

Adaptation of Learning Rate in Backpropagation Algorithm Using Fixed Structure Learning Automata

Hamid Beigy* M. R. Meybodi* M. B. Menhaj⁺

Amirkabir University of Technology

* Computer Eng. Department

+Electrical Eng. Department

Tehran, Iran

Email: beigy@cews1.aku.ac.ir

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 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 problem 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 bypasses the local minima in most cases.

Keywords: Neural network, Backpropagation, Fixed structure learning automata, Learning rate

1. Introduction

Backpropagation algorithm is a systematic method for training multilayer neural networks [1]. 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 continuous 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. [2] 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. [3] used optimum, time-varying learning-rate for multilayer neural network by linearizing the neural network around weight vector at each iteration. Parlos et al. [4] 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 [5] suggested having different learning rate for different pattern. Franzini [6], Vosl et al. [7], and Tesnuro and Janssens [8], Jacob [9], and Jutten et al. [16] 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 [10]. This latter capability of learning automata can be used fruitfully to search a multimodal mean-square error surface. Recently Menhaj and Meybodi [11,12] 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 problem 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 [11].

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. Section 6 concludes the paper.

2. Backpropagation Algorithm

Given a set of input/output pair of data $\{(X_p, T_p) | p = 1, 2, \dots, P\}$, 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 [13] (figure 1). In this algorithm μ is the learning rate; f and f' denote the activation function and the derivative of activation function, respectively.

```

Procedure StandardBackPropagationAlgorithm
  Initialize the weights to small random values
  repeat
    for all training pair  $(X, T)$  in training set do
      Call FeedForward
      Call ComputeGradient
    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  $l = 0$  to N [Layer - 1] do
        Activation = Activation + W[Layer, Node, l] * U [Layer-1, l]
      End for
      U [Layer, Node] = f (Activation);
    End for
  End for
End Procedure

```

```

Procedure ComputeGradient
  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] -  $\mu$  * G [Layer, J, I]
  End for
End Procedure

```

Figure 1: The standard backpropagation algorithm

3. Learning Automata

The automata 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 [10]. 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 [10] defined mathematically an environment by a triple $(\underline{\alpha}, \underline{C}, \underline{\beta})$, where $\underline{\alpha} = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ represents a finite input set, $\underline{C} = \{c_1, c_2, \dots, c_T\}$ shows a set of probabilities, where each element c_j of \underline{C} corresponds to

one input action α_i , and $\underline{\beta} = \{\beta_1, \beta_2\}$ represents a binary output set.

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

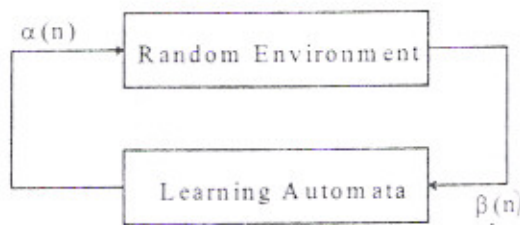


Figure 2: The automaton and its environment

The output of environment $\beta(n)$ forms the input to the automaton and the action of automaton $\alpha(n)$ 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 automaton is said fixed-structure automaton and otherwise the automaton is said variable-structure automaton. We summarize some of fixed-structure learning automata which used in this paper in the following subsections.

3.1- 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 , as shown in figure 3. 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).

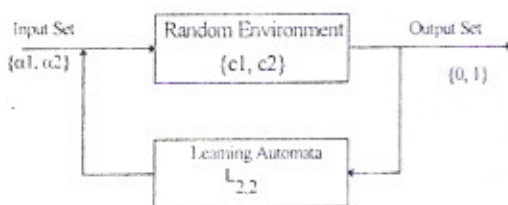


Figure 3: The two-state automata

An automaton that uses this strategy is referred 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.

3.2- 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 automaton $L_{2,2}$ 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 automaton 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 automaton is said to have a total memory of $2N$. The state transition graph of this automaton is shown in fig. 4.

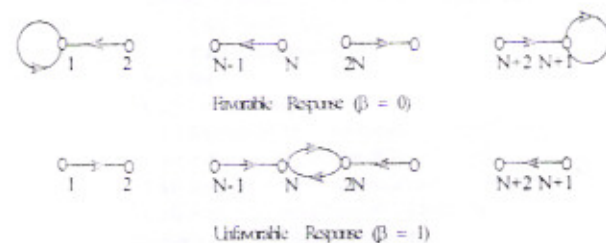


Figure 4: The state transition graph for $L_{2N,2}$

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

3.3- 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 $i = 1, \dots, N$) passes to the state ϕ_1 and any state ϕ_i (for $i = N+1, \dots, 2N$) 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.

3.4- The Krylov automata

This automaton has state transition 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 $i=N$, 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.

4. 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 figure 5. 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 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.

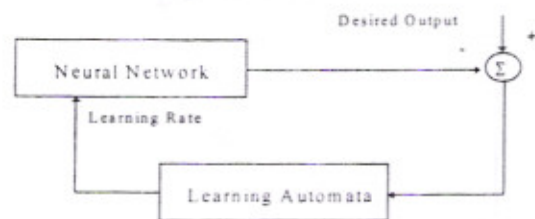


Figure 5: The interconnection of learning automata with neural network

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. The following algorithm is the backpropagation algorithm in which the learning automata is responsible for the adaptation of the learning rate. In this algorithm 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.

Algorithms of figures 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 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 following section.

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 ComputeGradient
  End for
  Call UpdateWeights
  Call ComputeResponseOfEnvironment
  Call AdjustLearningRate
until termination condition satisfied
End Procedure
```

Figure 6 : Backpropagation algorithm with a single learning automata

Procedure TwoLA_BPAlgorithm

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

Fig. 7 : Backpropagation algorithm with two learning automata

5. 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 [11, 12].

1- 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. Figure 8 compare the effect of different automata on the performance of learning. This figure shows that the fixed structure learning automata are more effective than variable structure automata, which are reported in [11, 12]. For all automata 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 figure 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.

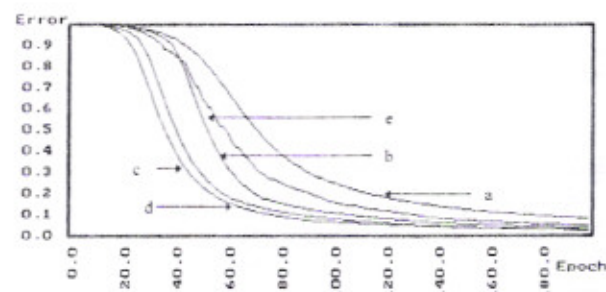


Figure 8

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

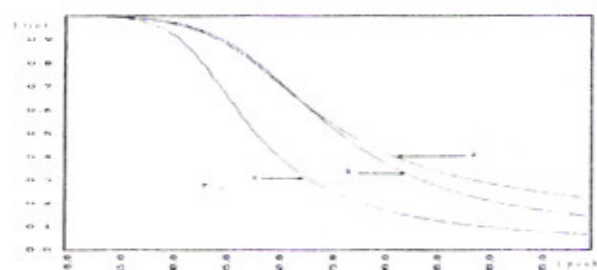


Figure 9

- a) Standard BP with learning rate of 0.7
b) one learning automata c) two learning automatons

2- $F(x) = \sin(x)$: This is a function approximation problem. A training set of 20 samples is selected uniformly over period $0..2\pi$. Simulations are carried out for different networks: a 1-5-1 (figure 10) network and 1-10-1 (figure 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 [12] 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 (figures 10 and 11).

3- 8×8 Dot Numeric Font Learning: We have ten numbers 0, ..., 9, and each represented by a 8×8 grid of black and white dot as shown in figure 12 [14]. 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.

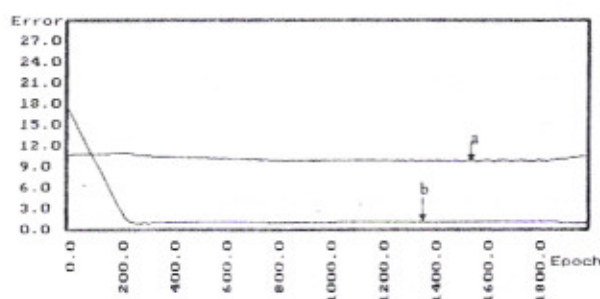


Figure 10

- a) Standard Bp b) BP with one automata

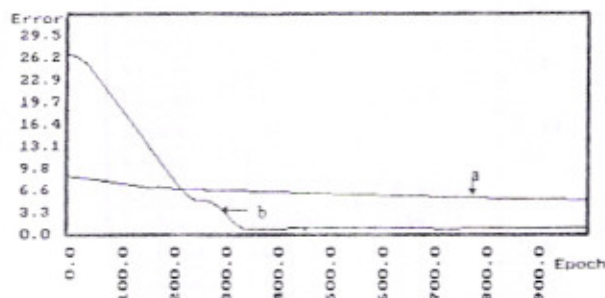


Figure 11

- a) Standard Bp b) BP with single automata

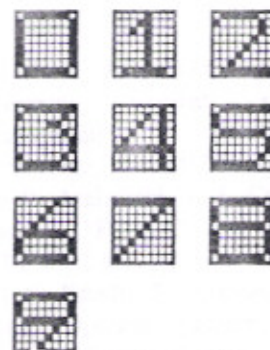


Figure 12:

Figure 13 compare the effect of different automata on the performance of learning. This figure shows that the fixed structure learning automatons are more effective than variable structure automatons, which are reported in [11, 12]. 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. Figure 14 shows the effect of association of different automatons to different layers on the performance of learning. In this simulation a

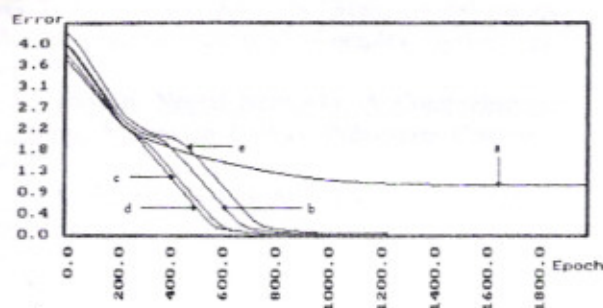


Figure 13

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

Tsetline learning automata is associated to the hidden layer and the effect of association of different learning automata are shown. For all automata 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 automata the memory depth of 4 are chosen.

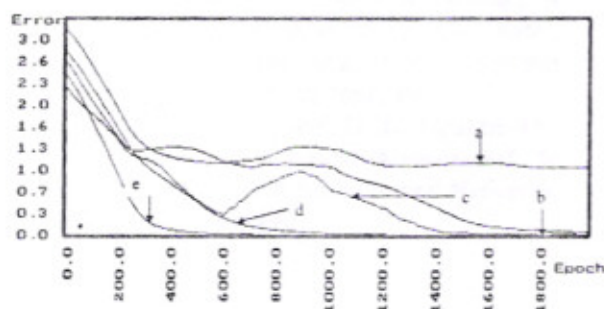


Figure 14

- a) Standard BP b) One Tsetline Automata
c) (Tsetline, Krinsky) Automata
d) (Tsetline, Tsetline) Automata
e) (Tsetline, TsetlineG) Automata

Table 1 shows the effect of association of different automata to different layers on the performance of learning. For all automata 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 automata 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 [15].

6. Conclusions

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.

Table 1

| Hidden Layer Automata | Output Layer Automata | 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 |

7. References

- [1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", In *Parallel distributed processing*, Cambridge, MA: MIT Press, 1986.
- [2] P. Arabshahi, J. J. Choi, R. J. Marks, and T. P. Caudell, "Fuzzy Control of Back propagation", *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pp. 967-972, 1992.
- [3] N. Kandil, K. Khorasani, R. V. Patel, and V. K. Sood, "Optimum Learning Rate for Back propagation Neural Networks", In *Neural Networks Theory, Technology, and Applications*, Edited by P. K. Simpson, pp. 249-251, 1996.
- [4] A. G. Parlos, B. Fernandez, A. F. Atya, J. Muthusami, and W. K. Tsai, "An Accelerated Learning Algorithm for Multi-Layer Preceptron Networks", *IEEE Trans. on Neural Networks*, Vol. 5, No. 3, pp. 493-497, 1994.
- [5] J. P. Cater, "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*, Vol. II, pp. 645-651, 1987.
- [6] M. A. Franzini, "Speech Recognition with BP", *IEEE Proc. of Ninth Annual Conf. on Eng. in Medicine and Biology*, pp. 1702-1703, 1987.
- [7] T. P. Vosl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the Convergence of Backpropagation Method", *Biological Cybernetics*, pp. 257-263, 1987.

- [8] G. Te Sauro and B. Janssens, "Scaling Relationships in Backpropagation Learning", *Complex Systems*, pp. 39-44, 1988.
- [9] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Company, 1994.
- [10] K. S. Narendra and M. A. L. Thathachar, Learning Automata : An Introduction, Prentice-hall, Englewood cliffs, 1989.
- [11] M. B. Menhaj and M. R. Meybodi, "A Novel Learning Scheme for Feedforward Neural Nets", *Proc. of ICEE-95*, Tehran, Iran, 1995.
- [12] M. B. Menhaj and M. R. Meybodi, "Flexible Sigmodal Type Functions for Neural Nets using Game of Automata", *Proc. of CSIC-96*, Tehran, Iran , pp. 221-232, Dec. 1996.
- [13] D. R. Hush, and B. G. Horn, "Progress in Supervised Neural networks: What's new Since Lippman", *IEEE Signal Processing Mag.*, pp. 8-39, Jan. 1993.
- [14] A. Sperduti and A. Starita, "Speed Up Learning and Network Optimization with Extended BP", *Neural Networks*, Vol. 6, pp. 365-383, 1993.
- [15] H. Beigy, M. R. Meybodi, and M. B. Menhaj , "A New Method for Determination of Learning Rate", *Technical Report*, Computer Eng. Dept., Amirkabir University of Technology, Tehran, Iran , 1997.
- [16] C. Jutten, A. Guerin, and H. L. Nguyen Thi, "Adaptive Optimazation of Neural Algorithms", In *LNCS 540*, A. Prietoter(Ed), *Artifitial Neural Networks*, Springer-Verlag, pp. 54-61, 1991.