ELSEVIER

Neurocomputing 48 (2002) 957–974

NEUROCOMPUTING

www.elsevier.com/locate/neucom

A note on learning automata-based schemes for adaptation of BP parameters

M.R. Meybodi*, H. Beigy

Soft Computing Laboratory, Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran

Received 30 October 2000; received in revised form 25 August 2001; accepted 5 September 2001

Abstract

In this paper, we study the ability of learning automata-based schemes in escaping from local minima when standard backpropagation (BP) fails to find the global minima. It is demonstrated through simulation that learning automata-based schemes compared to other schemes such as SAB, Super SAB, Fuzzy BP, adaptive steepness method, and variable learning rate method have a higher ability to escape from local minima.

Keywords: Neural network; Backpropagation; Learning automata; Learning rate; Steepness parameter; Momentum factor

1. Introduction

The multilayer feedforward neural network models with error backpropagation (BP) algorithm have been widely researched and applied. Despite the many successful applications of BP, it has many drawbacks. For complex problems it may require a long time to train the networks, and it may not converge at all. Long training time can be the result of the non-optimum values for the parameters of the training algorithm. It is not easy to choose appropriate values for these parameters for a particular problem. The parameters are usually determined by trial and error and using the past experiences. For example, if the learning rate is too small, convergence can be very slow: if too large, paralysis and continuous instability can result. Moreover, the best value at the beginning of training may not be so good later. Thus, research has suggested algorithms for

* Corresponding author.
E-mail addresses: meybodi@ce.aku.ac.ir (M.R. Meybodi), beigy@ce.aku.ac.ir (H. Beigy).

0925-2312/02/$ - see front matter
PII: S0925-2312(01)00686-5

automatically adjusting the parameters of the training algorithm as training proceeds, such as algorithms proposed by Arabshahi et al. [1], Kandil et al. [13], Parlos et al. [24], Cater [6], Franzini [9], Vosl et al. [30], Tesnuro and Janssens [28], Devos and Orban [8], Darken et al. [7], Tollenaere [29], Chan and Fallside [6a], Jacobs [12], Sperduti and Starita [27] and Riedmiller and Heinrich [25] to mention a few. Several learning automata-(LA) based procedures have been developed, recently [16–19,2,4,23]. In these methods, variable structure learning automata (VSLA) or fixed structure learning automata (FSLA) have been used to find the appropriate values of parameters for the BP training algorithm. In these schemes either a separate learning automaton is associated with each layer or each neuron of the network or a single automaton is associated with the whole network to adapt the appropriate parameters. It is shown that, the learning rate adapted in such a way increases the rate of convergence of the network by a large amount [23,3].

Many researchers have looked at the powerful interpolation capabilities of BP as a means for learning sophisticated concepts. When looking at it in this direction one soon becomes interested in finding out what BP can, and cannot learn. For example, learning complex Boolean functions is far more complicated than solving ordinary pattern recognition problems. These functions typically involve inputs with few coordinates, but very complex mappings. For this kind of problem BP may fail in discovering local minima, particularly with minimal architectures. As pointed out by numerous researchers, BP can be trapped in local minima during gradient descent, and in many of these cases it seems very unlikely that any learning algorithm could perform satisfactorily in terms of computational requirements.

When we use learning automata as the adaptation technique for BP parameters, the search for optimum is carried out in probability space rather than in parameter space as is the case with other adaptation algorithms. In the standard gradient method, the new operation point lies within a neighborhood distance of the previous point. This is not the case for adaptation algorithm based on stochastic principles such as LA, as the new operating point is determined by probability function and is therefore not considered to be near the previous operating point. This gives the algorithm a higher ability to locate the global minima. In this article, we study the ability of LA-based schemes in escaping from local minima when standard BP fails to find the global minima. In this paper, it is demonstrated through simulation that LA-based schemes compared to other schemes such as SAB [12], SuperSAB [12], adaptive steepness method (ASBP) [27], variable learning rate (VLR) method [15] and Fuzzy BP [1] have a higher ability to escape from local minima, that is, BP parameter adaptation using the LA-based schemes increases the likelihood of bypassing the local minimum.

The rest of the paper is organized as follows. Section 2 briefly presents the basic BP algorithm and LA. Existing LA-based adaptation schemes for BP parameters are described in Section 3. Section 4 demonstrates through simulations the ability of LA-based schemes in escaping from local minima. The last section is the conclusion.

2. Backpropagation algorithm and learning automata

In this section, in all brevity, we discuss the fundamentals of BP learning algorithm and LA.

Backpropagation algorithm: Error BP training algorithm which is an iterative gradient descent algorithm is a simple way to train multilayer feedforward neural networks [26]. The BP algorithm is based on the following gradient descent rule:

$$W(n+1) = W(n) + \mu G(n) + \alpha[W(n) - W(n-1)], \tag{1}$$

where W is the weight vector, n is the iteration number, μ is learning rate, α is momentum factor, and G is gradient of error function that is given by

$$G(n) = -\nabla E_p(n), \tag{2}$$

where E_p is the sum of squared error given by

$$E_p(n) = \frac{1}{2} \sum_{j=1}^{\#\text{outputs}} [T_{p,j} - O_{p,j}]^2 \quad \text{for } p = 1, 2, \ldots, \#\text{patterns}, \tag{3}$$

where $T_{p,j}$ and $O_{p,j}$ are the desired and actual outputs for pattern p at output node j. One of the major problems encountered during implementation of the BP learning rule is proper choice and update of the learning rate μ to allow convergence, while maintaining the number of iterations required for the training algorithm at a reasonable number. One of the main reasons for investigating the possibility of the adaptive learning rate rule is the desire to reduce the sensitivity of the learning to the learning rate, without adding more tuning parameters.

In the BP algorithm framework, each computational unit computes the same activation function. The computation of the sensitivity for each neuron requires the derivative of activation function, therefore, this function must be continuous and differentiable. The activation function is normally a sigmoid function chosen between $1/1 + \exp(-\lambda \,\text{net})$ and tanh $(\lambda \,\text{net})$. The coefficient of the exponent of the exponential term determines the steepness of linearity of that function. The steepness parameter λ is often set to a constant value and is not changed by the learning algorithm. We gain much flexibility, if we move the net inputs of the sigmoidal function near their active regions, where the associated gradient is not very close to zero. This prevents the BP algorithm to be trapped at some points in the network parameters space, where the BP algorithm would effectively stop, even though it is not close to a local minima point. This will cause the gradient of the error function to be small if the sigmoidal is shifted far outside the active region of the input to the function. Therefore, it is better to center each sigmoid to be inside the active region of the sigmoidal function.

The momentum term in weight adaptation Eq. (1) causes a large change in the weight if the changes are currently large, and decreases as the changes become less. This means that the network is less likely to get stuck in local minima early on, since the momentum term will push the changes towards the local downward trend. Momentum is of great assistance in speeding up convergence along shallow gradients, allowing the path the network takes towards the solution to pick up speed in the downhill direction. The error surface may consist of long gradually sloping ravines,

which finish at minima. Convergence along these ravines is slow, and usually, the algorithm oscillates across the ravine valley as it moves towards a solution. This is difficult to speed up without increasing the chance of overshooting the minima, but the addition of the momentum term is fairly successful. This difficulty could be removed if we select the momentum factor to be small near minima and to be large far from minima.

Learning automata: LA operating in unknown random environments have been used as models of learning systems. These automata choose an action at each instant from a finite action set, observe the reaction of the environment to the chosen action and modify the selection of the next action on the basis of this reaction.

A learning automaton is a quintuple $\langle \underline{\alpha}, \underline{\Phi}, \underline{\beta}, F, G \rangle$, where $\underline{\alpha} = (\alpha_1, \ldots, \alpha_R)$ is the set of actions that it must choose from, $\underline{\Phi} = (\Phi_1, \ldots, \Phi_s)$ is the set of states, $\underline{\beta} = \{0, 1\}$ is the set of inputs, where "1" represents a penalty and "0" a reward, $\underline{G} : \underline{\Phi} \to \underline{\alpha}$ is the output map and determines the action taken by the automaton if it is in state Φ_j, and $F : \underline{\Phi} \times \underline{\beta} \to \underline{\Phi}$ is the transition map and defines the transition of the state of the automaton on receiving an input, F 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 $\underline{\beta} = \{0, 1\}$ and is the feedback response of the environment to the automaton. The environment penalizes (i.e., $\beta(n) = 1$) the automaton with the penalty c_i, which is action-dependent. On the basis of the response $\beta(n)$, the state of the automaton $\Phi(n)$ is updated and a new action is chosen at the time $(n+1)$. Note that $\{c_i\}$ are unknown initially and it is desired that as a result of interaction with the environment, the automaton arrives at the action which presents it with the minimum penalty response in an expected sense. If the probability of the transition from one state to another state and probabilities of correspondence of action and state are fixed, the automaton is said to be a fixed-structure automaton FSLA, and otherwise the automaton is said to be a variable-structure automaton VSLA. Examples of the FSLA type that we use in this paper are Tsetline, Krinsky, TsetlineG, and Krylov automata [23].

We summarize some of the FSLA and variable structure automata in the following paragraphs.

The two-state automata ($L_{2,2}$): This automaton has two states, ϕ_1 and ϕ_2 and two actions α_1 and α_2. The automaton accepts input from a set of $\{0, 1\}$ and switches its states upon encountering an input 1 (unfavorable response) and remains in the same state on receiving an input 0 (favorable response). An automata that uses this strategy is refered to as $L_{2,2}$, where the first subscript refers to the number of states and second subscript to the number of actions.

The two-action automata with memory ($L_{2N,2}$): This automaton has $2N$ states and two actions and attempts to incorporate the past behavior of the system in its decision rule for choosing the sequence of actions. While the automata $L_{2,2}$ switches from one action to another on receiving a failure response from environment, $L_{2N,2}$ keeps an account of the number of success and failures received for each action. It is only when the number of failures exceeds the number of successes, or some maximum value N, the automata switches from one action to another. The procedure described above is one convenient method of keeping track of performance of the actions α_1 and α_2. As

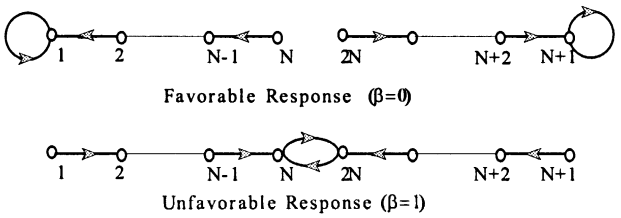

Fig. 1. The state transition graph for $L_{2N,2}$.

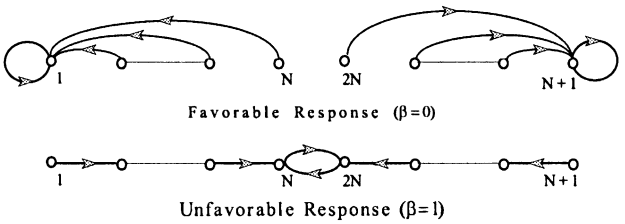

Fig. 2. The state transition graph for Krinsky automata.

such, N is called memory depth associated with each action, and automata is said to have a total memory of $2N$. For every favorable response, the state of automata moves deeper into the memory of corresponding action, and for an unfavorable response, moves out of it. The state transition graph of $L_{2N,2}$ automata is shown in Fig. 1.

The Krinsky automata: This automaton behaves exactly like $L_{2N,2}$ automaton when the response of the environment is unfavorable, but for favorable response, any state ϕ_i (for $i=1,\ldots,N$) passes to the state ϕ_1 and any state ϕ_i (for $i=N+1,\ldots,2N$) passes to the state ϕ_{N+1}. This implies that a string of N consecutive unfavorable responses is needed to change from one action to another. The state transition graph of Krinsky automata is shown in Fig. 2.

The Krylov automata: This automaton has state transitions that are identical to the $L_{2N,2}$ automaton when the output of the environment is favorable. However, when the response of the environment is unfavorable, a state ϕ_i ($i \neq 1, N, N+1, 2N$) passes to a state ϕ_{i+1} with probability 0.5 and to a state ϕ_{i-1} with probability 0.5. When $i=1$ or $i=N+1$, ϕ_i stays in the same state with probability 0.5 and moves to ϕ_{i+1} with the same probability. When $i=N$, automaton state moves to ϕ_{N-1} and ϕ_N with the same probability 0.5. When $i=2N$, automaton state moves from ϕ_{2N-1} and ϕ_N with the same probability 0.5. The state transition graph of Krylov automata is shown in Fig. 3.

The J automata: The ***J*** automata which we denote by $J(K,N)$ has KN states and K actions and attempts to incorporate the past behavior of the system in its decision rule for choosing the sequence of actions. States with numbers $(k-1)N+1$ through kN correspond to action k. The state transition graph of this automata for favorable response and unfavorable response is shown in Fig. 4.

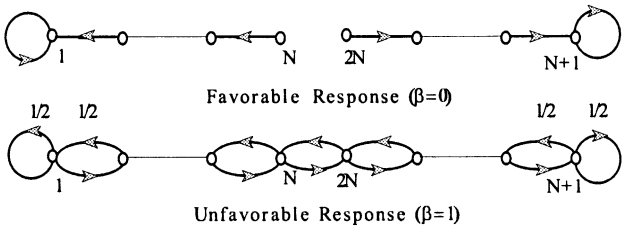

Fig. 3. The state transition graph for Krylov automata.

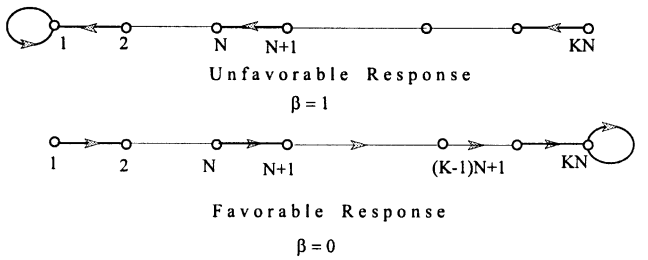

Fig. 4. The state transition graph for $J_{KN,K}$.

All these automata can be extended to multiple action automata. In this paper, we refer to an automaton by the name of automaton followed by the list of parameters for that automaton, the first parameter refers to the number of actions and the second parameter refers to the memory depth for each action. For the sake of simplicity in the presentation, we denote FSLA Automata with K actions and memory depth of N by Automata(K, N).

Variable structure learning automata: Variable structure automata are represented by the sextuple $\langle \underline{\beta}, \underline{\Phi}, \underline{\alpha}, \underline{p}, G, T \rangle$, where $\underline{\beta}$ is a set of input actions, $\underline{\Phi}$ is a set of internal states, $\underline{\alpha}$ is a set of outputs, $\underline{p}$ denotes 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 the learning algorithm for updating the action probabilities. Various learning algorithms have been reported in the literature [23]. Let α_i be the action chosen at time k as a sample realization from distribution $p(k)$. The linear reward-inaction algorithm (L_{R-1}) is one of the earliest schemes. In an L_{R-1} scheme, the recurrence equation for updating p is defined as

$$p_j(k) = \begin{cases} p_j(k) + a(1 - p_j(k)) & \text{if } i = j, \\ p_j(k)(1 - a) & \text{if } i \neq j \end{cases} \tag{4}$$

when β is zero and $\underline{p}$ is unchanged when β is one. The parameter 'a' is called step length. It determines the amount of increase (decrease) of the action probabilities. For the linear reward ε-penalty algorithm ($L_{R-\varepsilon P}$) scheme, the recurrence equation for updating $\underline{p}$ is defined as

$$p_j(k) = \begin{cases} p_j(k) + a(1 - p_j(k)) & \text{if } i = j, \\ p_j(k)(1 - a) & \text{if } i \neq j \end{cases} \tag{5}$$

when $\beta(k) = 0$ and

$$p_j(k) = \begin{cases} p_j(k)(1 - b) & \text{if } i = j \\ \dfrac{b}{r - 1} + (1 - b)p_j(k) & \text{if } i \neq j \end{cases} \tag{6}$$

when $\beta(k) = 1$. The parameters 'a' and 'b' represent reward and penalty parameters, respectively. The parameter 'a'('b') determines the amount of increase (decrease) of the action probabilities. For the sake of simplicity in presentation, we denote VSLA Automata with K actions by Automata(K). For more information on learning automata refer to [21–23,20,14].

3. LA-based schemes for adaptation of BP parameters

In this section, we first briefly describe LA-based schemes [16–18,2,4] for adaptation of BP parameters. In all the existing schemes, one or more automaton have been associated with the network. The LA based on the observation of the random response of the neural network, adapt one or more BP parameters. The interconnection of learning automata and neural network is shown in Fig. 5. Note that, the neural network is the environment for the learning automata. The learning automaton adjusts the parameters of the BP algorithm according to the amount of the error received from neural network. The actions of the automata correspond to the values of the parameter being calculated and input to the automata is some function of the error in the output of neural network.

Existing LA-based procedures for adaptation of BP parameters can be classified into four classes which we call class A, B, C, and D. In the next few paragraphs we briefly describe these classes.

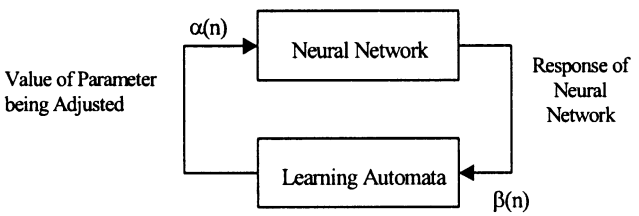

Fig. 5. The interconnection of learning automata and neural network.

Class A schemes: In class A schemes, an automaton is associated with the network to adapt the corresponding BP parameter. The learning automaton assigned to the network chooses an action and the value of this action is used for training the network for some number of epochs. Then a function of error between the desired and the actual outputs of network is considered as the response from the environment. A window on the past values of the errors are swiped and the average value of the error in this window is computed. If the difference of the average errors in the last two epochs is less than the predefined threshold value, then the response of the environment is favorable and if this difference is greater than the threshold value, then the response of the environment is unfavorable.

Class B schemes: In class B schemes, an automaton is associated with each layer of the network for adaptation of the corresponding BP parameter for that layer. The learning automaton chooses an action and the value of the chosen action is used for training of the network in that layer for some number of epochs. In this scheme, the response of the environment to the output layer automaton is computed in the same manner as in class A schemes. The responses to the hidden layers automata are computed as in class A except that the estimated error in each hidden layer is used instead of the actual error.

Class C schemes: In class C schemes, one automaton is associated with each link of the network in order to adjust the parameter of that link. In this class, the automaton receives favorable response from the environment if the algebraic sign of the derivative of error with respect to the weight in the two consecutive iterations are the same, and receives unfavorable response if the algebraic sign of the derivative of error with respect to the weight in the two consecutive iterations alternate.

Class D schemes: In class D schemes, one automaton is associated with each neuron of the network to adjust the parameter of that neuron. In this class, the automaton receives favorable response from the environment if the algebraic sign of the derivative of error with respect to the parameter being adjusted in the two consecutive iterations are the same and receives unfavorable response if the algebraic sign of the derivative of error with respect to the parameter in the two consecutive iterations alternates.

For the sake of convenience in the presentation, we use the following naming conventions to refer to different LA-based schemes in classes A, B, C, and D. Without loss of generality, we assume that in class A and class B, the neural network has one hidden layer.

Automata-AX(γ): A scheme in class A for adjusting parameter γ which uses X structure learning automata.

Automata.Automata$_1$-*Automata*$_2$-*BX*(γ): A scheme in class B which uses X structure learning automata for hidden layer and X structure learning automata Automata$_2$ for output layer.

Automata-CX(γ): A scheme in class C for adjusting parameter γ which uses X structure learning automata Automata.

Automata-DX(γ): A scheme in class D for adjusting parameter γ which uses X structure learning automata Automata.

The rate of convergence can be improved if both the learning rate and steepness parameter are adapted simultaneously. Simultaneous, use of class C and class D schemes

for adaptation of learning rate and steepness parameters is also reported in [23]. In [23] a class C scheme is used for adaptation of learning rate and a class D scheme is used for adaptation of steepness parameter. An LA-based scheme that simultaneously adapts learning rate and steepness parameter is denoted by Automata1-Automata2-CDF(μ, λ), if FSLA is used and Automata1-Automata2-CDV(μ, λ), if VSLA is used.

A simple method of increasing the learning rate and stability of training algorithm is to modify the standard BP by including the momentum factor [26] as given in Eq. (1). An LA-based scheme which simultaneously adapts the learning rate and momentum is denoted by Automata1-Automata2-CF(η, α).

The letters F and V in the above names denote FSLA and VSLA, respectively. X denotes either fixed or variable. For all the LA-based schemes reported in the literature, it is shown below through simulation that the use of LA for adaptation of BP-learning algorithm parameters increases the rate of convergence by a large amount.

In the remaining part of this section we first discuss non-LA methods for the purpose of comparison with LA-based methods described above and then show some simulation results.

Variable learning rate (*VLR*): VLR is a scheme in which the learning rate varies according to the performance of the algorithm [15]. If the error decreases after a weight update, then the learning rate is increased by some factor (e.g., 1.05). If the error increases more than some percentage (typically one to five percent), then the weight update is discarded and the learning rate is decreased by some factor (e.g., 0.7) and the momentum term (if it is used) is set to zero. When a successful step is taken the momentum term is reset to its original value. If the algorithm is working well, and the error continues to go down, then learning rate will increase and convergence will speed up.

Fuzzy control of BP (*FuzzyBP*): The central idea behind FuzzyBP is the implementation of heuristics for determining the values of learning rate and momentum factor in terms of fuzzy rules. This is done by considering the error, $E(n)$, and the change in the error, $\Delta E(n)$, at instant n, where $\Delta E(n) = E(n) - E(n-1)$. The values of $E(n)$ and $\Delta E(n)$ are categorized in the fuzzy linguistic sets such as *low*, *medium*, and *high*. The change in the learning rate, $\Delta\mu(n) = \mu(n) - \mu(n-1)$, also takes the fuzzy values of *negative small*, *zero*, and *positive small*. All these values are expressed in terms of membership functions. Based on the crisp values of $E(n)$ and $\Delta E(n)$ value of $\Delta\mu(n)$ is determined from fuzzy rules.

Adaptive steepness (*ASBP*): ASBP [27] method is a method, which uses gradient descent rule for adaptation of steepness parameter. In this method each neuron k has steepness parameter λ_k, which is changed by the following rule:

$$\Delta\lambda_k = -\varepsilon \frac{\partial E}{\partial \lambda_k}.$$

Self-adaptive BP (*SAB*): SAB was developed independently by Jacobs [12] and Devos and Orban [8]. SAB is a local method in which every weight has its own learning rate. In this method, every learning rate on every dimension is adapted based on the error surface independently. The learning rate increases if in two consecutive iterations gradient has the same sign and the learning rate should be a small constant

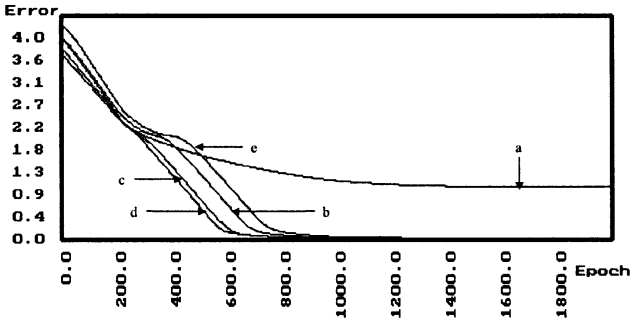

Fig. 6. (a) Standard BP; (b) Tsetline-AF(η); (c) Krinsky-AF(η); (d) Krylov-AF(η); and (e) $L_{R-\varepsilon p}$-AV(η).

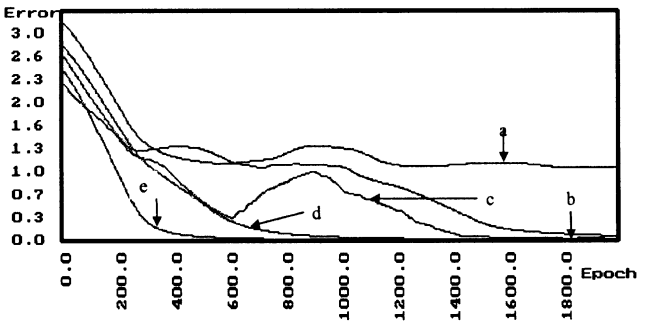

Fig. 7. (a) Standard BP; (b) Tsetline-AF; (c) Tsetline–Krinsky-BF; (d) Tsetline–Tsetline-BF; and (e) Tsetline–TsetlineG-BF.

value if the sign of gradient in two consecutive iterations alternates. SAB performs better than the BP, because it can adjust the learning rate over a wide range, but it has two drawbacks: (1) The selection of initial value η is hard to determine; (2) If the sign of gradient alternates, the learning rate is reset to initial value. SuperSAB algorithm which is proposed by Tollenaere [29] overcomes these problems.

In order to compare the effectiveness of different LA-based, schemes in different classes for the adaptation of BP parameters, we use Figs. 6–9 from Refs. [23,4]. Shown in these figures are typical error curves for different LA-based methods. As is shown, LA-based schemes result in dramatically faster convergence, and have significantly smaller tail than standard BP and some other non-LA-based schemes. In Fig. 6, the effectiveness of different schemes from class A is compared [4]. Fig. 7 compares different schemes in class B and one scheme from class A. For this simulation, Tsetline learning automata are associated with the hidden layer and the effect of association of different learning automata with the output layer is shown for digit problem. Fig. 8 compares the performance of different schemes from class A with J-CF scheme from class C for the digit problem. Fig. 9 compares the performance of ASBP method with J-DF(λ) scheme from class D and standard BP for parity problem. To the

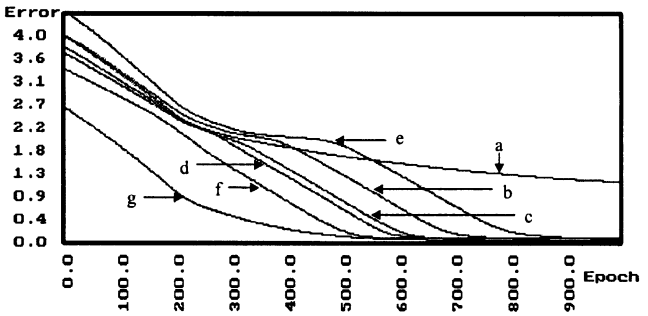

Fig. 8. (a) Standard BP; (b) Tsetline-AF(η); (c) Krinsky-AF(η); (d) Krylov-AF(η); (e) $L_{R-\varepsilon P}$-AV(η); (f) VLR; (g) J-CF(η).

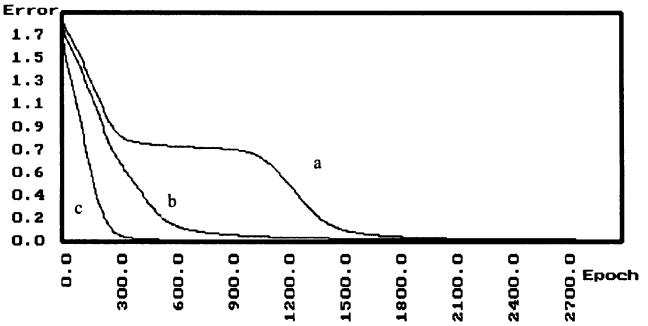

Fig. 9. (a) Standard BP; (b) ASBP; (c) J-DF(λ).

author's knowledge, ASBP method is the only method for the adaptation of steepness parameter.

4. LA-based schemes and local minima

In this section, we examine the ability of the learning automata-based schemes to escape from local minima. For this propose, we choose two problems in which local minima occur frequently [11]. These examples consider the sigmoidal network for the XOR Boolean function with the quadratic cost function and the standard learning environment.

To show the superiority of LA-based adaptation algorithm in terms of escaping from local minima, we test 12 different LA-based algorithms, five from class A, four from class B, 1 from class C, 1 from class D, and 1 from class CD, and compare their results with the standard BP and five other known adaptation methods: SAB [12], SuperSAB [29], VLR method [15], ASBP method [27], and fuzzy BP [1]. In all simulations, each learning automaton has K actions equally spaced in the interval $(0, 1]$. Window size, threshold value, and the maximum number of epochs are chosen to be 10, 0.01 and

Table 1

Pattern	X_0	X_1	Desired output
A	0	0	0
B	1	0	1
C	1	1	0
D	0	1	1
E	0.5	0.5	0

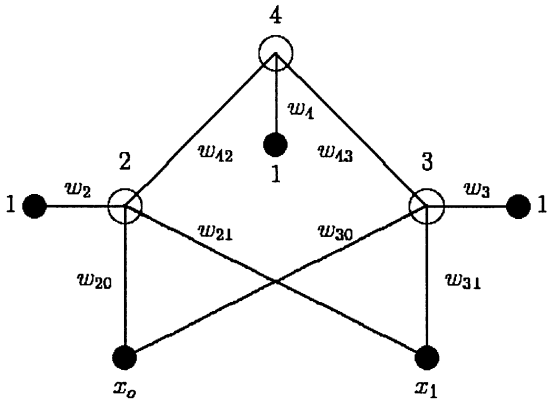

Fig. 10. Network architecture for test problems.

32 000, respectively. For $L_{R-\varepsilon P}$ learning automata the reward and penalty parameters are chosen to be 0.001 and 0.0001, respectively.

Example 1. The training set of this example is given in Table 1.

The network has two input nodes x_0 and x_1, two hidden units, and one output unit as shown in Fig. 10. In this problem, if hidden units produce the lines $2l$ and $3l$, then the local minima has occurred and if hidden units produce the lines $2g$ and $3g$, then the global minima occurred [10]. Fig. 11 shows these configurations. The error surface and contour plot of the network as a function of weights w_{20} aid w_{42} are given in Figs. 12 and 13.

Depending on the initial weights, the gradient can get stuck in points where the error is far from being zero. The presence of these local minima is intuitively related to the symmetry of the learning environment. Experimental evidence of the presence of local minima is given in Fig. 12. The local minimum is related to the weights $w_{20} = w_{21}$ and $w_{30} = w_{31}$ [11]. Table 2 shows the result of simulations for 100 runs for two different cases. For case 1, the initial weights are chosen in such a way that all the algorithms start from local minima. For case 2, the initial weights are chosen in such a way that all the algorithms start from a point near local minima ($w_{20}=w_{21}=w_{30}=0.025$). From Table 2 we note that for standard BP and also for standard BP when VLR, FuzzyBP, or ASBP are used to adapt the learning rate none of the 100 runs converge to the global minima.

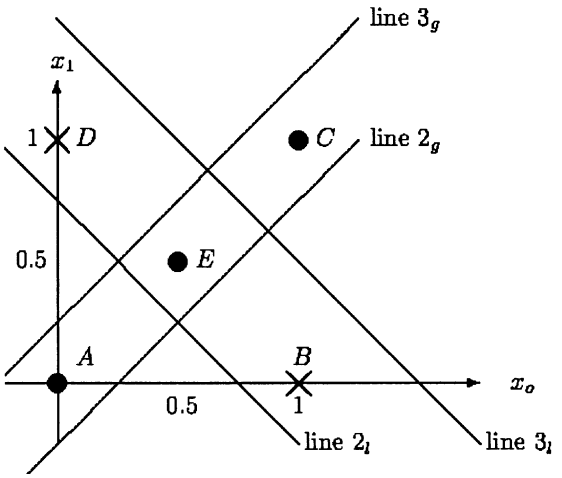

Fig. 11. Lines produced by hidden units of neural network.

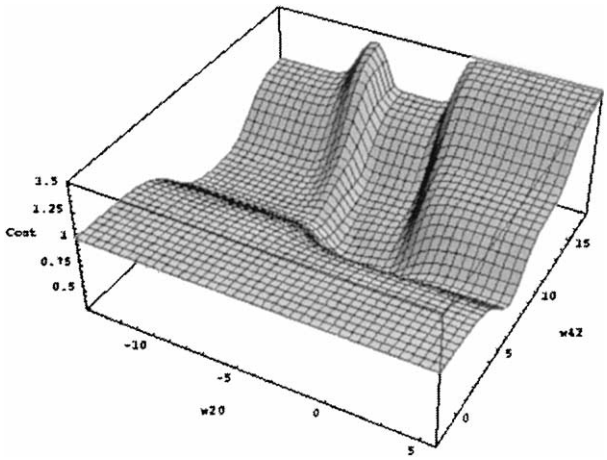

Fig. 12. Error surface as a function of weights w_{20} and w_{42}.

Among the non-LA-based methods, the SAB and SuperSAB methods perform the best. For the SuperSAB scheme, 10 out of 100 runs converge to global minima which is comparable to some of the LA-based schemes we have tested. The best result is obtained for algorithm J(4,4)-J(4,4)-CDF(μ,λ) for which 12 out of 100 runs converge to global minimum. The next best result belongs to J(4,4)-J(4,4)-CF(μ) scheme.

Example 2. This example considers the sigmoidal network for the XOR Boolean function (see Fig. 9) and standard training patterns (the first four rows of Table 1). Following [5], it can be shown that there is a manifold local minima given by

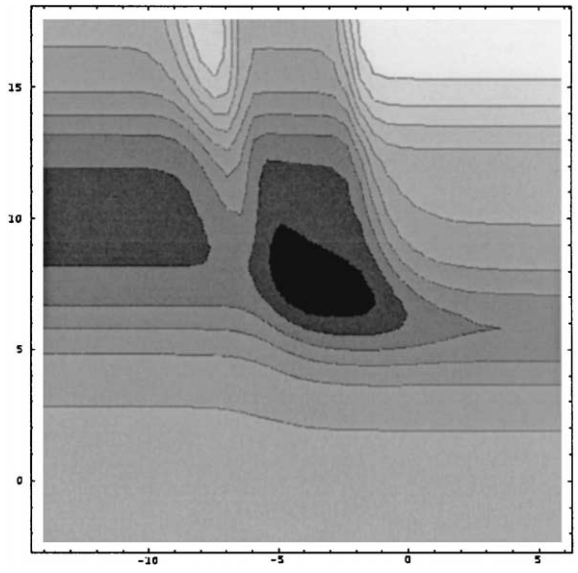

Fig. 13. Contour map of Fig. 11.

Table 2

Algorithm	Class	%Converged	
		Case 1	Case 2
BP		0	0
SAB		2	9
SuperSAB		5	10
VLR		0	0
FuzzyBP		0	0
ASBP		0	0
Tsetline(4, 4)-AF(μ)	A	2	9
Krinsky(4, 4)-AF(μ)	A	7	11
Krylov-(4, 4)-AF(μ)	A	3	9
$L_{R-\varepsilon P}$(10)-AF(μ)	A	8	11
Tsetline(4, 4)-AF(λ)	A	0	0
Tsetline(4, 4)-TsetlineG(2, 4)-BF(μ)	B	6	10
Tsetline(4, 4)-Krylov(4, 4)-BF(μ)	B	8	12
Tsetline(4, 4)-Krinsky(4, 4)-BF(μ)	B	6	8
Tsetline(4, 4)-Tsetline(4, 4)-BF(μ)	B	4	19
J(4, 4)-DF(λ)	D	6	18
J(4, 4)-CF(μ)	C	9	18
J(4, 4)-J(4, 4)-CDF(μ, λ)	CD	12	22

$\{w_{20} = w_{21} = w_2 = w_{30} = w_{31} = w_3 = 0, w_{42} = w_{43}, w_{42} + w_4 = 0\}$ with error of 0.5. An example of local minima and its contour are given in Figs. 14 and 15. Table 3 shows the result of simulations for 100 runs for two different cases. For Case 1, the initial weights are chosen in such a way that all the algorithms start from local minima. For

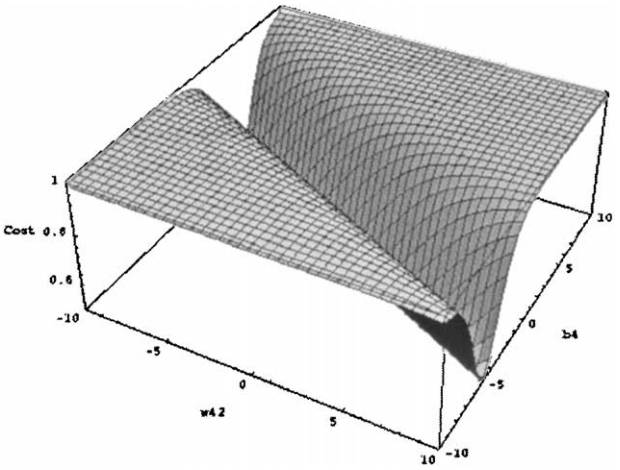

Fig. 14. Error surface as a function of weights w_{42} and w_4.

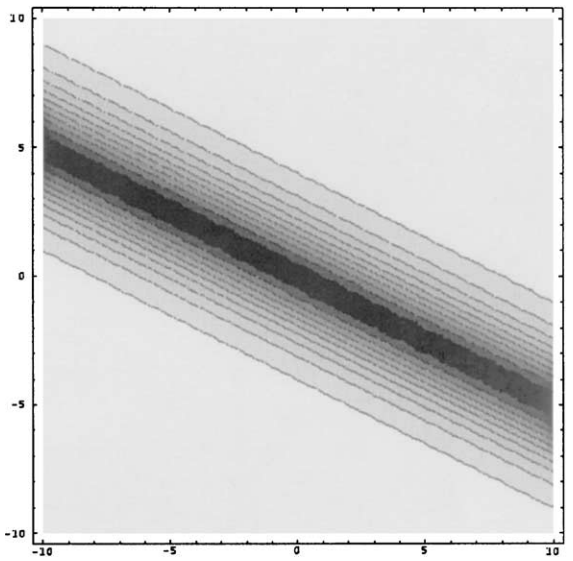

Fig. 15. Contour map of Fig. 14.

case 2, initial weights are chosen in such a way that all the algorithms start from a point near local minima ($w_{20} = w_{21} + 0.01$).

For this example, the results obtained for LA-based scheme are much better than the results obtained for example 1. Among the non-LA-based methods again SAB and SuperSAB perform the best and among the LA-based schemes J(4, 4)-J(4, 4)-CF(μ) produced the best result. Again note the superiority of the LA-based schemes over the

Table 3

Algorithm	Class	%Converged	
		Case 1	Case 2
BP		0	0
SAB		0	43
SuperSAB		0	45
VLR		0	0
FuzzyBP		0	0
ASBP		0	0
Tsetline$(4,4)$-AF(μ)	A	0	13
Krinsky$(4,4)$-AF(μ)	A	0	38
Krylov$(4,4)$-AF(μ)	A	0	36
$L_{R-\varepsilon P}(10)$-AF(μ)	A	0	46
Tsetline$(4,4)$-AF(λ)	A	0	0
Tsetline$(4,4)$–TsetlineG(2,4)-BG(μ)	B	0	29
Tsetline$(4,4)$–Krylov(4,4)-BF(μ)	B	0	51
Tsetline$(4,4)$–Krinsky(4,4)-BF(μ)	B	0	54
Tsetline$(4,4)$–Tsetline(4,4)-BF(μ)	B	0	64
J$(4,4)$-DF(λ)	D	0	56
J$(4,4)$-CF(μ)	C	54	76
J$(4,4)$-J$(4,4)$-CDF(μ,λ)	CD	38	74

non-LA schemes. For this example, for Case 1 none of the non-LA-based methods were able to converge to the global minima.

Remark 1. The reason for a higher performance of both J-J-CDF(μ,λ) and J-CF(μ) schemes compared to the other LA-based schemes is that they have a closer relationship with the Jacobs heuristics [12]. These heuristics, which are given below, are suggested as guidelines for accelerating the convergence of BP-learning algorithm through learning rate adaptation.

- Every adjustable network parameter of cost function should have its own individual learning-rate parameter.
- Every learning-rate parameter should be allowed to vary from one iteration to the other.
- When the derivative of cost function with respect to the synaptic weight has the same algebraic sign for several consecutive iterations of the algorithm, the learning-rate parameter for that particular weight should be increased.
- When the algebraic sign of the derivative of cost function with respect to the synaptic weight alternates for several consecutive iterations of the algorithm, the learning-rate parameter for that particular weight should be decreased.

Careful inspection of J-CF(μ) and J-J-CDF(μ,λ) shows that the above heuristics are somehow incorporated into these schemes.

Remark 2. The reason for such a good performance of LA-based schemes is that in the standard gradient method, the new operation point lies within a neighborhood

distance of the previous point. This is not the case for an adaptation algorithm based on stochastic principles, as the new operating point is determined by probability function and is therefore not considered to be near the previous operating point. This gives the algorithm higher ability to locate the global optimum. In general, the LA approach has two distinct advantages over classical hill climbing methods: (1) the parameter space need not be metric and (2) since the search space is conducted in the path probability space instead of parameter space, a global rather than a local optimum can be found.

5. Conclusion

In this paper, we studied the ability of LA-based schemes to escape from local minima when standard BP fails to find the global minimum. It is demonstrated through simulations that LA-based schemes compared to other schemes such as SAB, Super-SAB, Fuzzy BP, ASBP method, and VLR-method have a higher ability to escape from local minima. It must be mentioned that just as BP cannot guarantee convergence to the global minimum solution, neither can LA-based schemes. This is a problem inherent to a localized optimization technique such as steepest descent, of which backpropagation is a special case.

References

[1] P. Arabshahi, J.J. Choi, R.J. Marks, T.P. Caudell, Fuzzy control of backpropagation, Proceedings of 1st IEEE International Conference on Fuzzy Systems, (FUZZ-IEEE '92), IEEE Press, San Diego, (1992), pp. 967–972.

[2] H. Beigy, M.R. Meybodi, Adaptation of momentum factor and steepness parameter in backpropagation algorithm using fixed structure learning automata, Proceedings of the CSICC-99 Conference, Sharif University of Technology, Tehran, Iran, 1999, pp. 117–124.

[3] H. Beigy, M.R. Meybodi, Adaptation of momentum factor and steepness parameter in backpropagation algorithm using fixed structure learning automata, Int. J. Sci. Technol. (Scientia Iranica) 8 (4) (2001) 1–15.

[4] H. Beigy, M.R. Meybodi, M.B. Menhaj, Adaptation of learning rate in backpropagation algorithm using fixed structure learning automata, Proceedings of the ICEE-95 Conference, K.N. Tosi University of Technology, Tehran, Iran, 1998, pp. 117–123.

[5] E.K. Blum, Approximation of boolean functions by sigmoidal networks: part 1: XOR and other two variables functions, Neural Networks 1 (1989) 532–540.

[6] J.P. Cater, Successfully using peak learning rates of 10 (and greater) in BP networks with the heuristic learning algorithm, IEEE Proceedings of the First International Conference on Neural Networks, Vol. II, San Diego, 1987, pp. 645–651;

[6a] L.W. Chan, F. Fallside, An adaptive training algorithm for back propagation networks, Computer Speech and Language 2 (1987) 205–218.

[7] C. Darken, J. Chang, J. Moody, Learning rate schedules for faster stochastic gradients search, Proceedings of the 1992 IEEE Workshop Neural Networks for Signal Processing, IEEE Press, Piscataway, NJ, 1992, pp. 3–12.

[8] M.R. Devos, G.A. Orban, Self-adapting backpropagation, Proceedings of NeuroNimes, 1988, pp. 104–112.

[9] M.A. Franzini, Speech recognition with backpropagation, IEEE Proceedings of the Ninth Annual Conference on Engineering in Medicine and Biology, Boston, 1987, pp. 1702–1703.

[10] P. Frasconi, M. Gori, A. Tesi, Success and failures of backpropagation: a theoretical investigation, Technical Report, Dipartimento di Sistemi e Informatica, Universita di Firenze, Firenze, Italy, 1992.
[11] M. Gori, A. Tesi, On the problem of local minima in backpropagation, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 76–86.
[12] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks 1 (1988) 295–307.
[13] N. Kandil, K. Khorasani, R.V. Patel, V.K. Sood, Optimum learning rate for back propagation neural networks, in: P.K. Simpson (Ed.), Neural Networks Theory, Technology, and Applications, IEEE Press, 1996, pp. 249–251.
[14] S. Lakshmivarahan, Learning Algorithms: Theory and Applications, Springer, Newyork, 1981.
[15] M.B. Menhaj, M.H. Hagen, Rapid learning using modified BP algorithms for multi-layer feedforward neural nets, Proceedings of the ICEE-95 Conference, University of Science and Technology, Tehran, Iran, 1995.
[16] M.B. Menhaj, M.R. Meybodi, A novel learning scheme for feedforward neural nets. Proceedings of ICEE-95 Conference, University of Science and Technology, Tehran, Iran, 1995.
[17] M.B. Menhaj, M.R. Meybodi, Flexible sigmoidal type functions for neural nets using game of automata, Proceedings of the CSICC-96 Conference, Amirkabir University of Technology, Tehran, Iran, 1996, pp. 221–232.
[18] M.B. Menhaj, M.R. Meybodi, Application of learning automata to neural networks, Proceedings of the CSICC-96 Conference, Amirkabir University of Technology, Tehran, Iran, 1996, pp. 209–220.
[19] M.B. Menhaj, M.R. Meybodi, Using learning automata in backpropagation algorithm with momentum, Technical Report, CE Dept., Amirkabir University of Technology, Tehran, Iran, 1997.
[20] M.R. Meybodi, Results on strongly absolutely expedient learning automata, In: D.R. Mootes, R. Butrick (Eds.), Proceedings of the OU Inference Conference 86, Athens, Ohio University Press, Ohio, 1987, pp. 197–204.
[21] M.R. Meybodi, S. Lakshmivarhan, Optimality of a general class of learning algorithm, Inform. Sci. 28 (1982) 1–20.
[22] M.R. Meybodi, S. Lakshmivarhan, in: U. Herkenrath, D. Kalin, W. Vogel (Eds.), On a class of learning algorithms which have a symmetric behavior under success and failure, Lecture notes in statistics, Mathematical Learning Models Theory and Algorithms, Springer, Berlin, 1984, pp. 145–155.
[23] M.R. Meybodi, H. Beigy, New classes of learning automata based schemes for adaption of backpropagation algorithm parameters, International Journal of Neural Systems 12 (1) (2002) 45–68.
[24] A.G. Parlos, B. Fernandez, A.F. Atya, J. Muthusami, W.K. Tsai, An accelerated learning algorithm for multi-layer preception networks, IEEE Trans. Neural Networks 5 (1994) 493–497.
[25] M. Riedmiller, B. Heinrich, A direct method for faster backpropagation algorithm, Neural Networks 5 (1992) 465–471.
[26] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning Internal Representations by Error Propagation in Parallel Distributed Processing, MIT Press, Cambridge, MA, 1986.
[27] A. Sperduti, A. Starita, Speed up learning and network optimization with extended BP, Neural Networks 6 (1995) 365–383.
[28] G. Tesauro, B. Janssens, Scaling relationships in backpropagation learning, Complex Systems 2 (1) (1988) 39–44.
[29] T. Tollenaere, SuperSAB: fast adaptive backpropagation with good scaling properties, Neural Networks 3 (1990) 561–573.
[30] T.P. Vosl, J.K. Mangis, A.Z. Rigler, W.T. Zink, D.L. Alkon, Accelerating the convergence of backpropagation method, Biol. Cybernet. 59 (1988) 257–263.