LYAPUNOV THEORY BASED ADAPTIVE LEARNING ALGORITHM FOR MULTILAYER NEURAL NETWORKS

Nurettin Acır[*], *Engin Cemal Mengüç*[†]

Abstract: This paper presents a novel weight updating algorithm for training of multilayer neural network (MLNN). The MLNN system is first linearized and then the design procedure is proposed as an inequality constraint optimization problem. A well selected Lyapunov function is suitably determined and integrated into the constraint function for satisfying asymptotic stability in the sense of Lyapunov. Thus, the convergence capability of training algorithm is improved by using a new analytical adaptation gain rate which has the ability to adaptively adjust itself depending on a sequential square error rate. The proposed algorithm is compared with two types of backpropagation algorithms and a Lyapunov theory based MLNN algorithm on three benchmark problems which are XOR, 3-bit parity, and 8-3 encoder. The results are compared in terms of number of learning iterations and computational time required for a specified convergence rate. The results clearly indicate that the proposed algorithm is much faster in convergence than other three algorithms. The proposed algorithm is also comparatively tested on a real iris image database for multiple-input and multiple-output classification problem and the effect of adaptation gain rate for faster convergence and higher performance is verified.

Key words: *Lyapunov stability theory, multilayer neural network, Lagrange multiplier theory, adaptive learning.*

Received: February 11, 2014 **DOI:** 10.14311/NNW.2014.24.035
Revised and accepted: December 4, 2014

1. Introduction

This study is concerned with the training of a feedforward multilayer neural network (MLNN). The most popular methods for training MLNN are the gradient based backpropagation (BP) algorithms [9, 10, 15, 19]. The main disadvantage of gradient descent based method is its inability to guarantee global minimum and

[*]Nurettin Acır, Bursa Technical University, Electrical and Electronics Engineering Department, 16190, Bursa, Turkey, E-mail: nurettin.acir@btu.edu.tr, Tel.: +90 224 314 18 47

[†]Engin Cemal Mengüç – Corresponding author, Nigde University, Electrical and Electronics Engineering Department, 51245, Nigde, Turkey, E-mail: ecmenguc@nigde.edu.tr, Tel.: +90 388 225 40 13

its slow convergence rate [8]. Theoretically, these algorithms suffer from local minima problem [14, 16, 22]. Meanwhile some algorithms, which are fast or capable of achieving a global minimum, have been improved but these require intensive computation and storage [3, 4, 5, 20, 21, 24].

Recently, Lyapunov Stability Theory (LST) based algorithms have been proposed to train the parameters of neural networks due to its global minimum solution ability [14, 16, 22]. The LST definition and conditions may easily be found in literature [12]. LST based algorithms are all expressed by utilizing the classical recursive least square method. Unlike gradient descent based techniques, LST based algorithms aim to search for a global minimum through an error energy surface. On the other hand, the determination of adaptation gain rate in these algorithms is a crucial step. Because, the adaptation gain rate directly affects the convergence dynamics. In [14, 16, 22], the adaptation gain rate is determined as a user defined constant parameter after a great number of trials. Hence, an adaptive determination procedure is necessarily required.

In this paper, we have further investigated a LST based training algorithm in an optimization framework. A Lyapunov function candidate, $V_t(k) = a_t^k e_t^2(k)$ with $a_t > 1$, is first defined, and then the procedure is formulated as an inequality constraint optimization problem by using Lagrange multiplier theory similar to in our previous study [18]. Here, $e_k(k)$ and t represent the training error and the MLNN output index ($t = 1, 2, \ldots, m$, m is the number of the outputs), respectively. As a result of optimization problem solution, a new adaptation gain rate, which can adaptively adjust itself in accordance with the sequential square training error rate, is obtained. Thus, the weight coefficients are iteratively updated, from the output layer to the input layer, to make the Lyapunov energy function negative definite at each iteration, $\Delta V(k) = V_t(k) - V_t(k-1) < 0$. Therefore, the training error can asymptotically converge to zero as time goes to infinity. The most attractive feature of the proposed algorithm is the adjustable adaptation gain rate parameter improving the convergence ability and accelerating the MLNN learning process.

The performance of the proposed algorithm is measured by comparing with three existing algorithms. Two of them are backpropagation type algorithms and the other one is a LST based MLNN algorithm. They are tested on three benchmark problems which are XOR, 3-bit parity, and 8-3 encoder. It is found that the proposed algorithm is much faster in convergence than three other algorithms. The proposed algorithm is also comparatively tested on a real MMU Iris Image Database [11] and the effect of adaptation gain rate on the faster convergence ability is also verified for a multiple-input and multiple-output classification problem.

This paper is composed of four sections. In Section 2, the novel algorithm is formulated and the self-adjustable adaptation gain rate is expressed. In Section 3, simulation results are comparatively illustrated and discussed. The conclusion is presented in Section 4.

2. LST based learning algorithm for MLNN

In this section, we have first presented the MLNN system and its linearization process, and then the proposed algorithm is given.

2.1 MLNN system and its linearization

A structure model of an MLNN is shown in Fig. 1. Its characteristic model is a nonlinear input-output mapping from $\Re^n$ to $\Re^m$, where n is the input dimension and m is the output dimension, while u represents the number of the hidden neurons which connects the inputs and outputs [14]. A similar MLNN model for signal tracking problem is proposed in our previous study [17].

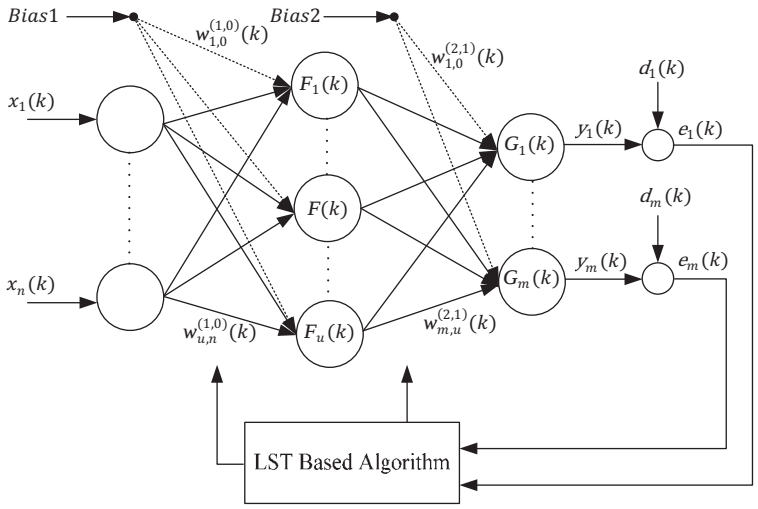


Fig. 1 *A model structure of Multilayer Neural Network [9].*

In a feed forward MLNN, let define an M dimensional weight vector $\boldsymbol{w}_t(k)$ including all the weights of each output node as [14]:

$$\boldsymbol{w}_t(k) = \left[\boldsymbol{w}_t^{(2,1)}(k),\ \boldsymbol{w}_1^{(1,0)}(k), \ldots, \boldsymbol{w}_u^{(1,0)}(k)\right]^{\mathrm{T}} \tag{1}$$

where $M = u(n+1) + (u+1)$, $t = 1, 2, ..., m$. In Eq. (1), upper index (1, 0) is the weight vector from input neurons to hidden neurons, whereas upper index (2, 1) is the weight vector from hidden neurons to output neurons at time step k (see Fig. 1). In Fig. 1, $Bias1$ and $Bias2$ represent unity bias terms for each hidden layer and output layer.

Modelling of a finite-dimensional discrete time system is represented by the following state equations [13, 14]:

$$\boldsymbol{w}_t(k) = \boldsymbol{w}_t(k-1) + v_t(k) \tag{2}$$

$$d_t(k) = f\left(\boldsymbol{w}_t(k-1), \boldsymbol{x}(k)\right) + c_t(k) \tag{3}$$

where $\boldsymbol{w}_t(k)$ is the system state at time k, $\boldsymbol{x}(k) = \{Bias1,\ x_1(k), \ldots, x_n(k)\} \in \Re^n$ is the input data, and $f(\cdot)$: $\Re^n \to \Re^m$ is the nonlinear function of the MLNN. The stochastic processes $\{v_t(k)\}$ and $\{c_t(k)\}$ containing their each covariance matrices are zero mean independent Gaussian random variables [13, 14]. We have assumed

that the unknown state $\boldsymbol{w}_t(k)$ has a Gaussian distribution, and hence in Eq. (2), *a priori* distribution of $\boldsymbol{w}_t(k)$ except desired output $\{d_t(k)\}$ is defined as the state update from previous values of $\boldsymbol{w}_t(k-1)$. However, Eq. (3) shows the system response using the desired output, $\{d_t(k)\}$ [13]. Thus, we can get maximum a posteriori (MAP) estimate $\hat{\boldsymbol{w}}_t(k-1)$ of $\boldsymbol{w}_t(k-1)$ after observing the outputs, from $d_t(1)$ up to $d_t(k)$. So, after getting one more observation $d_t(k+1)$, the updated MAP estimate $\hat{\boldsymbol{w}}_t(k)$ of $\boldsymbol{w}_t(k)$ will be obtained [13].

Similar to the study in [14], we have linearized the MLNN system as a first step for design of weight updating algorithm. Initially, the system state $\boldsymbol{w}_t(k)$ is taken fixed, i.e., $v_t(k)$ is set to zero, and so its covariance matrix is equal to an identity matrix [13]. Then, the Eq. (2) can be given as

$$\boldsymbol{w}_t(k) = \boldsymbol{w}_t(k-1) = \boldsymbol{w}_{\mathrm{o}} \tag{4}$$

where $\boldsymbol{w}_{\mathrm{o}}$ is the true value (optimal value) of the weight vector [13]. Subsequently, the nonlinear function $f(\boldsymbol{w}_t(k-1), \boldsymbol{x}(k))$ is expanded into a Taylor series around the current estimate parameter $\hat{\boldsymbol{w}}_t(k-1)$ to linearize Eq. (3) [1, 6, 7, 13, 14, 25]. After that the desired output is obtained as

$$d_t(k) = f(\hat{\boldsymbol{w}}_t(k-1), \boldsymbol{x}(k)) + \boldsymbol{h}_t^{\mathrm{T}}(k)(\boldsymbol{w}_{\mathrm{o}} - \hat{\boldsymbol{w}}_t(k-1)) + \rho(k) + c_t(k) \tag{5}$$

where $\boldsymbol{h}_t(k)$ is the tth column vector of the Jacobian matrix depend upon the current estimate of $\hat{\boldsymbol{w}}_t(k-1)$ and $\rho(k)$ is higher order terms that may be neglected.

$$\boldsymbol{h}_t(k) = \left.\frac{\partial f(\boldsymbol{w}, \boldsymbol{x}(k))}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\hat{\boldsymbol{w}}_t(k-1)} \tag{6}$$

After linearization, linearized desired output $\tilde{d}_t(k)$ is expressed as

$$\tilde{d}_t(k) = d_t(k) - f(\hat{\boldsymbol{w}}_t(k-1), \boldsymbol{x}(k)) + \boldsymbol{h}_t^{\mathrm{T}}(k)\hat{\boldsymbol{w}}_t(k-1) = \boldsymbol{h}_t^{\mathrm{T}}(k)\boldsymbol{w}_{\mathrm{o}} + c_t(k) \tag{7}$$

Then, $\tilde{d}_t(k)$ is computed for time step k using the weight vector estimate $\hat{\boldsymbol{w}}_t(k-1)$ [6, 7].

Linearized output of the MLNN, $\tilde{y}_t(k)$, and hidden layer outputs, $\boldsymbol{s}(k)$, are as follows [14]

$$\tilde{y}_t(k) = \boldsymbol{h}_t^{\mathrm{T}}(k)\boldsymbol{w}_t(k) \tag{8}$$

$$\boldsymbol{s}(k) = \begin{bmatrix} 1 \\ F_1\left(\boldsymbol{w}_1^{(1,0)}(k)\boldsymbol{x}(k)\right) \\ \vdots \\ F_u\left(\boldsymbol{w}_u^{(1,0)}(k)\boldsymbol{x}(k)\right) \end{bmatrix} \tag{9}$$

After substitution of Eq. (8) and (9) into Eq. (6), we have obtained $\boldsymbol{h}_t(k)$ as follows

$$\boldsymbol{h}_t(k) = \begin{bmatrix} \frac{\partial y_t(k)}{\partial \boldsymbol{w}_t^{(2,1)}(k)} \\ \frac{\partial y_t(k)}{\partial \boldsymbol{w}_1^{(1,0)}(k)} \\ \vdots \\ \frac{\partial y_t(k)}{\partial \boldsymbol{w}_u^{(1,0)}(k)} \end{bmatrix} = \begin{bmatrix} G'(\boldsymbol{s}^{\mathrm{T}}(k)\,\boldsymbol{w}_t^{(2,1)}(k))\boldsymbol{s}(k) \\ G'(\boldsymbol{s}^{\mathrm{T}}(k)\,\boldsymbol{w}_t^{(2,1)}(k))w_{t,1}^{(2,1)}(k)F_1'(\boldsymbol{w}_1^{(1,0)}(k)\,\boldsymbol{x}(k))\boldsymbol{x}(k) \\ \vdots \\ G'(\boldsymbol{s}^{\mathrm{T}}(k)\,\boldsymbol{w}_t^{(2,1)}(k))w_{t,u}^{(2,1)}(k)F_u'(\boldsymbol{w}_u^{(1,0)}(k)\,\boldsymbol{x}(k))\boldsymbol{x}(k) \end{bmatrix} \quad (10)$$

where

$$G'(z_r) = 1 - (z_r)^2,\ r = 1, \ldots, m$$
$$F'(z_l) = 1 - (z_l)^2,\ l = 1, \ldots, u$$

Here, z_r and z_l represent the summation output of each neurons. Here, $F(\cdot)$ and $G(\cdot)$ are determined as tangent sigmoid nonlinear functions. Also, $F'(\cdot)$ and $G'(\cdot)$ represent derivatives. Thus, the linearization procedure is completed. The LST based algorithm design is presented in the following subsection.

2.2 The proposed LST based algorithm

After the linearization of MLNN system, here, let express a novel adaptive learning process in an optimization framework by using LST with Lagrange multiplier theory. A Lyapunov function, $V_t(k) = a_t^k e_t^2(k)$ with $a_t > 1$, is first determined according to Lyapunov stability theory for the "t"th output nodes of the MLNN. Then, optimal $\boldsymbol{w}_t^*$ values are found by minimizing the cost function Eq. (11):

$$\Phi(\boldsymbol{w}_t) = \frac{1}{2}\delta\boldsymbol{w}_t^{\mathrm{T}}\delta\boldsymbol{w}_t \quad (11)$$

subject to inequality constraint, $\Delta V(k) = V_t(k) - V_t(k-1) < 0$, satisfying negative definiteness of Lyapunov function, $V_t(k) = a_t^k e_t^2(k)$ with $a_t > 1$, which contributes to the asymptotic stability of MLNN algorithm in the sense of Lyapunov.

$$\delta\boldsymbol{w}_t = \boldsymbol{w}_t(k) - \boldsymbol{w}_t(k-1) \quad (12)$$

The proposed inequality constrained optimization problem is solved by using the method of Lagrange multipliers [2]. Similar to our previous study in [18], the minimization problem is presented as follows

$$\boldsymbol{w}_t^* = \arg\min\left(\frac{1}{2}\delta\boldsymbol{w}_t^{\mathrm{T}}\delta\boldsymbol{w}_t\right)$$

subject to $\left(a_t^k e_t^2(k) - a_t^{k-1} e_t^2(k-1)\right) < 0$, $\forall k$ and $a_t \in \Re^+$. Here, $\boldsymbol{w}_t^*$ represents the optimal weight vector. In Eq. (13), the Lagrangian function is written in accordance with Lagrange method.

$$
\begin{aligned}
F(\boldsymbol{w}_t\mu) &= \frac{1}{2}\left(\boldsymbol{w}_t\left(k\right)-\boldsymbol{w}_t\left(k-1\right)\right)^{\mathrm{T}}\left(\boldsymbol{w}_t\left(k\right)-\boldsymbol{w}_t\left(k-1\right)\right) \\
&\quad +\mu\left(a_t^k\left(d_t\left(k\right)-\boldsymbol{h}_t^{\mathrm{T}}(k)\boldsymbol{w}_t\left(k\right)\right)^2-a_t^{k-1}\left(d_t\left(k-1\right)\right.\right. \\
&\quad \left.\left.-\boldsymbol{h}_t^{\mathrm{T}}(k-1)\boldsymbol{w}_t\left(k-1\right)\right)^2\right) \qquad (13)
\end{aligned}
$$

where μ is the nonnegative Lagrange multiplier variable. The solution of the inequality constrained optimization problem is determined by minimizing Lagrangian function $F(\boldsymbol{w}_t,\mu)$ with respect to $\boldsymbol{w}_t$ and μ, respectively [17]. After differentiation of $F(\boldsymbol{w}_t,\mu)$ with respect to $\boldsymbol{w}_t$ and μ, then setting results equal to zero, we obtain the following two conditions of optimality

$$
\text{Condition}\,1:\frac{\partial F(\boldsymbol{w}_t,\mu)}{\partial \boldsymbol{w}_t}=0 \qquad (14)
$$

$$
\text{Condition}\,2:\frac{\partial F(\boldsymbol{w}_t,\mu)}{\partial \mu}=0 \qquad (15)
$$

Application of optimality *Condition 1* to the Lagrange function in Eq. (13), we obtain the following updating formula after rearrangement of terms

$$
\boldsymbol{w}_t\left(k\right)=\boldsymbol{w}_t\left(k-1\right)+\frac{\boldsymbol{h}_t\left(k\right)}{\left\|\boldsymbol{h}_t\left(k\right)\right\|^2}\left(1-a_t^{-\frac{k}{2}}\frac{\left|e_t\left(k-1\right)\right|}{\left|\alpha_t\left(k\right)\right|}\right)\alpha_t\left(k\right) \qquad (16)
$$

where $\alpha_t\left(k\right)$ is the a priori estimation error

$$
\alpha_t\left(k\right)=d_t\left(k\right)-\boldsymbol{h}_t^{\mathrm{T}}(k)\boldsymbol{w}_t\left(k-1\right) \qquad (17)
$$

Adaptation gain function, $\boldsymbol{g}_t\left(k\right)$, is also defined as

$$
\boldsymbol{g}_t\left(k\right)=\frac{\boldsymbol{h}_t\left(k\right)}{\left\|\boldsymbol{h}_t\left(k\right)\right\|^2}\left(1-a_t^{-\frac{k}{2}}\frac{\left|e_t\left(k-1\right)\right|}{\left|\alpha_t\left(k\right)\right|}\right) \qquad (18)
$$

Weight vector update law in Eq. (16) can then simply be rewritten using $\boldsymbol{g}_t\left(k\right)$ and $\alpha_t\left(k\right)$ terms as below

$$
\boldsymbol{w}_t\left(k\right)=\boldsymbol{w}_t\left(k-1\right)+\boldsymbol{g}_t\left(k\right)\alpha_t\left(k\right) \qquad (19)
$$

The inequality constraint $\Delta V\left(k\right)$ is represented in terms of the adaptation gain rate "a_t" and the training error "$e_t(k)$" in order to clearly recognize the negative definiteness of the Lyapunov function, as follows

$$
\begin{aligned}
\Delta V\left(k\right) &= V_t\left(k\right)-V_t\left(k-1\right) \\
&= a_t^k e_t^2\left(k\right)-a_t^{k-1}e_t^2\left(k-1\right) \\
&\vdots \\
&= a_t^k[\alpha_t\left(k\right)-\alpha_t\left(k\right)\boldsymbol{g}_t^{\mathrm{T}}\left(k\right)\boldsymbol{h}_t\left(k\right)]^2-a_t^{k-1}e_t^2\left(k-1\right) \qquad (20)
\end{aligned}
$$

Using the adaptation gain function Eq. (18) and could be rewritten Eq. (20) as follows

$$\begin{aligned}
\Delta V(k) &= a_t^k \left[\alpha_t(k) - \alpha_t(k) \left(\frac{\boldsymbol{h}_t(k)}{\|\boldsymbol{h}_t(k)\|^2} \left(1 - a_t^{-\frac{k}{2}} \frac{|e_t(k-1)|}{|\alpha_t(k)|} \right) \right)^{\mathrm{T}} \boldsymbol{h}_t(k) \right]^2 \\
&\quad - a_t^{k-1} e_t^2(k-1) \\
&\vdots \\
&= e_t^2(k-1) - a_t^{k-1} e_t^2(k-1) \\
&= e_t^2(k-1)\left(1 - a_t^{k-1}\right) < 0, \text{ for all } k > 1 \text{ and } a_t > 1. \qquad (21)
\end{aligned}$$

As a result of implementing the optimality *Condition 2* in Eq. (15), we get the adaptation gain rate a_t as

$$a_t = \frac{e_t^2(k-1)}{e_t^2(k)} \qquad (22)$$

a_t in Eq. (22), represents the adaptation gain rate at current step, k. Therefore, we have rewritten the Eq. (22) as in Eq. (23) in an appropriate iterative form such as $a_t(k)$ for using in computational algorithm.

$$a_t(k) = \frac{e_t^2(k-1)}{e_t^2(k)} \qquad (23)$$

As seen from Eq. (21), the computational algorithm should be started with an initial "$a_t(0)$" value which is greater than 1 (one) for satisfying asymptotic stability in the sense of Lyapunov. Therefore, Eq. (23) is reformulated by just adding "1" to the right side of it satisfying $a_t(k) > 1$. Thus, the error $e_t(k)$ can asymptotically converge to zero with the last form of the adaptation gain rate given in Eq. (24).

$$a_t(k) = 1 + \left(\frac{e_t^2(k-1)}{e_t^2(k)} \right) \qquad (24)$$

Consequently, the "$a_t(k)$" in Eq. (24) can be used in the proposed computational algorithm by choosing any arbitrary positive initial value providing, $a_t(0) > 1$.

To prevent singularities in proposed computational algorithm, we have also added sufficiently small positive slack variables to adaptation gain function Eq. (18) as follows:

$$\boldsymbol{g}_t(k) = \frac{\boldsymbol{h}_t(k)}{\lambda + \|\boldsymbol{h}_t(k)\|^2} \left(1 - \frac{|e_t(k-1)|}{\lambda + a_t^{\frac{k}{2}}(k-1)\,|\alpha_t(k)|} \right) \qquad (25)$$

where $\lambda \ll 1$ and, $\lambda \in \Re^+$. Here, the argument of adaptation gain rate is taken as $(k-1)$ in Eq. (25) for initializing of algorithm in accordance with $e_t(k-1)$. This condition can easily be checked and verified by following the flow diagram given in Fig. 2.

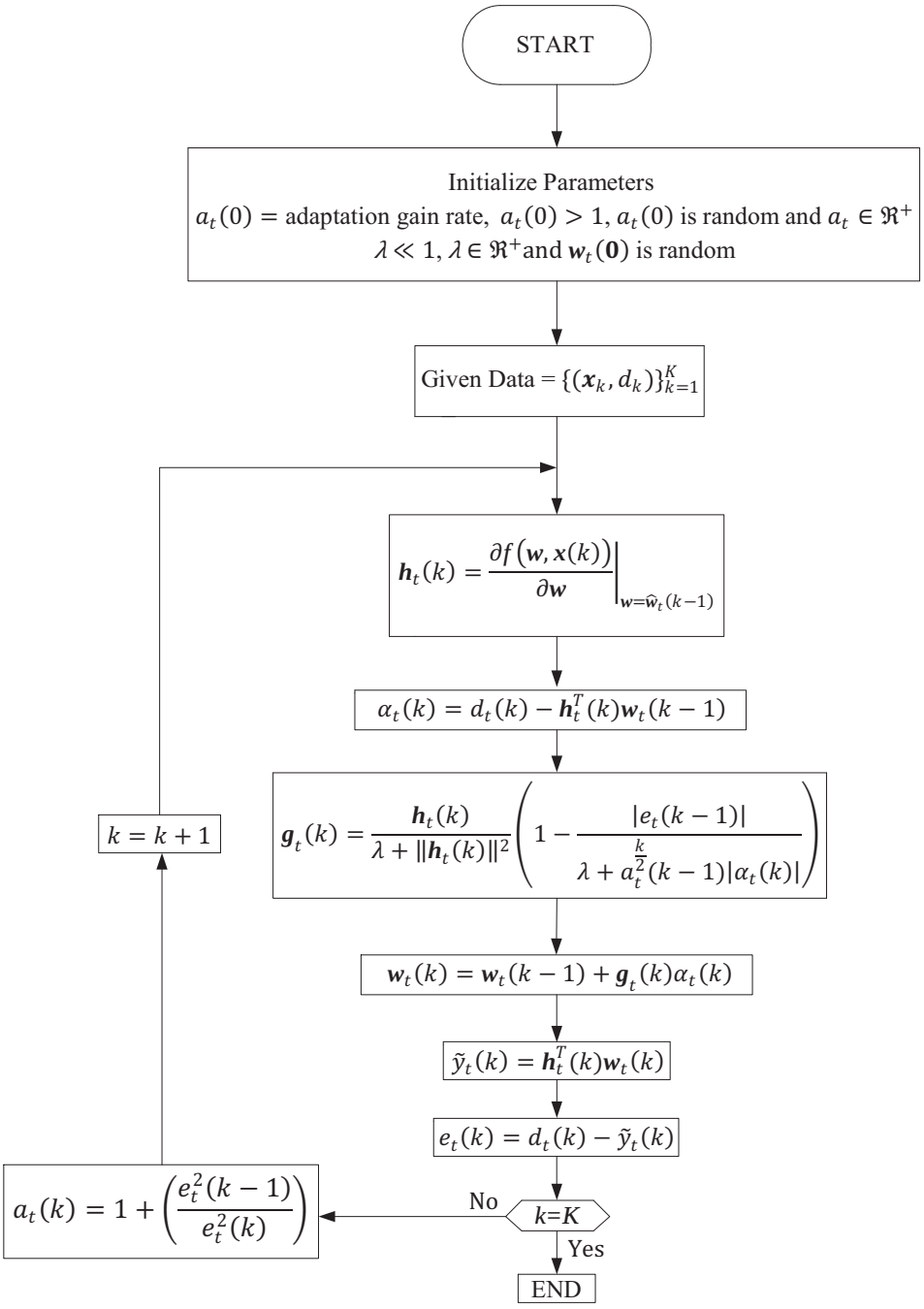


Fig. 2 *Flow diagram of the proposed LST based MLNN algorithm.*

3. Simulation results and discussion

In this study, a three-layered neural network is used for evaluating the performance of the proposed LST based MLNN learning algorithm. Unity bias terms are applied to each neuron. The proposed algorithm is performed on three benchmark problems such as XOR, 3-bit parity and 8-3 encoder. Additionally, the proposed algorithm is tested on a real iris recognition database to verify the effect of proposed adaptation gain rate for multiple-input and multiple-output classification problem. Tangent sigmoid function is selected as an activation function for each problem. The proposed algorithm is compared with two types of conventional BP algorithms and a LST based MLNN algorithm which has fixed adaptation gain rate parameter [14]. For benchmark problems, the patterns are presented to the network sequentially during the training procedure. The training process for each benchmark problems is finished when mean-square error (MSE) per epoch reaches at 10^{-5}. Due to the weight updating algorithm performed with small random initial different values, convergence time varies from run to run in an acceptable accuracy interval. Therefore, the convergence time for each algorithm is calculated by averaging fifty different runs for a better comparison. Each run means that the network is trained with any arbitrary random weight vector. In our simulation examples, we used gradient descent BP algorithm with adaptive learning rate which is taken to be $\eta = 0.5$ and $\eta = 0.95$, respectively but BP algorithm with its fixed learning rate which is taken to be only $\eta = 0.95$. For LST based MLNN algorithm [14], its fixed adaptation gain rate parameter is taken to be $a_t = 1.01$. In our proposed algorithm, initial value of adaptation gain rate, $a_t(0)$, is also chosen as $a_t(0) = 1.01$ which is the same with other LST based algorithm's parameter for an objective evaluation. The initial posterior error $e_t(0)$ is set to be 0.01. All algorithms used in this study have been performed on a personal computer with the speed of 2.53 GHz by using MATLAB R2009a software. The algorithm codes can also be obtained from correspondence author via e-mail.

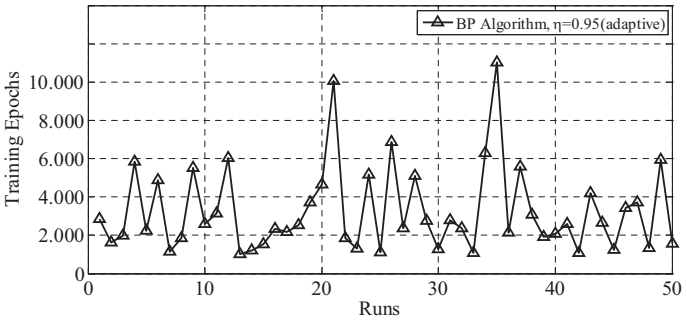
3.1 XOR problem

This application is indeed a basic classification problem. For the XOR problem, the input series are presented as $[00; 01; 10; 11]$ and output series are taken as $[0; 1; 1; 0]$. A network which has two inputs, one output and four neurons in the hidden layer is constructed for the simulation. Training set includes four patterns in one epoch.

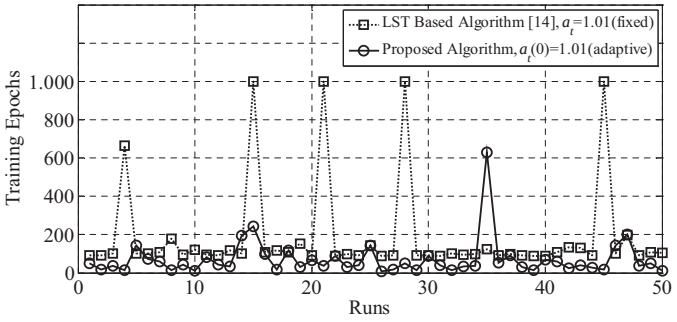
Algorithm	Epochs	Training Time (sec)	Computational Time For One Epoch (sec)	Parameters
BP	3134	32	0.01039	$\eta = 0.5$ (Adaptive)
BP	2569	26	0.01050	$\eta = 0.95$ (Adaptive)
LST based MLNN [14]	189	1.232	0.00652	$a_t = 1.01$ (Fixed)
Proposed	71	0.537	0.00757	$a_t(0) = 1.01$ (Adaptive)

Tab. I *Comparison of the algorithms for XOR problem.*

The simulation results for XOR problem are presented in Tab. I. It can be seen that the proposed algorithm with its adjustable adaptation gain rate, $a_t(k)$, takes minimum number of epochs to converge predefined MSE value (10^{-5}) as compared to BP algorithms and LST based MLNN [14] algorithm. In this case, the proposed algorithm is about 2.5 times faster than LST based MLNN [14] in terms of number of epochs. Moreover, it can also be seen that the proposed algorithm is nearly two times faster than the same LST based algorithm in terms of training time. BP with its fixed adaptive learning rate $\eta = 0.95$ does not converge to the MSE of 10^{-5} within 20 000 epochs, therefore, it is excluded from comparison process.



(a)



(b)

Fig. 3 *Convergence time (in terms of iteration) comparison among three algorithms for XOR problem. a) BP algorithm ($\eta = 0.95$ (Adaptive)), b) LST based MLNN algorithm ($a_t = 1.01$ (Fixed)) and Proposed algorithm ($a_t(0) = 1.01$ (Adaptive)).*

Fig. 3(a) and (b) shows the performance of three algorithms performed with different initial conditions. It is clear from the figures that the number of epochs for convergence fluctuates very much for BP and LST based MLNN algorithms [14]. This fluctuation is reduced in case of proposed algorithm and it behaves stable. The proposed algorithm with the obtained adjustable adaptation gain rate, $a_t(k)$, provides a tangible improvement over LST based MLNN algorithm [14] both in terms of training time and training epochs (See Fig. 3(b)).

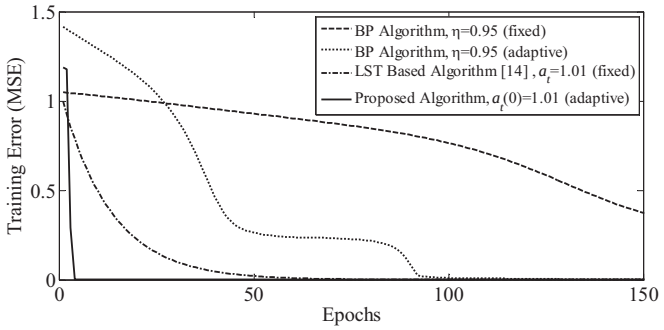


Fig. 4 *Comparison of training error in terms of MSE among three algorithms for XOR problem.*

The improvement with the proposed algorithm in error convergence can be clearly seen in Fig. 4. It is observed that as the number of epochs increases the proposed algorithm with its adjustable adaptation gain rate, $a_t(k)$, gives a better accuracy as compared to other algorithms. The adaptation gain rate parameter, $a_t(k)$, adaptively adjusts itself depending on a sequential square error rate for a better convergence at each iteration. Thus, the training error of the proposed algorithm asymptotically converges to zero as time goes to infinity and this situation can easily be observed in Fig. 4.

3.2 3-bit parity problem

This application is chosen as another classification problem. For the 3-bit parity problem, the input series are presented as $[000; 001; 010; 011; 100; 101; 110; 111]$ and output series are taken as $[0; 1; 1; 0; 1; 0; 0; 1]$. A three-layered network which has three inputs, one output and seven neurons in the hidden layer is constructed for simulation. There are eight patterns in one epoch for this problem.

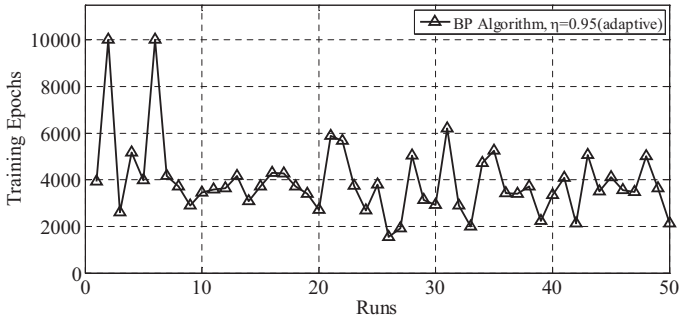
Algorithm	Epochs	Training Time (sec)	Computational Time For One Epoch (sec)	Parameters
BP	4227	43.48	0.01022	$\eta = 0.5$ (Adaptive)
BP	3729	38.14	0.01023	$\eta = 0.95$ (Adaptive)
LST based MLNN [14]	1449	12.6	0.00812	$a_t = 1.01$ (Fixed)
Proposed	1013	8.33	0.00822	$a_t(0) = 1.01$ (Adaptive)

Tab. II *Comparison of the algorithms for 3-bit parity problem.*

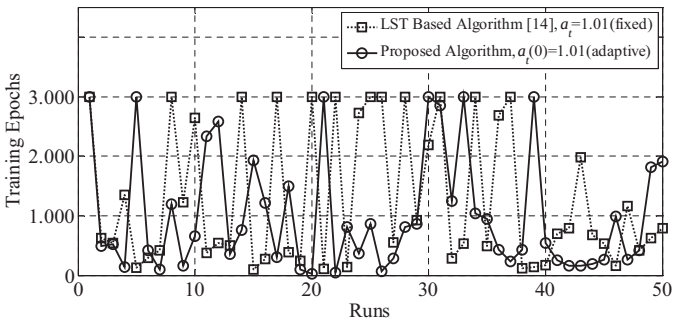
Tab. II shows the simulation results for 3-bit parity problem. As seen from Tab. II, the proposed algorithm outperforms both BP and LST based MLNN algorithm [14] in terms of training time as well as number of training epochs. In this

example, the proposed algorithm is about 1.5 times faster than LST based MLNN algorithm in terms of training epoch numbers. BP with its fixed adaptive learning rate $\eta = 0.95$ is also performed for 3-bit parity problem but this algorithm does not converge to the MSE of 10^{-5} within 20 000 epochs and thus, it is excluded from Tab. II.

From the Fig. 5 (a) and (b), it can clearly be observed that the proposed algorithm performs better than LST based MLNN algorithm [14] and BP type algorithms both in terms of training time and training epochs.



(a)



(b)

Fig. 5 *Convergence time (in terms of iteration) comparison among three algorithms for 3-bit parity problem. (a) BP algorithm ($\eta = 0.95$ (Adaptive)), (b) LST based MLNN algorithm ($a_t = 1.01$ (Fixed)) and Proposed algorithm ($a_t(0) = 1.01$ (Adaptive)).*

The MSE convergence improvement is shown in Fig. 6. It can be observed that as the number of epochs increases the proposed algorithm with its adjustable adaptation gain rate, $a_t(k)$, gives better accuracy as compared to other algorithms.

3.3 8-3 encoder problem

This experiment is chosen to solve the multiple-input multiple-output (MIMO) problem. For the 8-3 encoder problem, the input series are presented as [10000000; 01000000; 00100000; 00010000; 00001000; 00000100; 00000010; 00000001] and output

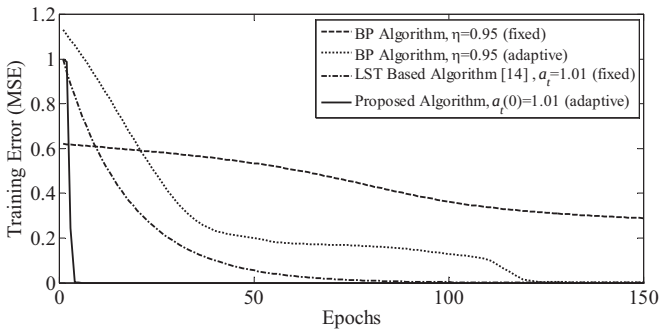


Fig. 6 *Comparison of training error in terms of MSE among three algorithms for 3-bit parity problem.*

series are taken as $[000; 001; 010; 011; 100; 101; 110; 111]$. The network architecture is composed of eight inputs, three outputs, and 16 hidden neurons.

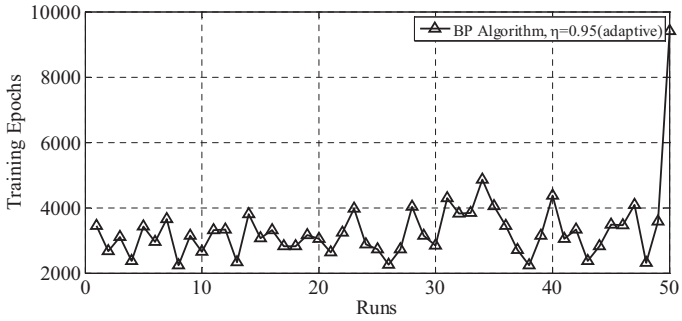
Tab. III shows the simulation results of 8-3 encoder problem. There are eight patterns in one epoch for this experiment. As shown in Tab. III, the proposed algorithm with its adjustable adaptation gain rate, $a_t(k)$, outperforms other algorithms. The proposed algorithm is nearly 1.5 times faster than LST based MLNN algorithm [14] both in terms of convergence time and training epochs for MIMO problem. BP algorithm with its fixed learning rate does not converge to MSE of 10^{-5} for this problem and that is why it is excluded from Tab. III.

Algorithm	Epochs	Training Time (sec)	Computational Time For One Epoch (sec)	Parameters
BP	3622	29.9	0.01029	$\eta = 0.5$ (Adaptive)
BP	3352	29.1	0.01023	$\eta = 0.95$ (Adaptive)
LST based MLNN [14]	84	2.1	0.02521	$a_t = 1.01$ (Fixed)
Proposed	57	1.5	0.02632	$a_t(0) = 1.01$ (Adaptive)

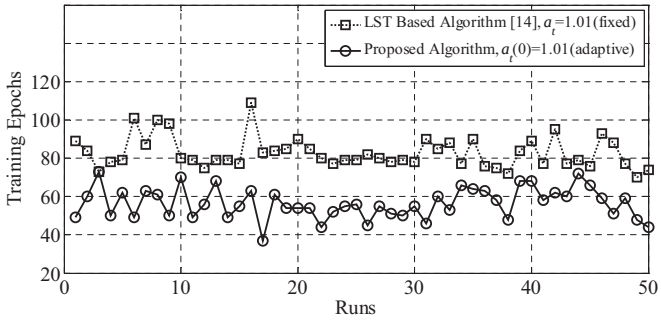
Tab. III *Comparison of the algorithm for 8-3-encoder problem.*

As shown in Fig. 7, the proposed algorithm performs better than all other algorithms both in terms of training time and training epochs for MIMO problem. The proposed algorithm also reduced the epoch fluctuations with respect to run. It oscillates (performs stable) between 40 and 70 epochs (Fig. 7 (b)). Reduction in convergence time and training epochs in case of the proposed algorithm provides tangible improvement over other algorithms as seen from Fig. 7 (a) and (b).

Similar to above applications, it can be observed that as the number of training epochs increases the proposed algorithm with its adjustable adaptation gain rate, $a_t(k)$, gives better accuracy as compared with other algorithms for 8-3 en-



(a)



(b)

Fig. 7 *Convergence time (in terms of iteration) comparison among three algorithms for 8-3 encoder problem. (a) BP algorithm ($\eta = 0.95$ (Adaptive)), (b) LST based MLNN algorithm ($a_t = 1.01$ (Fixed)) and Proposed algorithm ($a_t(k) = 1.01$ (Adaptive)).*

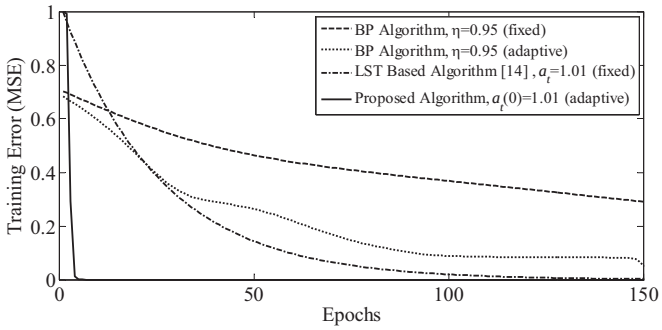


Fig. 8 *Comparison of training error in terms of MSE among three algorithms for 8-3 encoder problem.*

coder problem in Fig. 8. By the virtue of adjustable adaptation gain rate, $a_t(k)$, the convergence rate of the proposed algorithm is accelerated efficiently. Thus, the proposed algorithm provides a faster convergence for all cases at given above benchmark examples.

3.4 IRIS recognition problem

Iris recognition is considered as a real world MIMO classification problem for evaluating the performance of the proposed algorithm. In experiments, the Multimedia University (MMU) database of iris images are used [11]. MMU iris database contributes a total number of 450 iris images. There are five iris images for each eye in this database. We used only right eyes' data in the experiments (Totally, 135 training images and 90 testing images obtained from 45 different persons). With 5 iris images available for right eyes, the algorithm has been performed randomly using three images of each person for training, and the remaining two images for testing the network.

Due to the high dimensional input data, a feature extraction technique which is PCA (Principal Component Analysis) [23] is firstly used to reduce the iris image's dimension for the experiments. Then, these features are fed to the network. Feature vector size consists of 60 PCA features, which was determined after many trials involving the adjustment of size from 10 to 100.

In the experiments, the size of network input is set to the number of 60 (number of PCA features) and the number of output neurons is set to 45 (number of classes to be recognized). The number of hidden layer neurons is fixed to be 60. Each experimental result is obtained after five simulation runs. For performance evaluation, an average recognition rate [1] is defined as R_{ave};

$$R_{\text{ave}} = \frac{\sum_{i=1}^{q} n_c^i}{q n_{\text{tot}}} \tag{26}$$

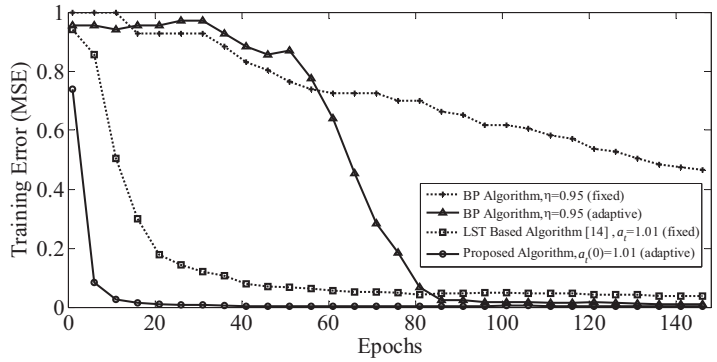
where q is the number of simulation runs, n_c^i is the number of correct recognitions for *i-th* run, and n_{tot} is the total number of the testing data for each run.

As shown in Fig. 9(a), the proposed training algorithm has been reached to the expected minimum MSE point within about 50 epochs. It is confirmed that the proposed MLNN training algorithm has faster error convergence ability than other algorithms. The training error exponentially converges to zero as the training epochs increase.

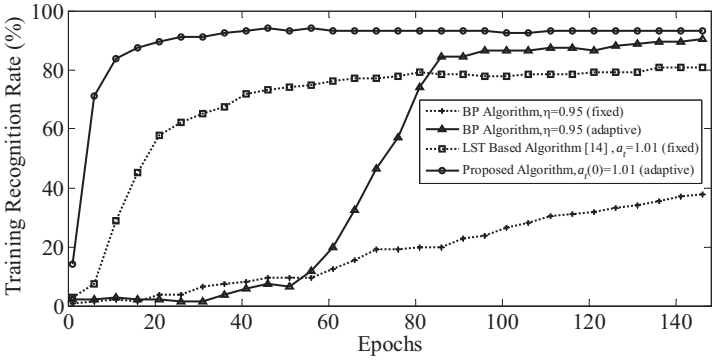
Fig. 9(b) shows the training recognition rate plotted versus the training epochs. The proposed MLNN classifier algorithm can achieve 94% of training recognition with a faster rate at about 50 epochs, as compared with other algorithms. Therefore, the proposed LST based recognition system provides a rapid training process. In the meantime, the proposed algorithm has also been performed for testing images. As a result, the proposed algorithm achieves a higher and faster testing recognition rate at about 83% within 50 epochs (Fig. 9(c)). As a result, it is verified that the proposed algorithm can be used as a perfect classifier in real world problem solutions.

4. Conclusion

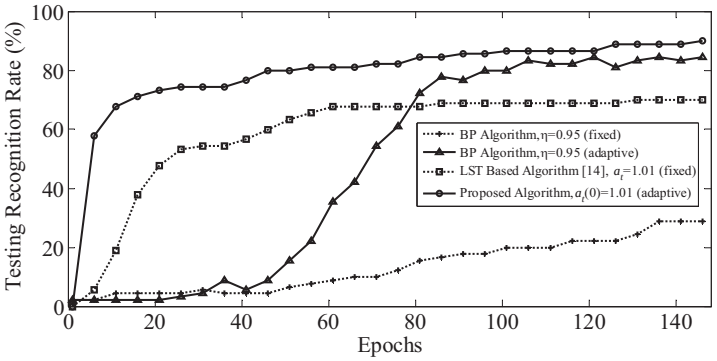
In this study, a novel LST based algorithm for training the feedforward MLNN is presented. The convergence capability of the LST based algorithm is improved by using a new adaptation gain rate "$a_t(k)$" which has the ability to adaptively adjust



(a)



(b)



(c)

Fig. 9 *(a) Training error versus epochs, (b) Training recognition rate versus epochs, (c) Testing recognition rate versus epochs.*

itself depending on a sequential square error rate. Experimental results show that the proposed algorithm is faster than both popular two BP algorithms and a LST based MLNN algorithm in terms of convergence time as well as training epochs.

The proposed algorithm is also comparatively tested on the Multimedia University (MMU) real Iris Image Database for MIMO classification problem and the effect of adaptive gain rate for faster convergence and higher performance is verified.

Acknowledgments

This study has partially been supported by Nigde University, The Scientific Research Project Unit with the project number of FEB-2010/32. We also would like to thank to the authorities of Multimedia University MMU1 Iris Image Database.

References

[1] ANG L. et al. A lyapunov theory-based neural network approach for face recognition. In: Chiong R., ed. *Intelligent Systems for Automated Learning and Adaptation Emerging Trends and Applications*, Hershey: IGI Global, 2010, pp. 23–48, doi: 10.4018/978-1-60566-798-0.ch002.

[2] BERTSEKAS D.P. *Nonlinear Programming*. MA: Belmont, Athenas Scientific, 1995.

[3] BILSKI J., RUTKOWSKI L. A fast training algorithm for neural networks. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*. 1998, 45(6), pp. 749–753, doi: 10.1109/82.686696.

[4] CHANSARKAR M.M., DESAI U.B. A robust recursive least squares algorithm. *IEEE Transactions on Signal Processing*. 1997, 45(7), pp. 1726–1735, doi: 10.1109/78.599942.

[5] CHARALAMBOUS C. Conjugate gradient algorithm for efficient training of artificial neural networks. *Circuits, Devices and Systems, IEE Proceedings G*. 1992, 139(3), pp. 301–310.

[6] DOUGLAS S.C., MENG T.H.Y. Linearized least-squares training of multilayer feedforward neural networks. In: *International Joint Conference on Neural Networks (IJCNN 1991)*, Seattle, USA. IEEE Xplore, 1991, pp. 307–312, doi: 10.1109/IJCNN.1991.155195

[7] GUPTA M.M., JIN L., HOMMA N. *Static and Dynamic Neural Networks*. NY: Wiley-Interscience, 2003.

[8] HAYKIN S. *Adaptive Filter Theory*. USA, NJ: Prentice Hall, 1996.

[9] HAYKIN S. *Neural Networks: A Comprehensive Foundation*. NY: Macmillan, 1999.

[10] HERTZ J., KROUGH A., PALMER R.G. *Introduction to the Theory of Neural Computation*. CA: Addison-Wesley, 1991.

[11] IRIS IMAGE DATABASE. Multimedia University (MMU) Iris Image Database [online]. 2004-12-03 [viewed 2014-01-17]. Available from: http://www.nigde.edu.tr/elektrikelektronikmuhendisligi/akademik.php?id=500&sec=5

[12] KHALIL H.K. *Nonlinear Systems*. NY: Macmillan, 1992.

[13] LEUNG C.S., TSOI A.C., CHAN L.W. Two regularizers for recursive least squared algorithms in feedforward multilayered neural networks. *IEEE Transactions on Neural Networks*. 2001, 12(6), pp. 1314–1332, doi: 10.1109/72.963768.

[14] LIM K.H., et al. Lyapunov theory-based multilayered neural network. *IEEE Transactions on Circuits and Systems II: Express Briefs*. 2009, 56(4), pp. 305–309, doi: 10.1109/TCSII.2009.2015400.

[15] LIPPMANN R.P. An introduction to computing with neural nets. *IEEE ASSP Magazine*. 1987, 4(2), pp. 4–22., doi: 10.1109/MASSP.1987.1165576.

[16] MAN Z., PHOOI S.K., WU H.R. Lyapunov stability-based adaptive backpropagation for discrete time system. In: *Proceedings of the Fifth International Symposium on Signal Processing and Its Applications (ISSPA '99)*, Brisbane, Australia. IEEE Xplore, 1999, pp. 661–664, doi: 10.1109/ISSPA.1999.815759.

[17] MENGÜÇ E.C., ACIR N. A novel nonlinear adaptive filter design and its implementation with FPGA. In: *20th IEEE Signal Processing and Communications Applications (SIU'12)*, Mugla, Turkey. IEEE Xplore, 2012, pp. 1–4, doi: 10.1109/SIU.2012.6204472.

[18] MENGÜÇ E.C., ACIR N. A novel adaptive filter design using Lyapunov stability theory [online manuscript accepted for publication]. *Turkish Journal of Electrical Engineering & Computer Sciences.* [viewed 2014-12-02]. Available from: http://mistug.tubitak.gov.tr/bdyim/kabul.php?dergi=elk

[19] NARENDRA K.S., PARTHASARATHY K. Gradient methods for optimisation of dynamical systems containing neural networks. *IEEE Transactions on Neural Networks*. 1991, 2(2), pp.252–262, doi: 10.1109/72.80336.

[20] OSOWSKI S., BOJARCZAK P., STODOLSKI M. Fast second order learning algorithm for feedforward multilayer neural networks and its applications. *Neural Networks*. 1996, 9(9), pp. 1583–1596, doi: 10.1016/S0893-6080(96)00029-9.

[21] SARKAR D. Methods to speed up error back propagation learning algorithm. *ACM Comput. Surv.* 1995, 27(4), 519–544, doi: 10.1145/234782.234785.

[22] SENG K.P., MAN Z., WU H.R. Lyapunov theory-based radial basis function networks for adaptive filtering. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*. 2002, 49(8), pp. 1215–1220, doi: 10.1109/TCSI.2002.801255.

[23] STAN Z.L., ANIL K.J. *Handbook of face recognition*. NY: Springer-Verlag, 2004.

[24] VERHAEGEN M. Round-off error propagation in four generally applicable, recursive, least squares estimation schemes. *Automatica*. 1989, 25(3), pp. 437–444, doi: 10.1016/0005-1098(89)90013-7.

[25] XU Y., WONG K., LEUNG C. Generalized RLS approach to the training of neural networks. *IEEE Transactions on Neural Networks*. 2006, 17(1), pp. 19–34, doi: 10.1109/TNN.2005.860857.