Pakistan Journal of Applied Sciences 2(4):437-443, 2002

Utilization of Fixed Structure Learning Automata for Adaptation of Learning Rate in Backpropagation Algorithm

[1]Hamid Beigy, [1]M. R. Meybodi and [2]M. B. Menhaj
[1]Computer Engineering Department, [2]Electrical Engineering Department,
Amirkabir University of Technology, Tehran, Iran

Abstract: Error backpropagation training algorithm (BP) which is an iterative gradient descent algorithm is a simple way to train multilayer feedforward neural networks. Despite the popularity and effectiveness of this algorithm, its convergence is extremely slow. The main objective of this paper is to incorporate an acceleration technique into the BP algorithm for achieving faster rate of convergence. By interconnection of Fixed Structure Learning Automata (FSLA) to the feedforward neural networks, we apply Learning Automata (LA) scheme for adjusting the learning rate based on the observation of random response of neural networks. The feasibility of proposed method is shown through simulations on three learning problems: Exclusive-or (XOR), approximation of function sin(x), and digit recognition. These problems are chosen because they possess different error surfaces and collectively present an environment that is suitable to determine the effect of proposed method. The simulation results show that the adaptation of learning rate using this method not only increases the convergence rate of learning but it increases the possibility of bypassing the local minima .

Keywords: Neural Network, Backpropagation, Fixed Structure Learning Automata, Learning Rate

Introduction

Backpropagation algorithm is a systematic method for training multilayer neural networks (Rumelhart *et al.*, 1986). Despite 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-optimum learning rate. It is not easy to choose appropriate value of learning rate for a particular problem. The learning rate is usually determined by trial and error using the past experiences. If the learning rate is too small, convergence can be very slow, if too large, paralysis and continuos instability can result. Moreover the best value at the beginning of training may not be so good later. Thus several researches have suggested algorithms for automatically adjusting the learning rate as training proceeds.

Arabshahi *et al.*, (1992) proposed an error back-propagation algorithm in which the learning-rate is adapted. In this algorithm, learning-rate is a function of error and changes in the error. They proposed that the learning-rate be adjusted using a fuzzy logic control system, in which the error and changes in error are the inputs and changes in learning-rate is the output of fuzzy logic controller. Kandil *et al.*, (1996) used optimum, time-varying learning-rate for multilayer neural network by linearizing the neural network around weight vector at each iteration. Parlos *et al.*, (1994) proposed an accelerated learning algorithm for supervised training of multilayer neural networks named adaptive error back-propagation (ABP) algorithm. In their proposed algorithm the learning-rate is a function of the error and the error gradient. Cater (1987) suggested having different learning rate for different pattern. Franzini (1987), Vosl *et al.*, (1987), Te Sauro and Janssens (1988), Jacobs (1988) and Jutten *et al.* (1991)have proposed other schemes for adaptation of learning rate.

Often the mean-square error surfaces for backpropagation algorithm are multimodal. The learning automata is known to have well established mathematical foundation and global optimization capability (Narendra and Thathachar, 1989). This latter capability of learning automata can be used fruitfully to search a multimodal mean-square error surface. Recently Menhaj and Meybodi (1995 and 1996) have used variable structure learning automata (VSLA) to find the appropriate learning rate for the backpropagation training algorithm. In this approach a learning automata is associated the whole network to adapt the appropriate learning rate. It is shown that learning rate adapted in such a way not only increases the rate of the convergence of the network but it bypasses the local minimum in most cases.

In this paper, we use Fixed Structure Learning Automata (FSLA) to adjust the learning rate of the BP training algorithm in order to achieve higher rate of convergence and also higher rate of escaping from the local minima. By interconnection of learning automata to the feedforward neural networks, we apply learning automata scheme for adjusting the learning rate based on the observation of random response of the neural networks. The feasibility of proposed method is shown through simulations on three learning problems: exclusive-or (XOR), approximation of function sin(x), and digit recognition. These problems are chosen because they possess different error surfaces and collectively present an environment that is suitable to determine the effect of proposed method. Simulation results on these problems show that adaptation of learning rate using this method not only increases the convergence rate but it increases the likelihood of bypassing the local minima. It must be noted that our studies show that FSLA approach performs much

better than the VSLA approach reported in (Menhaj and Meybodi, 1995).
The rest of the paper is organized as follows: Section 2 briefly presents the basic backpropagation algorithm. The fixed structure learning automata is introduced in section 3. Section 4 presents the proposed method. Simulation results and discussion are given in section 5 and 6. Section 7 concludes the paper.

Backpropagation Algorithm: Given a set of input/output pair of data $\{(X_p, T_p) \mid p = 1, 2, \ldots, P\}$, standard learning rule for multilayer feedforward neural net is the backpropagation algorithm which can be summarized as follows. Consider a feedforward neural network with L layers (i.e. 0_{th} layer denotes the input layer, L_{th} layer represents the output layer and there are L - 1 hidden layers), where the number of units in k_{th} layer is represented by N[k]. The network first uses the input vector X_p (which is equal to U [0]) to produce its own vector U[L] and then compares this with the desired output, or target vector T_p. If there is no difference (E = 0), no learning takes place, otherwise the weights are changed to reduce the difference. The error from output layer is propagated to hidden layers and the error for every units is estimated. Backpropagation algorithm is given as (Hush and Horn, 1993) (Fig. 1). In this algorithm μ is the learning rate ; f and f′ denote the activation function and the derivative of activation function, respectively.

Learning Automata (LA): The automatons approach to the learning involves the determination of an optimal action out of a set of allowable actions. These actions performed on abstract random environment. The environment responds to the action by producing an output, belonging to the set of allowable outputs, which are probabilistically related to the input action (Narendra and Thathachar, 1989). The term environment as commonly defined refers to aggregate of all external conditions and influences affecting the life and development of an organism. Narendra and Thathachar (1989) defined mathematically an environment by a triple $\langle \underline{\alpha}, \underline{C}, \underline{\beta} \rangle$, where $\underline{\alpha}=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents a finite input set, $\underline{C}=\{c_1, c_2, \ldots, c_r\}$ shows a set of probabilities, where each element c_i of $\underline{C}$ corresponds to one input action α_i , and $\underline{\beta} = \{\beta_1, \beta_2\}$ represents a binary output set.

```
Procedure StandardBackPropagationAlgorithm
  Initialize the weights to small random values
  repeat
     for all training pair (X, T) in training set do
       Call FeedForward
       Call ComputeGradiant
     End for
     Call UpdateWeights
  until termination condition Satisfied
End Procedure
Procedure FeedForward
   for Layer = 1 to L do
     for Node = 1 to N [Layer] do
       Activation = 0
       for I = 0 to N [Layer - 1] do
         Activation=Activation + W[Layer, Node, I] * U [Layer-1,I]
       End for
       U [Layer, Node] = f (Activation);
     End for
   End for
End Procedure
Procedure ComputeGradiant
  for Layer = L to 1 do
    for Node = 1 to N [Layer] do
      E [L, Node] = 0 ;
      if Layer = L then
        E [L, Node] = U [L, Node] - T [Node];
      else
        for m = 1 to N [Layer + 1] do
          E [L, Node]= E [L, Node] +E [Layer + 1, m] *
          f'(U [Layer + 1, m]) * W [Layer 1, m , Node]
        End for
      End if
    End for
  End for
End Procedure
Procedure UpdateWeights
  for all W [Layer, J, I] do
    W^{k+1} [Layer, J, I] = W^k [Layer, J, I] - μ* G [Layer, J, I]
  End for
End Procedure
```

Fig. 1: The Standard Backpropagation Algorithm

The automaton takes in a sequence of inputs and puts out a sequence of actions. The automatans mathematically defined by $\langle \underline{\phi}, \underline{\alpha}, \underline{\beta}, F(.,.), H(.,.) \rangle$, where $\underline{\phi}$ is a set of internal states, $\underline{\beta}$ a set of input actions, $\underline{\alpha}$ a set of outputs, $F(.,.) : \underline{\phi} \times \underline{\beta} \rightarrow \underline{\phi}$ is a function that maps the current input and current state into next state, and $H(.,.) : \underline{\phi} \rightarrow \underline{\alpha}$ in a function that maps the current state into the current output. In such an automatons the input and current state determine the next state as well as the current output. The automatons and environment connected in a feedback arrangement as shown in the Fig. 2.

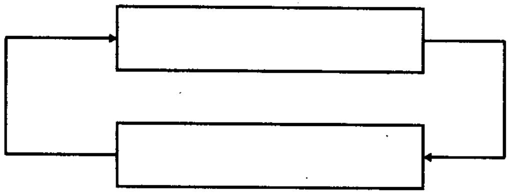


Fig. 2: The Automata and its Environment

The output of environment $\beta(n)$ forms the input to the automatans and the action of automatans $\alpha(n)$ provides the input to the environment. If the probability of the transition from one state to another state and probabilities of correspondence of action and state are fixed, the automatons is said fixed-structure automata and otherwise the automatons is said variable-structure automata. We summarize some of fixed-structure learning automata which are used in this paper in the following subsections.

The Two-state Automata : $L_{2,2}$: This automata has two states, ϕ_1 and ϕ_2 and two actions α_1 and α_2, as shown in Fig. 3. The automata 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).

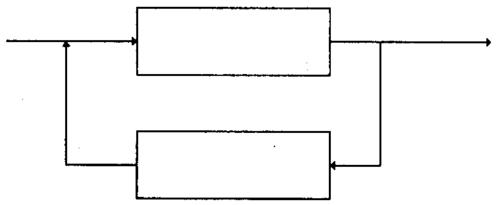


Fig. 3: The Two-State Automata

An automata that uses this strategy is refereed as $L_{2,2}$ where the first subscript refers to the number of states and second subscript to the number of actions. The environment is characterized by the set of penalty probabilities $\{c_1, c_2\}$ where c_i $(i = 1, 2)$ corresponds to the probability of getting a response $\beta = 1$ from the environment when the input is α_i. The simple strategy used by automata implies that it continues to perform whatever action it was using earlier as long as the response is good but changes to the other action as soon as the response is bad.

The Two-action Automata with Memory : $L_{2N,2}$: This automata 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 the another. The procedure described above is one convenient method of keeping track of performance of the actions α_1 and α_2. As such, N is called memory depth associated with each action, and automata is said to have a total memory of 2N. The state transition graph of this automata is shown in fig. 4.

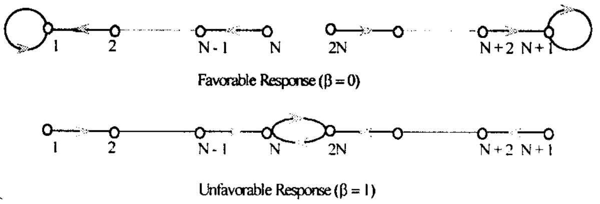

1
2
N - 1
N
2N
N + 2
N + 1
Favorable Response ($\beta = 0$)
1
2
N - 1
N
2N
N + 2
N + 1
Unfavorable Response ($\beta = 1$)

Fig. 4: The State Transition Graph for $L_{2N,2}$

For every favorable response, the state of automata moves deeper into the memory of corresponding action, and for an unfavorable response, moves out it. This automata can be extended to multiple action automata and this automata is named $L_{KN,K}$ automata.

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

The Krylov Automata: This automata has state transition that are identical to the $L_{2N,2}$ automata when the output of the environment is favorable. However, when the response of the environment is unfavorable, a state ϕi $(i \neq 1, N, N+1, 2N)$ 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, automata state moves ϕ_{N-1} to ϕ_{2N} and with the same probability 0.5. When i = 2N, automata state moves ϕ_{2N-1} to ϕ_N and with the same probability 0.5.

The Proposed Method: In our proposed method, we use the fixed-structure learning automata for adjusting the learning-rate. The interconnection of learning automata and neural network is shown in Fig. 5. The neural network is the environment for the learning automata. The learning automata according to the amount of the error recieved from neural network adjusts the learning rate of the backpropagation algorithm. The actions of the automata correspond to the values of the learning rate and input to the automata is some function of the error in the output of neural network.

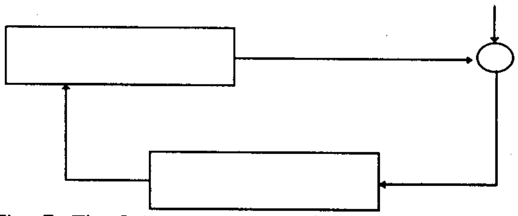


Fig. 5: The Interconnection of LA with Neural Network

A function of error between the desired output and network output is considered as the response of environment. A window on the past values of the errors are swiped and the average value of the error in this window computed and compared to a threshold value. If the difference of the average value in the two last steps is less than the threshold value, the response of the environment is favorable and if the difference of average value in the last two steps is greater than the threshold value, the response of the environment is unfavorable.

Algorithms of Fig. 6 and 7 describe how fixed structure learning automata can be used for determination of learning rate of backpropagation algorithm. In the first algorithm a single learning automata is responsible for determination of the learning rate 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 using separate learning automata for each layer of the network produces better results than when we use a single learning automata. These two algorithms have been tested on several problems and the results are presented in the following section. In these algorithms at each iteration one input of the training set is presented to the neural networks, then the network's response is computed and the weights are corrected. The amount of the correction is proportional to the learning rate.

Simulation: In order to evaluate the performance of the proposed method simulations are carried out on three learning problems: Exclusive-or (XOR), approximation of function sin(x), and digit recognition. The results are compared with results obtained from standard BP and variable structure learning automata based algorithm reported in (Menhaj and Meybodi, 1995 and 1996).

```
Procedure OneLA_BPAlgorithm
    Initialize the weights to small random values
    Initialize the parameters of learning automata
    repeat
        for all training pair (X, T) in training set do
            Call FeedForward
            Call ComputeGradiant
        End for
        Call UpdateWeights
        Call ComputeResponseOfEnvironment
        Call AdjustLearningRate
    until termination condition satisfied
End Procedure
```

Fig. 6 : Backpropagation algorithm with a single LA

```
Procedure TwoLA_BPAlgorithm
    Initialize the weights to small random values
    Initialize the parameters of two learning automatons
    repeat
        for all training pair (X, T) in training set do
            Call FeedForward
            Call ComputeGradiant
        End for
        Call UpdateWeights
        Call ResponseOfEnvironmentForOutputLayer
        Call AdjustLearningRateOfOutputLayer
        Call ResponseOfEnvironmentForHiddenLayer
        Call AdjustLearningRateOfHiddenLayer
    until termination condition satisfied
End Procedure
```

Fig. 7 : Backpropagation Algorithm with Two LA

XOR : The network architecture used for solving problem consist of 2 input units, 2 hidden units, and 1 output unit. Actions in these simulations are selected in interval [0, 1] with equal distance. Fig. 8 compare the effect of different automata on the performance of learning. This Fig. shows that the fixed structure learning automatons are more effective than variable structure automatons, which are reported in (Menhaj and Meybodi, 1995 and 1996).
For all automatons in this simulation the memory depth of 4, and the threshold of 0.01, and window size of 1 is chosen. For the Tsetline automata the number of action 4 and for linear reward-penalty automata the reward and penalty coefficient 0.001 and 0.0001 are chosen. Note that for this application Krylov automata is the best automata for adaptation of learning rate. In Fig. 9 a $L_{KN,K}$ automata is associated to output layer and a $L_{KN,K}$ automata to the hidden layer. For this simulation, number of action of 4, the memory depth of 4, window size of 1 and threshold of 0.001 are chosen for both automatons.

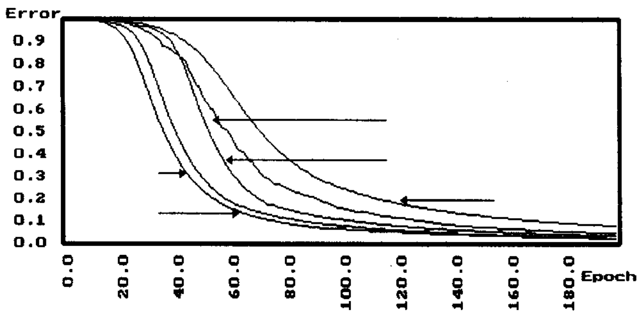


Fig. 8
a) Standard Backpropagation
b) Tsetline Automata
c) Krinsky Automata
d) Krylov Automata
e) Linear Reward-Penalty Automata

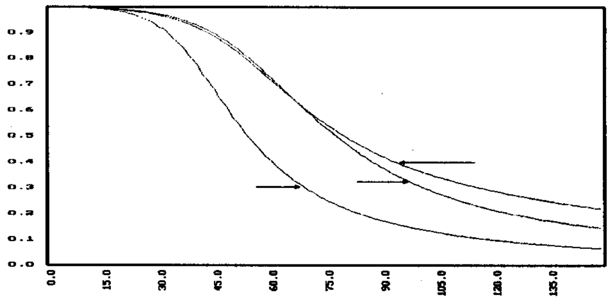


Fig. 9
a) Standard BP with Learning Rate of 0.7
b) One Learning Automata
c) Two Learning Automatons

F(x) = SIN (x): This is a function approximation problem. A training set of 20 samples is selected uniformly over period 0..2π. Simulations are carried out for different networks: a 1-5-1 (Fig. 10) network and 1-10-1 (Fig. 11) network. For both networks a FSLA is used for adaptation of learning rate. For these simulations number of action of 4, memory depth of 4, threshold of 0.001, and window size of 1 are chosen. In (Menhaj and Meybodi, 1996). was reported that a 1-5-1 network for this example that fails to train the network when using the standard BP algorithm can be trained when variable structure learning automata is incorporated in the BP for the adaptation of the learning rate. We have also obtained the same result when FSLA is used (Fig. 10 and 11).

8 $\times$ 8 Dot Numeric Font Learning: We have ten numbers 0, ... , 9, and each represented by a 8 $\times$ 8 grid of black and white dot as shown in Fig. 12 (Sperduti and Starita, 1993). The network must learn to distinguish these classes. The network architecture used for this problems consists of 64 input units which are connected to 6 hidden units which are connected to 10 output units.

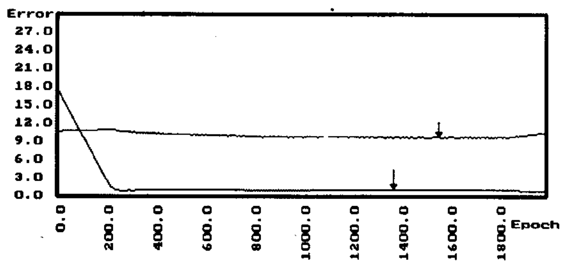


Fig. 10
a) Standard Bp b) BP With One Automa

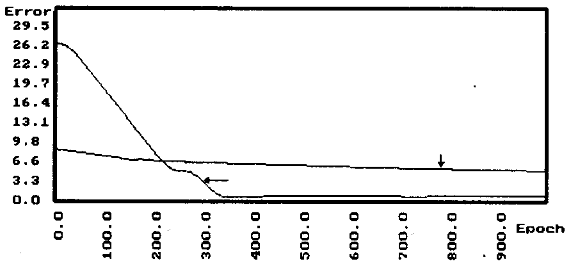


Fig. 11
a) Standard Bp B) BP With One Automata

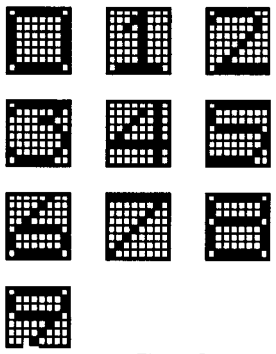


Fig. 12:

Fig. 13 compare the effect of different automata on the performance of learning. This Fig. shows that the fixed structure learning automatons are more effective than variable structure automatons, which are reported in (Menhaj and Meybodi, 1995 and 1996) For all automatons in this simulation the threshold of 0.01 and window size of 1 is chosen. For the Tsetline automata the number of action 4, the memory depth of 4 and for Krinsky and krylov automatons the memory depth of 4 are chosen. For linear reward-penalty automata the reward and penalty coefficient 0.001 and 0.0001 are chosen. Note that for this application Krylov automata is the best automata for adaptation of learning rate. Fig. 14 shows the effect of association of different automatons to different layers on the performance of learning. In this simulation a Tsetline learning automata is associated to the hidden layer and the effect of association of different learning automatons are shown. For all automatons in this simulation the threshold of 0.01 and window size of 1 is chosen. For the Tsetline automata the number of action 4, the memory depth of 4 and for Krinsky and TsetlineG automatons the memory depth of 4 are chosen.

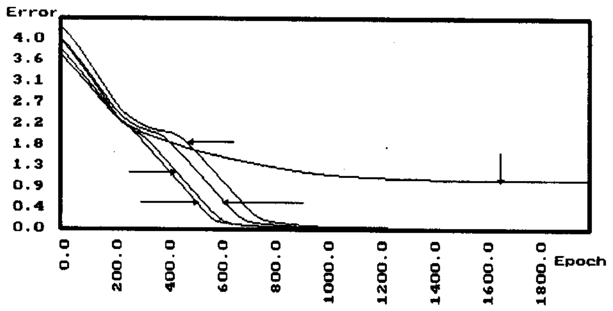


Fig. 13
a) Standard BP b) Tsetline Automata
c) Krinsky Automata d) Krylov Automata
e) Linear Reward-Penalty Automata

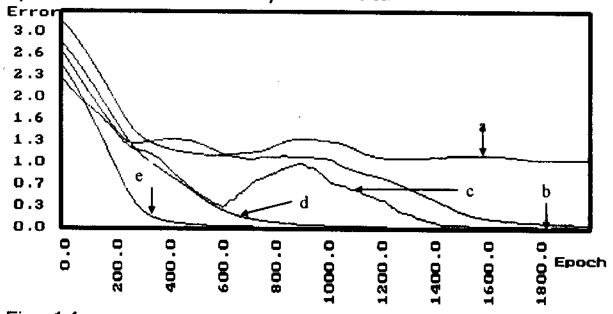


Fig. 14
a) Standard BP
b) One Tsetline Automata
c) (Tsetline, Krinsky) Automatons
d) (Tsetline, Tsetline) Automatons
e) (Tsetline, TsetlineG) Automatons

Table 1 shows the effect of association of different automatons to different layers on the performance of learning For all automatons in this simulation the threshold of 0.01 and window size of 1 is chosen. For the Tsetline automata the number of action 4, the memory depth of 4 and for Krinsky and krylov automatons the memory depth of 4 are chosen. The error of standard BP after 3000 epochs is 0.734397. The plot for each simulation is averaged over 200 runs. For more simulations refer to (Beigy *et al.*, 1997)

Table 1

Hidden Layer LA	Output Layer LA	Error After 3000 Epochs	Epochs For Error of 0.01
Tsetline	Tsetline	1.075709	
Tsetline	Krinsky		1466
Tsetline	Krylov	1.654468	
Tsetline	TsetlineG		413
Krinsky	Tsetline		1863
Krinsky	Krinsky		935
Krinsky	Krylov		724
Krinsky	TsetlineG	2.467577	
Krylov	Tsetline		1049
Krylov	Krinsky		594
Krylov	Krylov	1.494493	
Krylov	TsetlineG	2.344173	
TsetlineG	Tsetline		1031
TsetlineG	Krinsky		1089
TsetlineG	Krylov		918
TsetlineG	TsetlineG		1029

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 chose a problem in which local minima are occurred frequently (Gori and Tesi, 1992). This example considers the sigmoidal network for the XOR boolean function with the quadratic cost function and the standard learning environment The training set of this problem is given in table 2. 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 (Frasconi *et al* ., 1992).

Table 2

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

Fig. 15 shows these configurations. The error surface of the network as a function of weights $w_{2,1,1}$ and $w_{1,1,1}$ is given in Fig. 16.

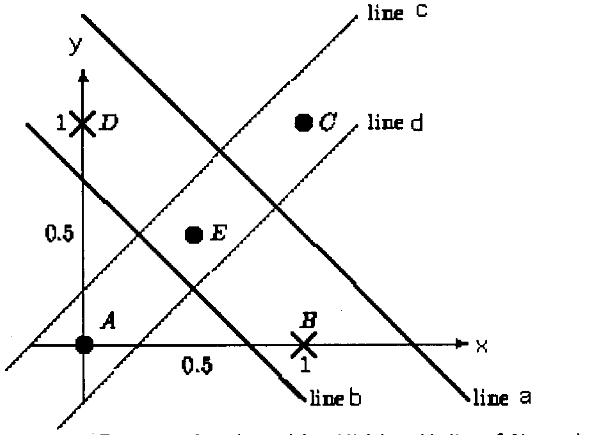


Fig. 15: Lines Produced by Hidden Units of Neural Network

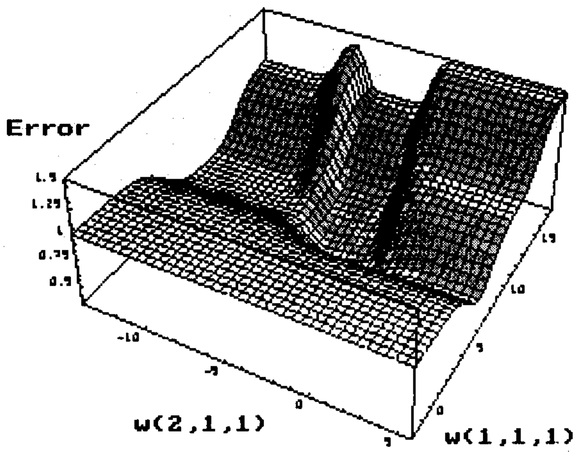


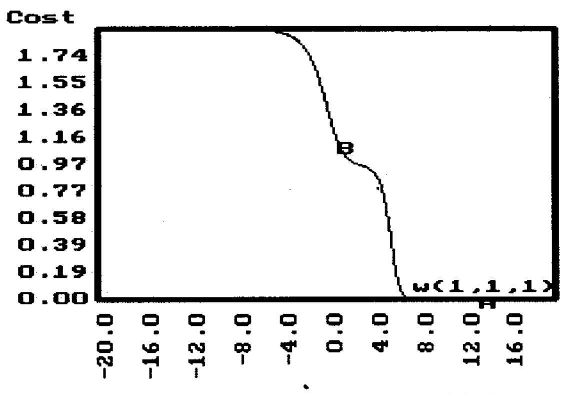
Fig. 16: Error Surface as a Function of Weights $W_{2,1,1}$ and $W_{1,1,1}$

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. 16.

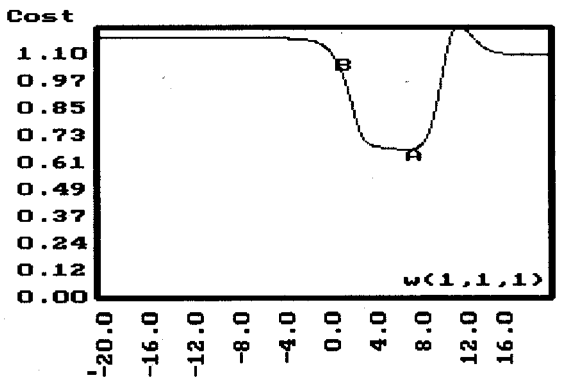
Table 2

Algorithm	Not Converged	Converged
BP	20	0
SAB	20	0
SuperSAB	20	0
VLR	18	2
FuzzyBP	18	2
Tsetline	18	2
Krinsky	14	6
Krylov	17	3
L_{R-P}	16	4
Tsetline-TsetlineG	18	2
Tsetline-Krylov	18	2
Tsetline-Krinsky	15	5
Tsetline-Tsetline	15	5

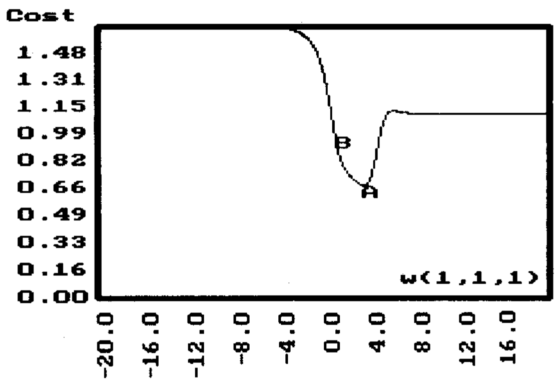
To show how well the LA based adaptation algorithm escapes local minima we test eight different LA based algorithms, 4 from class A and 4 from class B and compare their results with the standard BP and five other known adaptation methods: SAB(Jacobs, 1988), SuperSAB(Jacobs, 1988), VLR method(Menhaj and Hagen, 1995), and fuzzy BP(Arabshahi *et al.*, 1996) ((Arabshahi *et al.*, 1992). The result of simulation for 20 runs are summarized in table 1. Note that for standard BP and also for standard BP when SAB or SuperSAB method is used to adapt the learning rate none of the 20 runs converges to the global minima. Among the non-LA based methods the fuzzy BP and VLR methods method perform the best. For each of these methods 2 out of 20 runs converges to global minima which is comparable to some of the LA based schemes we have tested. The best result is obtained for Krinsky scheme from class A for which 6 runs out of 20 runs converge to global minimum. The next best result belongs to L_{RP} scheme. Fig. 17a through 17c show some typical runs. Each run uses a random initial point near the local minima. In these Fig. the initial point is denoted by letter 'B' and the converged point is denoted by letter 'A'. The curves in these figures are obtained by projecting the error surface on axis $w_{1,1,1}$. 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 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 than parameter space, a global rather than a local optimum can be found.



a) Krinsky Scheme (Converged to Global Minima)



b) Krinsky Scheme (Stuck At Local Minima)



c) BP Algorithm (Stuck at Local Minima)
Fig. 17

Conclusion

In this paper, we have proposed the use of FSLA for adaptation of learning rate of BP algorithm. We have demonstrated through simulations that the use of FSLA for adaptation of learning rate of BP algorithm not only increases the rate of convergence by a large amount, but it is possible to compute a new point that is closer to the optimum than the point computed by BP algorithm. In the all problems we studied so far, the convergence of BP which uses FSLA or VSLA for adaptation of learning rate is faster than the standard BP. All the simulations show this important fact that use of FSLA performs much better than the use of VSLA for adaptation of learning rate.

References

A. G. Parlos, B. Fernadez, A. F. Atya, J. Muthusami, and W. K. Tsai, 1994. "An Accelerated Learning Algorithm for Multi-Layer Preceptron Networks", IEEE Trans. on Neural Networks, 3: pp. 493-497.

A. Sperduti and A. Starita, 1993. "Speed Up Learning and Network Optimization with Extended BP", Neural Networks, 6: 365-383.

Arabshahi, P. *et al.*, 1996. Fuzzy Parameter Adaptation in Optimazation: Some Neural Net Training Examples, IEEE Computational Science and Engineering, pp. 57-65.

C. Jutten, A. Guerin, and H. L. Nguyen Thi, 1991. "Adaptive Optimazation of Neural Algorithms", In LNCS 540, A. Prietoter(Ed), Artifitial Neural Networks, Springer-Verlag, pp. 54-61.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams, 1986. "Learning internal representations by error propagation", In Parallel distributed processing, Cambridge, MA: MIT Press.

D. R. Hush, and B. G. Horn, 1993. "Progress in Supervised Neural networks: What's new Since Lippman", IEEE Sig. Proc. Mag., pp. 8-39.

Frasconi, P., Gori, M., and Tesi, A., 1992. Success and Failures of Backpropagation: A Theoretical Investigation. Technical Report, Dipartimento di Sistemi e Information, Universita di Firenze, Firenze, Italy.

G. Te Sauro and B. Janssens, 1988. "Scaling Relationships in Backpropagation Learning", Complex Systems. pp. 39-44, 1988.

Gori, M. and Tesi, A., 1992. On the Problem of Local Minima in BP. IEEE Trans. on Pattern Analysis and Machine Intelligence, 14, pp. 76-86.

H. Beigy, M. R. Meybodi, and M. B. Menhaj, 1997. "A New Method for Determination of Learning Rate", Technical Report, Computer Eng. Dept., Amirkabir University of Technology, Tehran, Iran.

J. P. Cater, 1987. "Successfully Using Peak Learning Rates of 10 (and Greater) in Backpropagation Networks with the Heuristic Learning Algorithm", IEEE Proc. of First Int. Conf. on Neural Networks, pp. 645-651.

Jacobs, R. A., 1988. Increased Rates of Convergence Through Learning Rate Adaptation. Neural Networks, 1, pp. 295-307.

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

M. A. Franzini, 1987. "Speech Recogntion with BP", IEEE Proc. of Ninth Annual Conf. on Eng. in Medicine and Biology, pp. 1702-1703.

M. B. Menhaj and M. R. Meybodi, 1995. "A Novel Learning Scheme for Feedforward Neural Nets", Proc. of ICEE-95, Tehran, Iran.

M. B. Menhaj and M. R. Meybodi, 1996. "Flexible Sigmodal Type Functions for Neural Nets using Game of Automata", Proc. of CSIC-96, Tehran, Iran , pp. 221-232.

Menhaj, M. B. and Hagen, M. H., 1995. Rapid Learning Using Modified BP Algorithms for Multi-Layer Feedforward Neural Nets. Proc. of ICEE-95 Conference, Uni. of Science and Tech. Tehran, Iran.

N. Kandil, K. Khorasani, R. V. Patel, and V. K. Sood, 1996. "Optimum Learning Rate for Back propagation Neural Networks", In Neural Networks Theory, Technology, and Applications, Edited by P. K. Simpson,pp. 249-25.

P. Arabshahi, J. J. Choi, R. J. Marks, and T. P. Caudell, 1992. "Fuzzy Control of Back propagation", Proc. of IEEE Int. Conf. on Fuzzy Systems, pp. 967-972.

S. Haykin, 1994. Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Company.

T. P. Vosl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, 1987. "Accelerating the Convergance of Backpropagation Method", Bilogical Cybernitics, pp. 257-263.