning automata [11]. Examples of the FSLA type which we use in this paper are the Tsetline, Krinsky, and Krylov automata. A fixed structure automata is quintuple $< \alpha, \phi, \beta, F, G >$ where $\alpha = (\alpha_1, ..., \alpha_r)$ is the set of actions, $\phi = (\phi_1, ..., \phi_s)$ is the set of states, $\beta = 0, 1$ is the set of inputs where $\beta = 1$ represents a penalty and $\beta = 0$ a reward, $F : \phi \times \beta \rightarrow \phi$ is transition map and $G : \phi \rightarrow \alpha$ is the output map. F and G may be stochastic.

The selected action serves as the input to the environment which in turn emits a stochastic response $\beta(n)$ at the time n. $\beta(n)$ is an element of $\beta = 0, 1$ and is the feedback response of the environment to the automata. The environment penalize (i.e. $\beta(n) = 1$) the automata with the penalty probability c_i, which is the action dependent. On the basis of the response $\beta(n)$, the state of the automata is $\phi(n)$ is updated and a new action chosen at (n+1). Note that the c_i are unknown initially and it is desired that as a result of the interaction between the automata and the environment arrives at the action which presents it with the minimum penalty response in an expected sense.

Variable structure learning automata is represented by sextuple $< \beta, \phi, \alpha, P, G, T >$, where β a set of inputs actions, ϕ is a set of internal states, α a set of outputs, P denotes the state probability vector governing the choice of the state at each stage k, G is the output mapping, and T is learning algorithm. The learning algorithm is a recurrence relation and is used to modify the state probability vector. It is evident that the crucial factor affecting the performance of the variable structure learning automata, is learning algorithm for updating the action probabilities. Various learning algorithms have been reported in the literature [10]. Let α_i be the action chosen at time k as a sample realization from distribution $p(k)$.

The linear reward-penalty algorithm (L_{R-P}) is one of the earliest schemes. In this scheme the recurrence equation for updating p is defined as

$$p_j(k+1) = \begin{cases} p_j(k) + \theta \times (1 - p_j(k)) & \text{if } i = j \\ (1 - \theta) \times p_j(k) & \text{if } i \neq j \end{cases} \quad (3)$$

if $\beta(k) = 0$ and

$$p_j(k+1) = \begin{cases} p_j(k) \times (1 - \gamma) & \text{if } i = j \\ \frac{\gamma}{r-1} + (1 - \gamma) p_j(k) & \text{if } i \neq j \end{cases} \quad (4)$$

if $\beta(k) = 1$. The parameters θ and γ represent step lengths. They determines the amount of increase (decreases) of the action probabilities.

3 The Proposed Method

In our proposed method, we use the fixed-structure learning automata for adjusting the momentum factor and steepness parameter. The interconnection of learning automata and neural network is shown in figure 1. The neural network is the environment for the learning automata. The learning automata according to the amount of the error received from neural network adjusts the values of parameters of the back-propagation algorithm. The actions of the automata correspond to the values of the momentum factor (or steepness parameter) and input to the automata is some function of the error in the output of neural network.

At the beginning of each epoch of BP algorithm, the learning automata selects one of its action (in the fixed structure learning automata action is selected by the means of output function G, and in variable structure learning automata, the action is selected by a sample realization of probability vector p). The value of selected action is used in BP algorithm for that epoch. The response of the environment which is given to the learning automata is a function

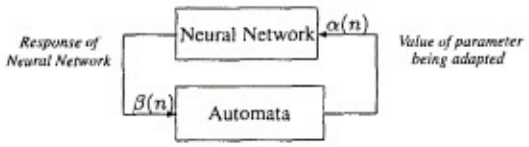

Figure 1. Automata-neural network connection

of the mean square error as explained below. In the kth epoch, the average of mean square error in past W epoch is computed by equation 5. We call W the window size.

$$MSE_W(k) = \frac{1}{W}\sum_{m=1}^{W} MSE(k-m) \qquad (5)$$

where $MSE(n)$ and $MSE_W(k)$ denote mean square error in nth epoch, and average mean square error in the past W epoch, respectively. The $MSE(k)$ compared with $MSE_W(k)$ and automata receives penalty from the environment if $MSE_W(k) - MSE(k)$ is less than a threshold and receives reward otherwise. Now the response of the environment (input to the learning automata) can be formulated as follows.

$$\beta(n) = \begin{cases} 0 & \text{if } MSE_W(n) - MSE(n) \leq T \\ 1 & \text{if } MSE_W(n) - MSE(n) > T \end{cases} \qquad (6)$$

At the beginning of the first epoch the action of the learning automata is selected randomly from the set of allowable actions.

The algorithms given later in this paper are backpropagation algorithm in which the learning automata is responsible for the adaptation of the BP parameters. In this algorithm at each iteration one input of the training set is presented to the neural networks, then the networks response is computed and the weights are corrected. The weights correction is applied at the end of each epoch. The amount of the correction is proportional to the BP parameters. Using the learning automata as a adaptation technique, the search for optimum values for the BP parameters is carried out in probability space rather than parameter space, and therefore this gives the algorithm the ability to locate the global optimum.

The simulations are carried out for two algorithms for adaptation of momentum factor and steepness parameter. In the first algorithm a single learning automata is responsible for determination of the BP parameter for the whole network, whereas in the second algorithm a separate learning automata has been used for each layer(hidden and output layers). Simulation results show that by using separate learning automata for each layer of the network not only the performance of the network improves over the case where we use a single automata, but it increases the likelihood of bypassing the local minima. These two algorithms have been tested on several problems and the results are presented in the section 4.

Simultaneous adaptation of momentum factor and steepness parameter: The rate of convergence and the stability of the training algorithm can be improved if both momentum factor and steepness parameter are adapted simultaneously. Two algorithms used for simultaneous adaptation of momentum factor and steepness parameter. In the first algorithm the network uses one automata to adjust the momentum factor and another automata to adjust the steepness parameter. Both automata work simultaneously to adjust the momentum factor and steepness parameter. In the second algorithm the network uses two pairs of automata, the first pair of automata (one for each layer) is responsible for adjusting the steepness parameter, and the second pair (one for each layer) is responsible for adjusting the momentum factor. These four automata work simultaneously to adapt steepness parameter and momentum factor. These two algorithms have been tested on several problems and the results are presented in the section IV.

4 Simulation

In order to evaluate the performance of the proposed method simulations are carried out on three learning problems: exclusive-or, encoding problem, and digit recognition. The results are compared with results obtained from standard BP and variable structure learning automata based algorithm reported in [9][8][1] These problems are chosen because they have different error surfaces and collectively present an environment that is suitable to determine the effect of proposed method. Actions of the learning automata in these simulations are selected in interval [0, 1] with equal distance. That is, the value of the ith action of learning automata with K actions is chosen to be $\frac{i}{K}$. For the sake of convenience in presentation, we use **automata (K, N)** to refer to the fixed structure learning automata, automata with K actions and memory depth of N. For all simulations reported in this paper, the same values of BP parameters are used for all experimentation of different algorithms, except for parameters which are being adapted by the algorithm.

1- XOR: The network architecture used for solving this problem consist of 2 input units, 2 hidden units, and 1 output unit [12]. Figure 3 shows the effectiveness of using FSLA and VSLA on the adaptation of momentum factor and figure 3 shows the effectiveness of FSLA and VSLA on the adaptation of steepness parameter. For automata in figure 2 the threshold of 0.001 and window size of 1 and for figure 3 the threshold of 0.0001 and window size of 1 are chosen. For linear reward-penalty automata the reward and

penalty coefficients are 0.001 and 0.0001, respectively.

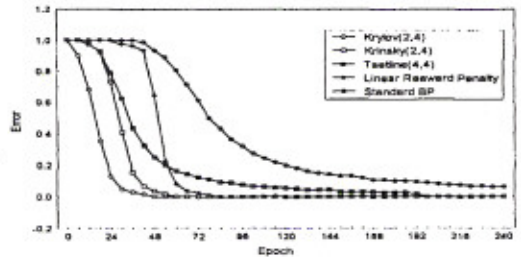

Figure 2. Adaptation of momentum factor

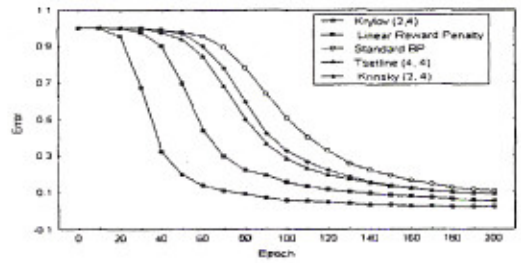

Figure 3. Adaptation of steepness parameter

2-Encoding Problem: In this problem a set of orthogonal input patterns are mapped to a set of orthogonal output patterns through a small set of hidden units [12]. The network architecture used for solving this problem consist of 8 input units, 3 hidden units, and 8 output units. Figures 4 and 5 show the effectiveness of using FSLA and VSLA on the adaptation of momentum factor and steepness parameter respectively. For automata in these figures the threshold of 0.01 and window size of 1 are chosen. For linear reward-penalty automata the reward and penalty coefficients are 0.001 and 0.0001, respectively.

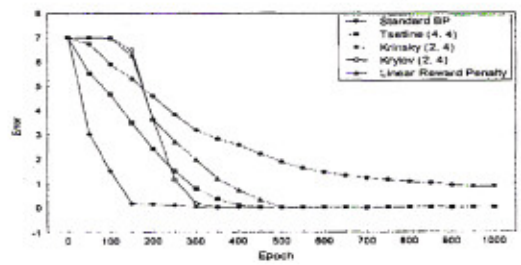

Figure 4. Adaptation of momentum factor

3- 8 × 8 Dot Numeric Font Learning: We have ten numbers 0, ... , 9, and each represented by a 8 × 8 grid of black

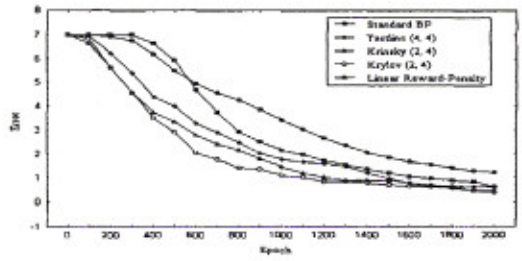

Figure 5. Adaptation of steepness parameter

and white dot as shown in figure 6 [13]. The network must learn to distinguish these classes. For this problem a network consists of 64 input units which are connected to 6 hidden units which are connected to 10 output units is used.

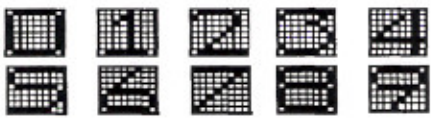

Figure 6. The training set of neural network

Figures 7 and 8 show the effectiveness of using FSLA and VSLA on the adaptation of momentum factor and steepness parameter, respectively. For automata in these figures the threshold of 0.01 and window size of 1 are chosen. For linear reward-penalty automata the reward and penalty coefficients are 0.001 and 0.0001, respectively.

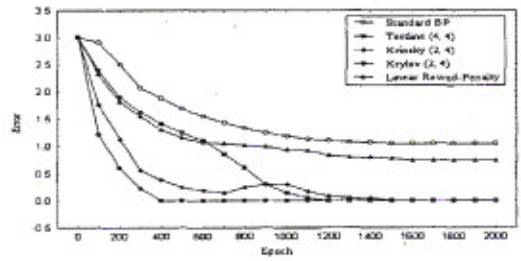

Figure 7. Adaptation of momentum factor

The results obtained when a single learning automata is assigned to the whole network for adjusting the momentum factor are presented in table 1. For all automata in this simulation the threshold of 0.01 and window size of 1 is chosen. The error of standard BP after 5000 epoch is 0.1675668 .

Table 2 shows the results obtained when a single learning automata assigned to whole network for adjusting the steepness parameter. For all automata in this simulation the

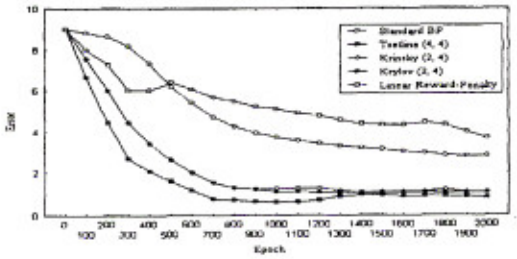

Figure 8. Adaptation of steepness parameter for digit problem

Table 1. Simulation results for digit problem

Learning Automata	Final Mean Square Error	Epoch for Error Goal = 0.01
Tsetline (4, 4)	0.0099858	3289
TsetlineG (2,4)	0.0099850	564
Krinsky (2, 4)	0.0099918	1101
Krylov (2, 4)	0.0099962	879

threshold of 0.01 and window size of 6 is chosen. The error of BP with constant momentum factor after 5000 epoch is 0.1017534 .

Table 2. Simulation results for digit problem

Learning Automata	Final Mean Square Error	Epoch for Error Goal = 0.01
Tsetline (4, 4)	0.0099855	4560
TsetlineG (2, 4)	0.0074391	2886
Krinsky (2, 4)	0.0074391	2886
Krylov (2, 4)	0.0099975	4005

Table 3 shows the performance of the network when different automata are assigned to different layers to adapt the momentum factor. Each automata is responsible for adaptation of momentum factor for its assigned layer. For all automata in this simulation the threshold of 0.01 and window size of 1 is chosen. The error of BP with constant momentum factor after 5000 epoch is 0.1675668.

Table 4 shows the results of experiments in which one automata is used by each layer of the network to adjust the steepness parameter of that layer. For all automata in this simulation the threshold of 0.01 and window size of 6 is chosen. The error of standard BP after 5000 epoch is 0.1017534.

Remark 1: The simulations given in figures 9 through 16compare the proposed method with conjugate-gradient method, Baba's method, and adaptive steepness method.

Table 3. Simulation results for digit problem

Hidden Layer Automata	Output Layer Automata	Final Error	Epoch for Error = 0.01
Tsetline (4, 4)	Tsetline (4, 4)	0.0099984	3010
Tsetline (4, 4)	Krinsky (2, 4)	0.0089071	2407
Tsetline (4, 4)	Krylov(2, 4)	0.0099510	775
Tsetline (4, 4)	TsetlineG(2, 4)	0.0099897	622
Krinsky(2, 4)	Tsetline (4, 4)	0.0099577	112
Krinsky(2, 4)	Krinsky(2, 4)	0.0099736	649
Krinsky(2, 4)	Krylov(2, 4)	0.0099965	789
Krinsky(2, 4)	TsetlineG(2, 4)	0.0099998	822
Krylov(2, 4)	Tsetline (4, 4)	0.0099859	899
Krylov(2, 4)	Krinsky(2, 4)	0.0099850	417
Krylov(2, 4)	Krylov(2, 4)	0.0099891	880
Krylov(2, 4)	TsetlineG(2, 4)	0.0099806	614
TsetlineG(2, 4)	Tsetline (4, 4)	0.0099882	113
TsetlineG(2, 4)	Krinsky(2, 4)	0.0099924	801
TsetlineG(2, 4)	Krylov(2, 4)	0.0099941	635
TsetlineG(2, 4)	TsetlineG(2, 4)	0.0099970	895

Figures 9 through 12 compare the performance of conjugate-gradient method with the proposed scheme when momentum factor or steepness parameter are adapted for encoding and digit recognition problems. For these simulations the threshold of 0.01 and window size of 1 are used.

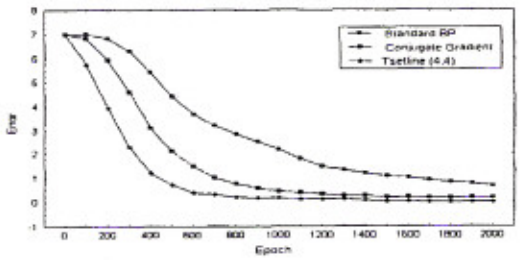

Figure 9. Adaptation of momentum factor for encoding problem

Figures 13 and 14 compares the performance of Babas method [1] with the proposed scheme for encoding and digit recognition problems, when the momentum factor is adapted. For this simulation the threshold of 0.01 and window size of 1 are used. The parameters of Babas method are the same as parameters used in their work [1]. As shown in figures 13 and 14 the proposed scheme exhibits higher performance than Babas method.

Careful inspection of Babas algorithm reveals the fact that this algorithm is a special case of our proposed algorithm when window size is 1 and threshold value is 0. The proposed algorithm is superior to Babas scheme because 1) The Babas algorithm does not use the advantages of the

Table 4. Simulation results for digit problem

Hidden Layer Automata	Output Layer Automata	Final Error	Epoch for Error = 0.01
Tsetline (4, 4)	Tsetline (4, 4)	0.0099448	2969
Tsetline (4, 4)	Krinsky (2, 4)	0.0089643	2477
Tsetline (4, 4)	Krylov(2, 4)	0.0014180	3656
Tsetline (4, 4)	TsetlineG(2, 4)	0.0098581	4949
Krinsky(2, 4)	Tsetline (4, 4)	0.0990697	5000
Krinsky(2, 4)	Krinsky(2, 4)	0.1082491	5000
Krinsky(2, 4)	Krylov(2, 4)	0.0006661	3439
Krinsky(2, 4)	TsetlineG(2, 4)	0.1110588	5000
Krylov(2, 4)	Tsetline (4, 4)	0.0087339	2317
Krylov(2, 4)	Krinsky(2, 4)	0.0097019	1726
Krylov(2, 4)	Krylov(2, 4)	0.0796143	5000
Krylov(2, 4)	TsetlineG(2, 4)	0.0029425	3983
TsetlineG(2, 4)	Tsetline (4, 4)	0.0098257	1384
TsetlineG(2, 4)	Krinsky(2, 4)	0.0097729	1355
TsetlineG(2, 4)	Krylov(2, 4)	0.0095585	4715
TsetlineG(2, 4)	TsetlineG(2, 4)	0.0098836	2520

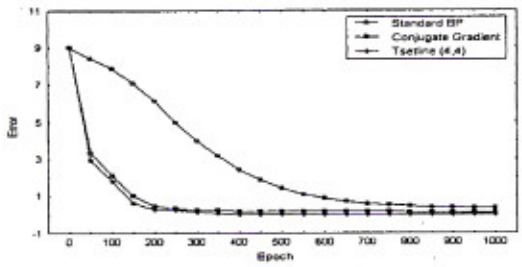

Figure 10. Adaptation of momentum factor for digit problem

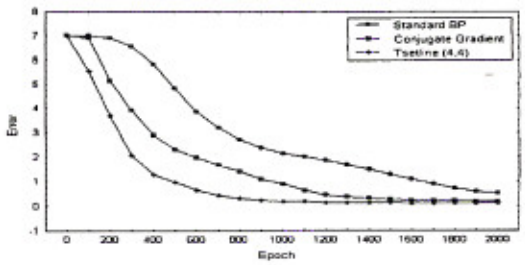

Figure 11. Adaptation of steepness parameter for encoding problem

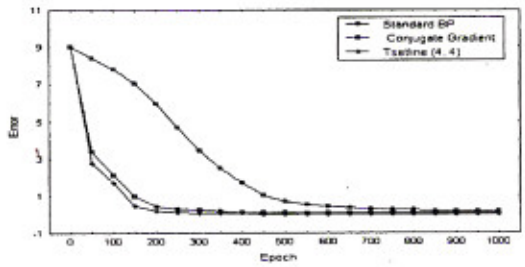

Figure 12. Adaptation of steepness parameter for digit problem

window size and threshold value. 2) Babas scheme uses hierarchical automata which makes the algorithm more complicated and time consuming. 3) Babas method is proposed for adaptation of momentum factor only.

Figures 15 and 16 compare the performance of Adaptive steepness (ASBP) [13]and proposed scheme for encoding and digit recognition problems, respectively. For these simulations the threshold of 0.01 and window size of 1 are chosen.

Simulation Results for simultaneous adaptation: In another experiment whose results are given in table 5, two automata of the same kind are used to adapt momentum factor and steepness parameter simultaneously. As it can be seen in this table, best result is for the case when two Krinsky automata are used for adaptation of both steepness parameter and momentum factor. For these experiments the threshold of 0.01 and window size of 3 are chosen. The error of BP after 5000 epoch is 0.0976520.

Table 6 shows the effects of association of different automata to different layers of the network for adjusting the momentum factor and steepness parameter simultaneously. A same pair of automata are used for adjusting both steepness parameter and momentum factor. It can be seen that the best pair of automata which gives the highest rate of convergence is Krylov for hidden layer and Tsetline for output layer. Note that this pair of automata is used for adaptation of momentum factor as well as steepness parameter. For all automata in this experiments the threshold of 0.01 and window size of 3 are chosen. The error of BP with constant momentum factor after 5000 epoch is 0.0976520.

Remark 4: In this remark, we examine the ability of the proposed algorithm to escape from local minima. For this propose, we chose a problem in which local minima are occurred frequently [4]. The training set of this problem is given in the table 7.

The network which is used has two input nodes x and y, two hidden units, and one output unit. In this problem, if hidden units produce the lines a and b the local minima has been occurred and if hidden units produce the lines c and d the global minima occurred [4]. Figure 17 shows these configurations.

In each simulation, we chose same initial points for

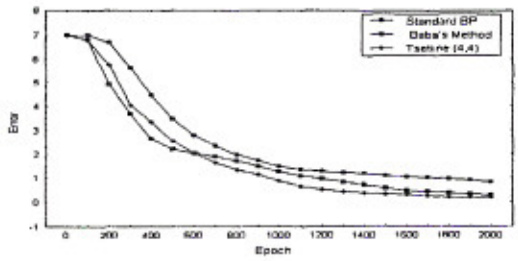

Figure 13. Adaptation of momentum factor for encoding problem

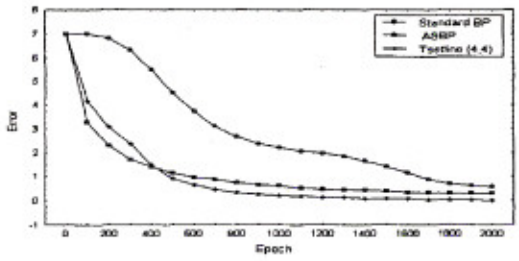

Figure 15. Adaptation of steepness parameter for encoding problem

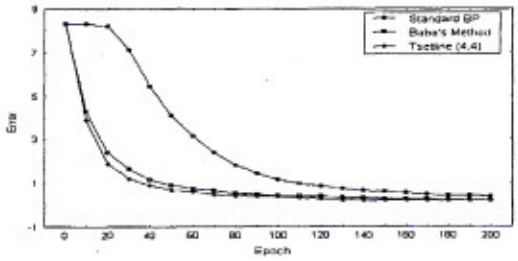

Figure 14. Adaptation of momentum factor for digit problem

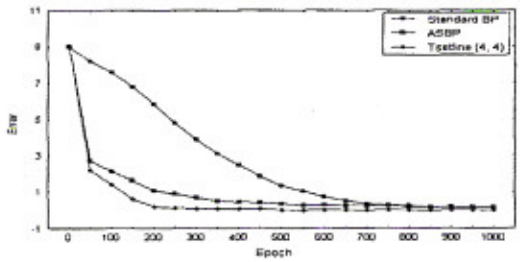

Figure 16. Adaptation of steepness parameter for digit problem

both BP (with constant momentum) and proposed algorithm when used to adapt the steepness parameter. Simulations have been carried out for 20 runs. Each run uses a random initial point near the local minima. In these simulations, the BP stuck at local minima for all 20 runs whereas the proposed algorithm escapes from local minima for 13 runs and stuck at local minima for 7 runs after 10000 epoch.

5 Conclusions

In this paper, we applied the fixed structure learning automata for adjusting the parameters of the BP algorithms based on the observation of the random response of neural network. We have demonstrated through simulations that the use of fixed structure learning automata for adaptation of momentum factor and steepness parameter of BP algorithm increase the rate of convergence by a large amount. By using fixed structure learning automata in BP algorithm it is possible to compute a new point that is closer to the optimum than the point computed by BP algorithm. In all problems we have studied so far, the convergence of BP which uses fixed structure learning automata or variable structure learning automata for adaptation of momentum factor or steepness parameter have been higher than the standard BP. Simulation results also indicate that speed of convergence can be improved if both momentum factor and steepness parameter are adapted simultaneously. It should be mentioned that for almost all the experiments we have conducted the FSLA approach has performed better than VSLA approach when used for adaptation of steepness parameter and momentum factor. The result of this paper can be generalized to be applied to the multi-layer neural networks.

References

[1] N. Baba and H. Handa. Utilization of hierarchical structure stochastic automata for the backpropagation method with momentum. In *Proc. of IEEE ICNN-95*, pages 389–393, 1995.

[2] H. Beigy, M. R. Meybodi, and M. B. Menhaj. Adaptation of learning rate in backpropagation algorithm using fixed structure learning automata. In *Proc. of ICEE-98*, volume III, pages 117–123, Tehran Iran, 1998.

[3] Y. Dali and L. Zemin. A LMS algorithm with stochastic momentum factor. In *Proc. of ISCAS-93*, pages 1250–1253, 1993.

Table 5. Simultaneous adaptation for digit problem

Learning Automata	Final Mean Square Error	Epoch for Error
Tsetline (4, 4)	0.0092784	807
TsetlineG (2, 4)	0.099826	982
Krinsky (2, 4)	0.0084366	49
Krylov (2, 4)	0.0098485	699

Table 6. Simultaneous adaptation for digit problem

Hidden Layer Automata	Output Layer Automata	Error	Epoch
Tsetline (4, 4)	Tsetline (4, 4)	0.0098906	1822
Tsetline (4, 4)	Krinsky (2, 4)	0.0099240	1124
Tsetline (4, 4)	Krylov(2, 4)	0.0094257	1614
Tsetline (4, 4)	TsetlineG(2, 4)	0.0099214	814
Krinsky(2, 4)	Tsetline (4, 4)	0.0036989	2076
Krinsky(2, 4)	Krinsky(2, 4)	0.0099466	566
Krinsky(2, 4)	Krylov(2, 4)	0.0070825	1187
Krinsky(2, 4)	TsetlineG(2, 4)	0.0095368	62
Krylov(2, 4)	Tsetline (4, 4)	0.0050398	31
Krylov(2, 4)	Krinsky(2, 4)	0.0098945	1303
Krylov(2, 4)	Krylov(2, 4)	0.0099984	386
Krylov(2, 4)	TsetlineG(2, 4)	0.0099983	724
TsetlineG(2, 4)	Tsetline (4, 4)	0.0081699	1758
TsetlineG(2, 4)	Krinsky(2, 4)	0.0099911	606
TsetlineG(2, 4)	Krylov(2, 4)	0.0098906	131
TsetlineG(2, 4)	TsetlineG(2, 4)	0.0099330	780

Table 7. Training set for given problem

Pattern	x	y	Desired output
A	0	0	0
B	1	0	1
C	1	1	0
D	0	1	1
E	0.5	0.5	0

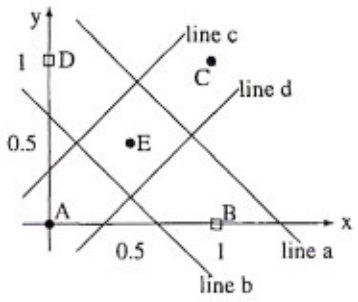

Figure 17. Lines produced by hidden units

[4] P. Frasconi, M. Gori, and A. Tesi. Success and failures of backpropagation: A theoretical investigation. Technical report, Dipartimento di Sistemi e Information, Universita di Firenze, Firenze, Italy, 1992.

[5] L. Guan, S. Cheng, , and R. Zhou. Artificial neural network power system stabilizer trained with an improved bp algorithm. *IEE Proc. of Power Generation, Transmission and Distribution*, 143(2):135–141, 1996.

[6] Y. Jia and Y. Dali. Analysis of the misadjustment of BP network and an improved algorithm. In *Proc. of ISCAS-93*, pages 2592–2595, 1993.

[7] D. McLean, Z. Bandar, and J. OShea. Improved interpolation and extrapolation from continuos training examples using a new neuronal model with adaptive steepness. In *Proc. of ANZIIS- 94*, pages 125–129, 1994.

[8] M. B. Menhaj and M. R. Meybodi. Flexible sigmodal type functions for neural nets using game of automata. In *Proc. of CSICC-96*, pages 221–232, Tehran Iran, 1996.

[9] M. B. Menhaj and M. R. Meybodi. Using learning automata in backpropagation algorithm with momentum. Technical report, Computer Eng. Dept., Amirkabir university of Technology, Tehran, Iran, 1997.

[10] M. R. Meybodi and S. Lakshmivarhan. *On a Class of Learning Algorithms which have a Symmetric Behavior Under Success and Failure*, pages 1456–155. Springer Verlag Lecture Notes in Statistics. Springer Verlag, Berlin, 1984.

[11] K. S. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction*. Prentice-hall, Englewood cliffs, 1989.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, volume 1, chapter 8, pages 318–362. Cambridge, MA: MIT Press, 1986.

[13] A. Sperduti and A. Starita. Speed up learning and network optimization with extended backpropagation. *Neural Networks*, 6:365–383, 1993.

[14] X. H. Yu and G. A. Chen. Efficient estimation of dynamically optimal learning rate and momentum for backpropagation learning. In *Proc. of IEEE ICNN-95*, pages 385–388, 1995.