On Adaptive Learning Rate That Guarantees Convergence in Feedforward Networks

Laxmidhar Behera, *Senior Member, IEEE*, Swagat Kumar, and Awhan Patnaik

***Abstract*—This paper investigates new learning algorithms (LF I and LF II) based on Lyapunov function for the training of feedforward neural networks. It is observed that such algorithms have interesting parallel with the popular backpropagation (BP) algorithm where the fixed learning rate is replaced by an adaptive learning rate computed using convergence theorem based on Lyapunov stability theory. LF II, a modified version of LF I, has been introduced with an aim to avoid local minima. This modification also helps in improving the convergence speed in some cases. Conditions for achieving global minimum for these kind of algorithms have been studied in detail. The performances of the proposed algorithms are compared with BP algorithm and extended Kalman filtering (EKF) on three bench-mark function approximation problems: XOR, 3-bit parity, and 8-3 encoder. The comparisons are made in terms of number of learning iterations and computational time required for convergence. It is found that the proposed algorithms (LF I and II) are much faster in convergence than other two algorithms to attain same accuracy. Finally, the comparison is made on a complex two-dimensional (2-D) Gabor function and effect of adaptive learning rate for faster convergence is verified. In a nutshell, the investigations made in this paper help us better understand the learning procedure of feedforward neural networks in terms of adaptive learning rate, convergence speed, and local minima.**

***Index Terms*—Adaptive learning rate, backpropagation (BP), extended Kalman filtering (EKF), feedforward networks, Lyapunov function, Lyapunov stability theory, system-identification.**

I. Introduction

THIS paper is concerned with the problem of training a multilayered feedforward neural network. Faster convergence and function approximation accuracy are two key issues in choosing a training algorithm. The popular method for training a feedforward network has been the backpropagation (BP) algorithm [1], [2]. The neural network literature is inundated with papers that focus on its application to various problems and its real-time implementation. However, there are very few papers which address the issue of convergence in BP networks [3]–[5]. One of the main drawbacks of BP algorithm is its slow rate of convergence and its inability to ensure global convergence. Some heuristic methods like adding a momentum term to original BP algorithm and standard numerical optimization techniques using quasi-Newton methods have been proposed to improve the convergence rate of BP algorithm [6], [7]. The problems with quasi-Newton methods are that the storage and memory requirements go up as the square of the size of the network. The nonlinear optimization technique such as the Newton method, conjugate-gradient, etc., [8], [9] have been used for training. Though the algorithm converges in fewer iterations than BP algorithm, it requires too much computation per pattern. Other algorithms for faster convergence include extended Kalman filtering (EKF) [10], recursive least square (RLS) [11] and Levenberg–Marquardt (LM) [12], [13]. In order to overcome the computational complexity in these algorithms, a number of improvements have also been suggested [14], [15]. However, these improvements do not bring them closer to BP algorithm as far as simplicity and ease of implementation is concerned.

Use of Lyapunov stability theory in control problems is very well known. Its use in deriving training algorithms for neural networks (NN) has been quite recent. Yu *et al.* [5] have derived a generalized weight update law using a Lyapunov function that guarantees global convergence. However, the update law is of theoretical interest since its implementation will be very much computationally intensive due to the presence of Hessian terms. However, authors have shown that other learning algorithms such as BP, Gauss–Newton, LM, etc., are special cases of this general weight update law. In another work, Yu *et al.* [16] have used Lyapunov stability theory to derive a stable learning law for multilayer dynamic neural network. They showed that their learning algorithm is similar to BP algorithm for multilayered perceptron (MLP) with an additional term which ensures the stability of the identification error. In all these works, Lyapunov stability theory has been used to devise learning algorithms to adapt the weights of the network so as to minimize certain cost criteria.

In this paper, we have proposed two novel algorithms (LF I and LF II) using Lyapunov stability theory. Interestingly, the proposed algorithm has exact parallel with the popular BP algorithm where the fixed learning rate in BP algorithm is replaced by an adaptive learning rate in the proposed algorithm. Earlier, network inversion algorithm using Lyapunov function approach has been studied [17]. However, detailed study of this algorithm for neural network weight update has not been done. It is shown that LF I becomes globally convergent if local minima are avoided along the convergence trajectory. But this is not possible without adding a second-order term in the weight update algorithm as shown by Yu *et al.* [5]. But the addition of a second-order term gives rise to computational difficulties as discussed earlier in this section. In the modified version LF II, we show that it is possible to avoid local minima to some extent. It is well known that in gradient–descent, the direction of weight update reverses at the local minima and that is why BP

Manuscript received September 19, 2004; revised December 27, 2005. This work was supported by DST under the Project DST/EE/20050331 "Intelligent control schemes and application to dynamic and visual control of redundant manipulator systems."

The authors are with the Department of Electrical Engineering, Indian Institute of Technology, Kanpur 208 016, India (e-mail: lbehera@iitk.ac.in).

Digital Object Identifier 10.1109/TNN.2006.878121

algorithm is prone to getting stuck at such points. It is shown here that by controlling the rate of weight update, it is possible to drive out of a local minimum by opposing the change in the direction of weight update. Moreover, the choice of network architecture plays a crucial role in ensuring global convergence of a learning algorithm. Through proposed algorithms, the conditions on network architecture for avoiding local minima have been studied in detail. Thus, the paper presents a way to improve the convergence properties of conventional algorithms without resorting to heuristics.

There are many heuristic approaches to decide on adaptive learning rate as reported in [18]–[21]. However, this is the first time an adaptive learning rate for BP network has been derived with accelerated convergence. It is observed that this adaptive learning rate increases the speed of convergence. Moreover, it is also shown that LF I and II algorithms are sensitive to initial conditions in a reduced scale. Although Yu *et al.* [5] and Yu *et al.* [16] have used Lyapunov function based weight update algorithms, none of them address the issue of computation of adaptive learning rate in a formal manner, nor any of these approaches have investigated the nature of convergence for these kind of algorithms. Since BP algorithm is very popular among users of feedforward networks, they would find the present approach more insightful and intuitive.

It is well known that a momentum term may be added to BP algorithm in order to speed up its convergence rate. Such a term arises naturally in our algorithm and thus it provides a theoretical justification for such kind of modifications. We also show that by adding an acceleration term, it is possible to avoid local minima to a greater extent. Through simulations, it is shown that such a term also leads to faster convergence in some cases.

The proposed algorithms, LF I and II, are tested on three bench-mark function approximation problems namely, XOR, 3-bit parity, and 8-3 encoder. The results are compared with those of BP and EKF. Finally, the efficacy of the proposed algorithm to approximate a two-dimensional (2-D) Gabor function is analyzed and a comparison is made with its gradient–descent counterpart.

The paper is organized as follows. Sections II and III describe Lyapunov function-based learning algorithm (LF I) and its modified version (LF II), respectively. The simulation results are provided in Section IV. The concluding remarks are given in Section V. Two algorithms, BP and EKF, are given in Appendix.

II. Lyapunov Function (LF I)-Based Learning Algorithm

A simple feedforward neural network with single output is shown in Fig. 1. $\phi(.)$ is the nonlinear activation function for neurons. The network is parameterized in terms of its weights which can be represented as a weight vector $\mathbf{W} \in R^m$. For a specific function approximation problem, the training data consists of N (say) patterns $\{\mathbf{x}^p, y^p\}$, $p = 1, 2, \ldots, N$. For a specific pattern p, if the input vector is $\mathbf{x}^p$, then the network output is given by

$$y^p = f(\mathbf{W}, \mathbf{x}^p), \qquad p = 1, 2, \ldots N. \tag{1}$$

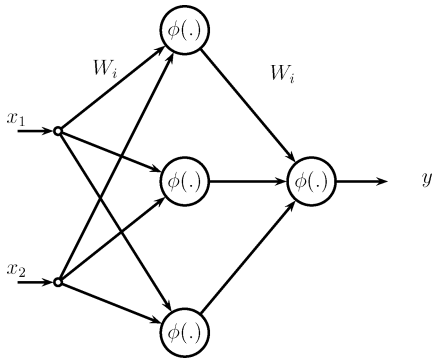


Fig. 1. Feedforward neural network.

The usual quadratic cost function which is minimized to train the weight vector $\mathbf{W}$ is given by

$$E = \frac{1}{2}\sum_{p=1}^{N}(y_d^p - y^p)^2. \tag{2}$$

In order to derive a weight update algorithm for such a network, we consider a Lyapunov function candidate as

$$V_1 = \frac{1}{2}(\tilde{\mathbf{y}}^T\tilde{\mathbf{y}}) \tag{3}$$

where $\tilde{\mathbf{y}} = [y_d^1 - y^1, \ldots\ldots, y_d^p - y^p, \ldots\ldots, y_d^N - y^N]^T$. As can be seen, in this case the Lyapunov function is the same as the usual quadratic cost function minimized during batch update using BP learning algorithm. The time derivative of the Lyapunov function V_1 is given by

$$\dot{V}_1 = -\tilde{\mathbf{y}}^T\frac{\partial \mathbf{y}}{\partial W}\dot{\mathbf{W}} = -\tilde{\mathbf{y}}^T\mathbf{J}\dot{\mathbf{W}} \tag{4}$$

where

$$J = \frac{\partial \mathbf{y}}{\partial \mathbf{W}} \quad \in R^{N\times m}. \tag{5}$$

Theorem 1: If an arbitrary initial weight $\mathbf{W}(0)$ is updated by

$$\mathbf{W}(t') = \mathbf{W}(0) + \int_0^{t'}\dot{\mathbf{W}}dt \tag{6}$$

where

$$\dot{\mathbf{W}} = \frac{\| \tilde{\mathbf{y}} \|^2}{\| \mathbf{J}^T\tilde{\mathbf{y}} \|^2}\mathbf{J}^T\tilde{\mathbf{y}} \tag{7}$$

then $\tilde{\mathbf{y}}$ converges to zero under the condition that $\dot{\mathbf{W}}$ exists along the convergence trajectory.

Proof: Substituting (7) into (4), we have

$$\dot{V}_1 = -\| \tilde{\mathbf{y}} \|^2 \leq 0 \tag{8}$$

where $\dot{V}_1 < 0$ for all $\tilde{\mathbf{y}} \neq 0$. If $\dot{V}_1$ is uniformly continuous and bounded, then according to *Barbalat's lemma* [22] as $t \to \infty$, $\dot{V}_1 \to 0$, and $\tilde{\mathbf{y}} \to 0$. ■

The weight update law given in (7) is a batch update law. Analogous to instantaneous gradient-descent (GD) (or BP) algorithm, the instantaneous LF I learning algorithm can be derived as

$$\dot{\mathbf{W}} = \frac{\| \tilde{y} \|^2}{\| \mathbf{J_p}^T \tilde{y} \|^2} \mathbf{J_p}^T \tilde{y} \tag{9}$$

where $\tilde{y} = y_d^p - y^p \in R$ and $\mathbf{J_p} = (\partial y^p / \partial \mathbf{W}) \in R^{1 \times m}$ is the instantaneous value of the Jacobian. The difference equation representation of the weight update algorithm based on (9) is given by

$$\begin{aligned} \mathbf{W}(t+1) &= \mathbf{W}(t) + \mu \dot{\mathbf{W}}(t) \\ &= \mathbf{W}(t) + \left(\mu \frac{\| \tilde{y} \|^2}{\| \mathbf{J_p}^T \tilde{y} \|^2} \right) \mathbf{J_p}^T \tilde{y}. \end{aligned} \tag{10}$$

Here μ is a constant which is selected heuristically. We can add a very small constant ϵ to the denominator of (9) to avoid numerical instability when error $\tilde{y}$ goes to zero. Now, we compare the LF I with the BP algorithm based on GD principle.

In the instantaneous GD method, we have

$$\triangle \mathbf{W} = -\eta \left(\frac{\partial E}{\partial \mathbf{W}} \right)^T = \eta \mathbf{J_p}^T \tilde{y} \tag{11}$$

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta \mathbf{J_p}^T \tilde{y} \tag{12}$$

where η is the learning rate. Comparing (12) with (10), we see a very interesting similarity where the fixed learning rate η in BP algorithm is replaced by its adaptive version η_a given by

$$\eta_a = \left(\mu \frac{\| \tilde{y} \|^2}{\| \mathbf{J_p}^T \tilde{y} \|^2} \right). \tag{13}$$

This is the most remarkable finding of this paper. Earlier, there have been many research papers concerning the adaptive learning rate [6], [20], [21], [23]. However, in this paper, we formally derive this adaptive learning rate using Lyapunov function approach, and is a natural key contribution in this field. We analyze the nature of adaptive learning rate through simulations in Section IV, where we will show that this adaptive learning rate makes the algorithm faster than the conventional BP.

A. Convergence of LF I

Theorem 1 states that the global convergence of the learning algorithm (7) is guaranteed provided $\dot{\mathbf{W}}$ exists and is nonzero along the convergence trajectory. This, in turn, necessitates $\| \partial V_1 / \partial \mathbf{W} \| = \| J^T \tilde{\mathbf{y}} \| \neq 0$. The condition $\| \partial V_1 / \partial \mathbf{W} \| = 0$ represents local minima of the scalar function (3). Thus, Theorem 1 says that the global minimum is reached only when local minima are avoided during training. Since instantaneous update rule introduces noise, it may be possible to reach global minimum in some cases, however, the global convergence is not guaranteed.

III. Modified Lyapunov Function (LF II)-Based Learning Algorithm

In this section, we consider a modified Lyapunov function in order to further improve the convergence properties of LF I. We consider the following Lyapunov function:

$$V_2 = \frac{1}{2} (\tilde{\mathbf{y}}^T \tilde{\mathbf{y}} + \lambda \dot{\mathbf{W}}^T \dot{\mathbf{W}}) = V_1 + \frac{\lambda}{2} \dot{\mathbf{W}}^T \dot{\mathbf{W}} \tag{14}$$

where λ is a positive constant. The time derivative of (14) is given by

$$\dot{V}_2 = -\tilde{\mathbf{y}}^T \frac{\partial \mathbf{y}}{\partial \mathbf{W}} \dot{\mathbf{W}} + \lambda \ddot{\mathbf{W}}^T \dot{\mathbf{W}} = -\tilde{\mathbf{y}}^T (\mathbf{J} - \mathbf{D}) \dot{\mathbf{W}} \tag{15}$$

where $\mathbf{J} = \partial \mathbf{y} / \partial \mathbf{W} : N \times m$ is the Jacobian matrix, and

$$\mathbf{D} = \lambda \frac{1}{\|\tilde{\mathbf{y}}\|^2} \tilde{\mathbf{y}} \ddot{\mathbf{W}}^T \quad \in R^{N \times m}. \tag{16}$$

Theorem 2: If the update law for weight vector $\mathbf{W}$ follows a dynamics given by following nonlinear differential equation:

$$\dot{\mathbf{W}} = \alpha(\mathbf{W}) \mathbf{J}^T \tilde{\mathbf{y}} - \alpha(\mathbf{W}) \ddot{\mathbf{W}} \tag{17}$$

where $\alpha(\mathbf{W}) = \| \tilde{\mathbf{y}} \|^2 / (\| \mathbf{J}^T \tilde{\mathbf{y}} \|^2 + \epsilon)$ is a scalar function of weight vector $\mathbf{W}$ and ϵ is a small positive constant, then $\tilde{\mathbf{y}}$ converges to zero under the condition that $(\mathbf{J} - \mathbf{D})^T \tilde{\mathbf{y}}$ is nonzero along the convergence trajectory.

Proof: Equation (17) may be rewritten as

$$\dot{\mathbf{W}} = \frac{\|\tilde{\mathbf{y}}\|^2}{\|\mathbf{J}^T \tilde{\mathbf{y}}\|^2 + \epsilon} (\mathbf{J} - \mathbf{D})^T \tilde{\mathbf{y}}. \tag{18}$$

Substituting for $\dot{\mathbf{W}}$ from (17) into (15), we get

$$\dot{V}_2 = -\frac{\| \tilde{\mathbf{y}} \|^2}{\| \mathbf{J}^T \tilde{\mathbf{y}} \|^2 + \epsilon} \| (\mathbf{J} - \mathbf{D})^T \tilde{y} \|^2 \leq 0. \tag{19}$$

Since $(\mathbf{J} - \mathbf{D})^T \tilde{\mathbf{y}}$ is nonzero, $\dot{V}_2 < 0$ for all $\tilde{\mathbf{y}} \neq 0$ and $\dot{V}_2 = 0$ iff $\tilde{\mathbf{y}} = 0$. If $\dot{V}_2$ is uniformly continuous and bounded, then according to *Barbalat's lemma* [22] as $t \to \infty$, $\dot{V}_2 \to 0$ and $\tilde{\mathbf{y}} \to 0$. ■

We discuss various convergence conditions later in Section III.

As derived in LF I, the instantaneous weight update equation using modified Lyapunov function can be finally expressed in difference equation model as follows:

$$\begin{aligned} \mathbf{W}(t+1) &= \mathbf{W}(t) + \left(\mu \frac{\|\tilde{y}\|^2}{\|\mathbf{J_p}^T \tilde{y}\|^2 + \epsilon} \right) (\mathbf{J_p} - \mathbf{D})^T \tilde{y} \\ &= \mathbf{W}(t) + \mu \frac{\|\tilde{y}\|^2}{\|\mathbf{J_p}^T \tilde{y}\|^2 + \epsilon} \mathbf{J_p}^T \tilde{y} \\ &\quad - \mu_1 \frac{\ddot{\mathbf{W}}(t)}{\|\mathbf{J_p}^T \tilde{y}\|^2 + \epsilon} \end{aligned} \tag{20}$$

where $\mu_1 = \mu \lambda$ and the acceleration $\ddot{\mathbf{W}}(t)$ is computed as

$$\ddot{\mathbf{W}}(t) = \frac{1}{(\triangle t)^2} [\mathbf{W}(t) - 2\mathbf{W}(t-1) + \mathbf{W}(t-2)]$$

and $\triangle t$ is taken to be one time unit for simulation.

To draw out a similar comparison between LF II and BP algorithms, we reconsider the cost function (14) and apply gradient–descent to compute the weight update as follows:

$$\begin{aligned}\triangle \mathbf{W} &= -\eta \left(\frac{\partial V_2}{\partial \mathbf{W}}\right)^T \\ &= -\eta \left(\frac{\partial V_1}{\partial \mathbf{W}}\right)^T - \eta \left[\frac{d}{d\mathbf{W}}\left(\frac{\lambda}{2}\dot{\mathbf{W}}^T\dot{\mathbf{W}}\right)\right]^T \\ &= \eta \left(\frac{\partial y}{\partial \mathbf{W}}\right)^T \tilde{y} - \eta\lambda\ddot{\mathbf{W}}.\end{aligned}$$

Thus, the weight update equation for GD method may be written as

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta' {\mathbf{J_p}}^T \tilde{y} - \mu'\ddot{\mathbf{W}} \tag{21}$$

where $\mu' = \eta\lambda$. The third term on the right-hand side of (21) is an acceleration term similar to a momentum term used with conventional BP algorithm. Comparing (20) and (21), the adaptive learning rate in this case is given by

$$\eta'_a = \mu \frac{\|\tilde{y}\|^2}{\|{\mathbf{J_p}}^T\tilde{y}\|^2 + \epsilon} \tag{22}$$

and the adaptive acceleration rate is given by

$$\mu'_a = \frac{\lambda}{\|{\mathbf{J_p}}^T\tilde{y}\|^2 + \epsilon}. \tag{23}$$

Previously, attempts have been made in obtaining adaptive learning rate and momentum rate [21], [23] and this algorithm also belongs to this category with the distinction that the adaptive terms are derived using Lyapunov stability concept.

Although, above algorithms are derived for a single-output network, they are applicable to multioutput networks as well. From now onward, we will use following notations for discussions pertaining to instantaneous update:

- $\tilde{\mathbf{y}} \in R^n$ represents error signal for a n output network;
- $J = (\partial \mathbf{y}/\partial \mathbf{W}) \in R^{n\times m}$ where $\mathbf{W} \in R^m$.

A. Convergence of LF II

The scalar function V_2 in (14) is a positive–definite function whose equilibrium point is given by

$$\tilde{\mathbf{y}} = 0 \quad \dot{\mathbf{W}} = 0.$$

According to Theorem 2, starting from any initial condition, one can reach the global minimum provided $\dot{\mathbf{W}}$ is nonzero along the convergence trajectory. $\dot{\mathbf{W}}$ vanishes under following condition:

$$J = D. \tag{24}$$

Both J and D matrices are of dimension $R^{n\times m}$. In case of NN, it is very unlikely that each element of J would be equal to that of D, thus this possibility can easily be ruled out for a multilayer perceptron network.

The following observations can be made regarding convergence of LF II algorithm.

- $\dot{\mathbf{W}}$ vanishes whenever

  $$(J-D)^T\tilde{\mathbf{y}} = 0$$

  that is, $\tilde{\mathbf{y}}$ belongs to the null space of $(J-D)$. Assuming that $J \neq D$, rank of $(J-D)$ is at most n and usually $n < m$ for a neural network. If rank of $(J-D)$ is n, then $(J-D)$ has a trivial null space given by $\tilde{\mathbf{y}} = 0$, which is the global minimum. Hence rank $\rho(J-D) = n$ ensures global convergence.
- Rewriting the local minima condition as

  $$J^T\tilde{\mathbf{y}} = D^T\tilde{\mathbf{y}}$$

  or

  $$J^T\tilde{\mathbf{y}} = \lambda\ddot{\mathbf{W}}$$

  we see that the solutions of aforementioned equation represent local minima of the cost function (14). The solution to this equation exists for every vector $\ddot{\mathbf{W}} \in R^m$ whenever rank $\rho(J) = m$ [24]. It is to be noted that the size of output vector is usually smaller than the size of weight vector (i.e, $n < m$); so, rank $\rho(J) \leq n$. Hence there are at least $m-n$ vectors $\ddot{\mathbf{W}} \in R^m$ for which solutions do not exist and, hence, local minima do not occur; so by increasing hidden neurons, it is possible to increase m without increasing n, thereby decreasing the chances of encountering local minima. Increasing the number of hidden layers also have same effect of reducing the chances of encountering local minima.
- Increasing the number of output neurons increases both m and n but not at the same rate, that is, the ratio n/m increases with increasing number of outputs. Thus, for multi-output systems, there are more local minima (for fixed number of hidden neurons) as compared to single-output systems.

We see that LF II also does not guarantee global convergence because it is not possible to ensure rank $\rho(J-D) = n$. However, it is possible to avoid local minima by a suitable choice of network architecture (hidden neurons, outputs, and activation function). The issue of global convergence has also been studied by Yu *et al.* [5]. The Lyapunov function candidate considered in this work has the following form:

$$V_2 = \mu V_1 + \frac{1}{2}\sigma \left\| \frac{\partial V_1}{\partial \mathbf{W}} \right\|^2 \tag{25}$$

where $V_1 = (1/2)\tilde{\mathbf{y}}^T\tilde{\mathbf{y}}$. The function V_2 is minimum when V_1 and $\partial V_1/\partial \mathbf{W}$ are simultaneously at minimum. Thus the next step would be to select a weight update law $\dot{\mathbf{W}}$ such that the global minimum, given by $V_1 = 0$ and $\partial V_1/\partial \mathbf{W} = 0$, is reached.

Assuming that V_2 is an implicit function of input $\mathbf{x}$ and time t, the time derivative of the Lyapunov function $\dot{V}_2$ is given as

$$\dot{V}_2 = \frac{\partial V_1}{\partial \mathbf{W}}\left[\mu I + \sigma \frac{\partial^2 V_1}{\partial \mathbf{W}\partial \mathbf{W}^T}\right]\dot{\mathbf{W}}. \tag{26}$$

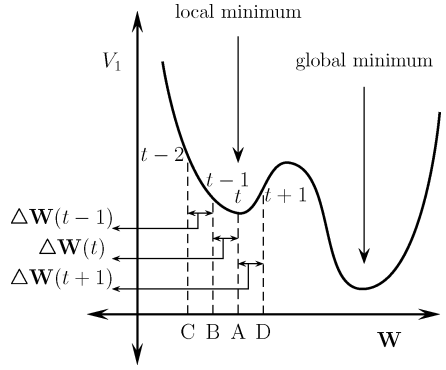


Fig. 2. Local minimum of a cost function.

If the weight update law $\dot{\mathbf{W}}$ is selected as

$$\dot{\mathbf{W}} = -\left[\mu I + \sigma \frac{\partial^2 V_1}{\partial \mathbf{W} \partial \mathbf{W}^T}\right]^{-1} \frac{\left(\frac{\partial V_1}{\partial \mathbf{W}}\right)^T}{\left\|\frac{\partial V_1}{\partial \mathbf{W}}\right\|^2} \left(\zeta \left\|\frac{\partial V_1}{\partial \mathbf{W}}\right\|^2 + \eta \|V_1\|^2\right) \quad (27)$$

with $\zeta > 0$ and $\eta > 0$, then

$$\dot{V}_2 = -\zeta \left\|\frac{\partial V_1}{\partial \mathbf{W}}\right\|^2 - \eta \|V_1\|^2 \quad (28)$$

which is negative–definite with respect to V_1 and $\partial V_1/\partial \mathbf{W}$. According to La Salle and Yoshizaw's theorem [22], V_2 will finally converge to its equilibrium point given by $V_1 = 0$ and $\partial^T V_1/\partial \mathbf{W} = \mathbf{0}$.

However, the implementation of the weight update algorithm (27) becomes very difficult due to the presence of a Hessian term $\partial^2 V_1/\partial \mathbf{W} \partial \mathbf{W}^T$. Thus the weight update law (27) turns out to be of theoretical interests only. Interestingly, the algorithms such as BP, Gauss–Newton, LM can be shown as special cases of this generalized weight update algorithm (27). Our motivation has been to achieve faster convergence without sacrificing the simplicity of BP algorithm and it is shown in the simulation section that this objective has been achieved through LF algorithms.

B. Avoiding Local Minima

Since we are using the difference equation model (20) for our simulation, we give an intuitive argument about how an acceleration term as in (21) can possibly avoid local minima present in BP algorithm.

Equation (21) may be rewritten as

$$\begin{aligned}\mathbf{W}(t+1) &= \mathbf{W}(t) + \triangle\mathbf{W}(t+1) \\ &= \mathbf{W}(t) - \eta' \frac{\partial V_1}{\partial \mathbf{W}}(t) - \mu' \ddot{\mathbf{W}}(t). \end{aligned} \quad (29)$$

A local minimum condition for cost function V_1 is shown in Fig. 2. This local minimum point is indicated by letter "A." $\triangle\mathbf{W}(t+1)$ is the weight increment for the interval $(t, t+1]$. We use this notation to facilitate discussion in the remaining part of Section III-B.

On the left-hand side of "A" in Fig. 2, $\partial V_1/\partial \mathbf{W} < 0$ and BP algorithm tends to increase the weights ($\because \triangle\mathbf{W} \propto -\partial V_1/\partial \mathbf{W}$). Since the slope $\partial V_1/\partial \mathbf{W}$ decreases towards the local minimum, the increment $\triangle\mathbf{W}$ or rate of weight update $\dot{\mathbf{W}}$ also goes on decreasing towards the local minimum. In other words, the acceleration of weight update $\ddot{\mathbf{W}}$ is negative on left-hand side of local minimum. This makes the third term positive and, hence, helps in increasing the rate of weight update, thereby speeding up the convergence. Things would become clearer through following discussion.

Consider the point "B" on left-hand side of local minimum (refer to Fig. 2). Let the incremental contribution due to BP term [second term in (29)] be $\triangle\mathbf{W}_1$ and the contribution due to acceleration term be $\triangle\mathbf{W}_2$. Also assume that point "B" corresponds to time $t-1$. The weight update for the interval $(t-1, t]$ computed at this instant is given by $\triangle\mathbf{W}(t) = \triangle\mathbf{W}_1(t-1) + \triangle\mathbf{W}_2(t-1)$. At this point, we have

$$\begin{aligned}\triangle\mathbf{W}_1(t-1) &= -\eta \frac{\partial V_1}{\partial \mathbf{W}}(t-1) > 0 \\ \triangle\mathbf{W}_2(t-1) &= -\mu \ddot{\mathbf{W}}(t-1) \\ &= -\mu(\triangle W(t-1) - \triangle W(t-2)) > 0.\end{aligned}$$

It is to be noted that $\triangle W(t-1) < \triangle W(t-2)$ as the velocity is decreasing towards the point of local minimum. This makes the right-hand side of second equation positive. As we can see here, the terms $\triangle\mathbf{W}_1$ and $\triangle\mathbf{W}_2$ are assisting each other on the left-hand side of local minimum and, hence, helps in speeding up the convergence.

Now, consider the point "A" at time instant t which is a local minimum. At this point, we have

$$\begin{aligned}\triangle\mathbf{W}_1(t) &= -\eta \frac{\partial V_1}{\partial \mathbf{W}}(t) = 0 \\ \triangle\mathbf{W}_2(t) &= -\mu \ddot{\mathbf{W}}(t) = -\mu(\triangle W(t) - \triangle W(t-1)) > 0.\end{aligned}$$

Here also, we have $\triangle W(t) < \triangle W(t-1)$ and this leads to a positive contribution due to acceleration term. We see that at local minimum, the weight increment $\triangle\mathbf{W}(t+1) = \triangle\mathbf{W}_1(t) + \triangle\mathbf{W}_2(t)$ is contributed only by the acceleration term. Due to this positive contribution, the weight vector moves to a point (say) "D" at time $t+1$. At this point, we have

$$\begin{aligned}\triangle\mathbf{W}_1(t+1) &= -\eta \frac{\partial V_1}{\partial \mathbf{W}}(t+1) < 0 \\ \triangle\mathbf{W}_2(t+1) &= -\mu \ddot{\mathbf{W}}(t+1) \\ &= -\mu(\triangle W(t+1) - \triangle W(t)) > 0.\end{aligned}$$

The contribution due to BP term becomes negative as the slope $\partial V_1/\partial \mathbf{W} > 0$ on the right-hand side of local minimum. We can argue that $\triangle W(t+1) < \triangle W(t)$ because the term $\triangle\mathbf{W}(t)$ is contributed by both BP as well as acceleration terms of the same sign, while the term $\triangle\mathbf{W}(t+1)$ is contributed only by acceleration term. Moreover, the contribution due to acceleration term $\triangle\mathbf{W}_2(t)$ in the interval $(t, t+1]$ is less than $\triangle\mathbf{W}_2(t-1)$ in the interval $(t-1, t]$ because of the fact that

acceleration is negative on the left-hand side of local minima. Both of these two reasons make $\triangle \mathbf{W}_2(t+1)$ positive.

Thus, $\triangle \mathbf{W}(t+2) = \triangle \mathbf{W}_1(t+1) + \triangle \mathbf{W}_2(t+1)$ may become positive if the incremental contribution due to acceleration dominates the incremental contribution due to BP. In such a case, it might be possible to overcome a local minimum of smaller potential barrier as shown in Fig. 2 and reach the global minimum.

Through simulations, we found that the addition of term $-\mu \ddot{\mathbf{W}}$ to BP results in faster convergence, in the same manner as seen with additional momentum term. Now, it is observed that the LF II update law (20) is of the same form as (29) with adaptive coefficients η'_a and μ'_a. Thus, it is natural to expect that such a term would help in avoiding local minima to a greater extent as compared to LF I. In Section IV, we analyze the effect of this additional acceleration term on convergence performance of LF II.

This discussion can be summarized as follows:

- LF I and LF II improves the convergence rate of BP algorithm by introducing an adaptive learning rate;
- the generalized weight update law (27) proposed in [5] is of theoretical interest only while LF I and II are practically implementable;
- LF II helps in avoiding local minima to a greater extent as compared to LF I.

IV. Simulation Results

A two-layered feedforward network is selected for each problem. Unity bias is applied to all the neurons. We test proposed algorithms LF I and LF II on three bench-mark problems, XOR, 3-bit parity, and 8-3 encoder, and a system-identification problem. The proposed algorithms are compared with popular BP and EKF algorithms which are provided in the Appendix. For XOR, 3-bit parity, and 8-3 encoder problems, we have taken unipolar sigmoid as our activation function. The patterns are presented sequentially during training. For benchmark problems, the training is terminated when the root-mean-square (rms) error per epoch reaches 10^{-4}. Since the weight search starts from initial small random values, and each initial weight vector selection can lead to a different convergence time, average convergence time is calculated for fifty different runs. *Each run implies that the network is trained from any arbitrary random weight initialization*. In BP algorithm, the value of learning rate η is taken to be 0.95. It is to be noted that in usual cases, learning rate for BP is taken to be much smaller than this value. But in our case, the problems being simpler, we are able to increase the speed of convergence by increasing this learning rate. We have deliberately done this to show that the proposed algorithms are still faster than this. The initial value of λ in EKF is 0.9. The value of constant μ in both LF I and II is selected heuristically for best performance. Its value lies between 0.2–0.8 and λ in LF II (16) lies between 0.01–0.1.

A. XOR

For XOR, we have taken four neurons in the hidden layer and the network has two inputs and one output. The adaptive learning rates of LF I and LF II are shown in Figs. 3 and 4, respectively. It can be seen that the adaptive learning rate becomes

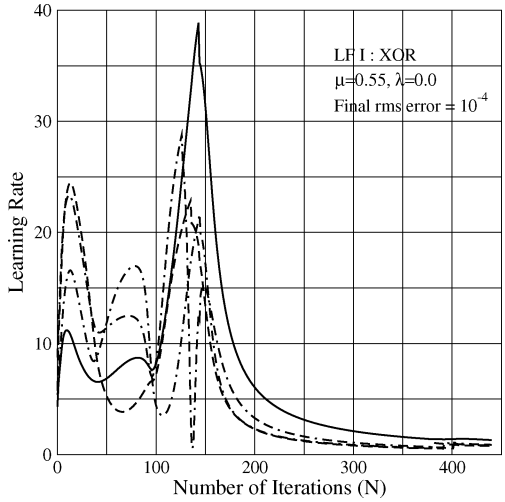


Fig. 3. Adaptive learning rates for LF I. The four curves correspond to four patterns of the XOR problem. The number of epochs of training data required can be obtained by dividing the number of iterations by four.

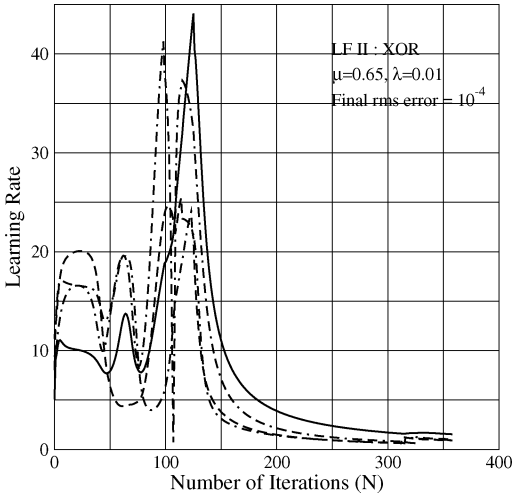


Fig. 4. Adaptive learning rates for LF II. The four curves correspond to four patterns of the XOR problem. The number of epochs of training data required can be obtained by dividing the number of iterations by four.

TABLE I
COMPARISON AMONG THREE ALGORITHMS FOR XOR PROBLEM

Algorithm	epochs	time (sec)	parameters
BP	5620	0.0578	$\eta = 0.5$
BP	3769	0.0354	$\eta = 0.95$
EKF	3512	0.1662	$\lambda = 0.9$
LF-I	165	0.0062	$\mu = 0.55$
LF-II	120	0.0038	$\mu = 0.65, \lambda = 0.01$

zero as the network gets trained. The simulation results for XOR are given in Table I. It can be seen that LF I takes minimum number of epochs for convergence as compared to BP and EKF – in this case LF I is 20 times faster in terms of number of epochs compared to BP with η =0.95. However, it can be seen that LF I is nearly five times faster than the same BP algorithm in terms of computation time. Fig. 5 gives a better insight into the performance of various algorithms for different initial conditions. In Fig. 5, each run refers to a different random initialization of the weight vector $\mathbf{W}$. It can be seen that the number of epochs (or

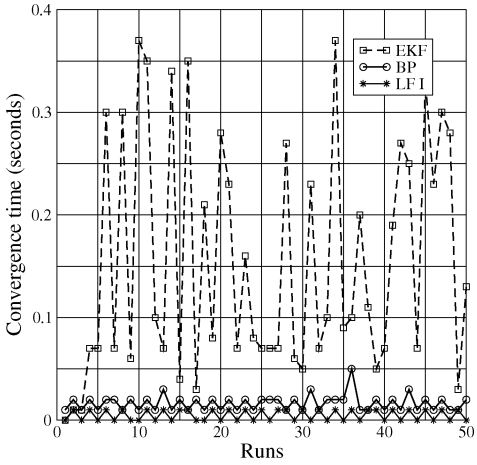


Fig. 5. Convergence time comparison for among BP, EKF, and LF II.

TABLE II
COMPARISON AMONG THREE ALGORITHMS FOR 3-BIT PARITY PROBLEM

Algorithm	epochs	time (sec)	parameters
BP	12032	0.483	$\eta = 0.5$
BP	5941	0.2408	$\eta = 0.95$
EKF	2186	0.4718	$\lambda = 0.9$
LF-I	1338	0.1176	$\mu = 0.47$
LF-II	738	0.0676	$\mu = 0.47, \lambda = 0.03$

convergence time) for training fluctuates very much for both BP and EKF. This fluctuation is very much reduced in case of LF I. LF II provides tangible improvement over LF I both in terms of convergence time as well as training epochs as can be seen in Fig. 6(a). This is possibly because LF II helps in avoiding local minima to a greater extent.

B. 3-Bit Parity

For parity problem, we have chosen a network with three inputs and seven hidden neurons. Table II shows the simulation results for this problem. Here also, we find that LF I and LF II outperform both BP and EKF in terms of convergence time as well as number of training patterns. In this case, we find that LF II is nearly five times faster as compared to BP as far as training patterns are concerned. However, there is only a marginal improvement as far as computation time is concerned. EKF might be faster for a particular choice of initial condition but on an average it is slower as compared to BP in terms of computation time. In Fig. 6(b), we can observe that LF II performs better than LF I both in terms of computation time as well as training epochs. Reduction in computation time and training examples in case of LF II is nearly half as compared to that in LF I.

C. 8-3 Encoder

For encoder, we take a network with eight inputs, three outputs, and 16 hidden neurons. The simulation results are shown in the Table III. There are eight patterns in one epoch. EKF does not converge to the rms error of 0.01 within 10 000 epochs with this architecture and that is why it is excluded from comparison analysis. As it can be seen, LF I and LF II perform better than

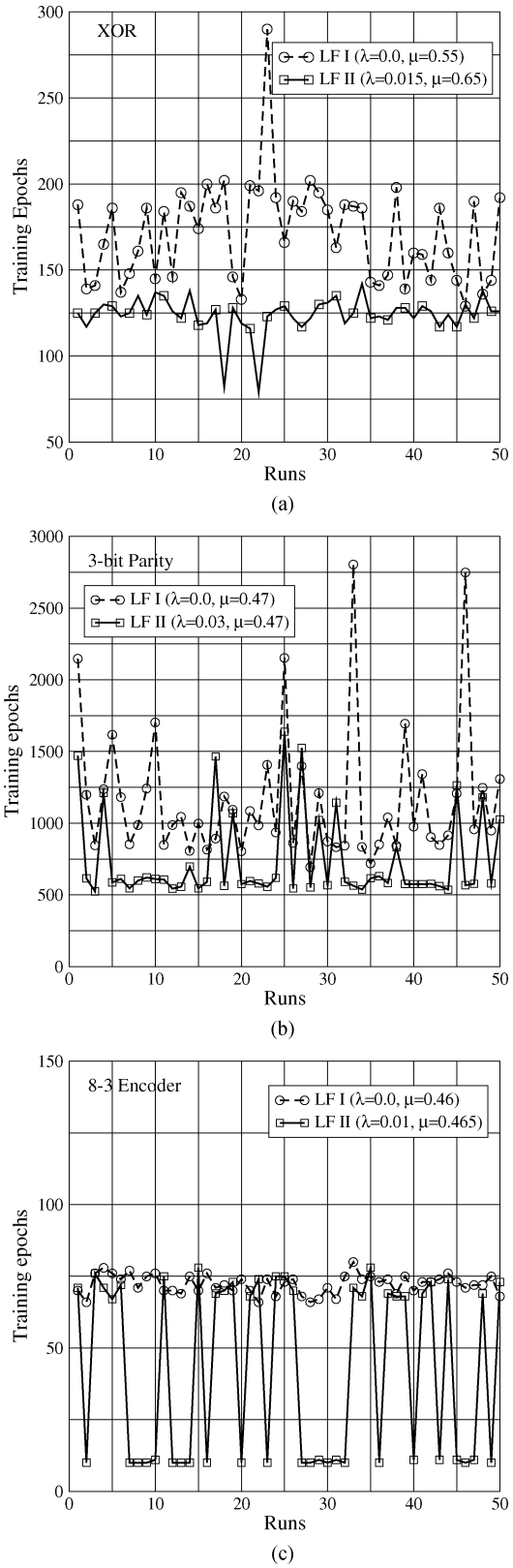


Fig. 6. Comparison of convergence time in terms of iterations between LF I and LF II. (a) XOR. (b) 3-bit parity. (c) 8-3 encoder.

BP. It is seen that LF I does not converge for values of μ greater than 0.5, but with LF II it is possible to find a suitable λ for any μ so that it converges. Here, parameters μ and λ are so chosen as to get faster response. This is shown in Fig. 6(c).

TABLE III
COMPARISON AMONG THREE ALGORITHMS FOR 8-3 ENCODER PROBLEM

Algorithm	epochs	time (sec)	parameters
BP	326	0.044	$\eta = 0.7$
BP	255	0.0568	$\eta = 0.9$
LF-I	72	0.0582	$\mu = 0.46$
LF-II	42	0.051	$\mu = 0.465, \lambda = 0.01$

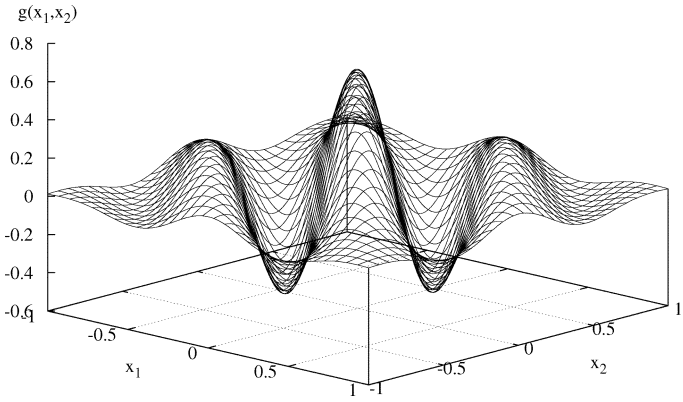


Fig. 7. Two-dimensional (2-D) Gabor function.

D. Summary

We see that by the virtue of adaptive learning rate, it is possible to speed up the convergence rate of BP learning algorithm. LF II provides faster convergence in all cases as it helps in avoiding local minima to a greater extent.

E. Two-Dimensional Gabor Function

The convolution version of complex 2-D Gabor function [25] has the following form:

$$g(x_1, x_2) = \frac{1}{2\pi\lambda\sigma^2} e^{-[(x_1/\lambda)^2 + {x_2}^2]/(2\sigma^2)} e^{2\pi i(u_0 x_1 + v_0 x_2)} \tag{30}$$

where λ is an aspect ratio, σ is a scale factor, and u_0 and v_0 are modulation parameters. In this simulation, the following Gabor function is used:

$$g(x_1, x_2) = \frac{1}{2\pi(0.5)^2} e^{-[({x_1}^2 + {x_2}^2)/2(0.5)^2]} \cos(2\pi(x_1 + x_2)). \tag{31}$$

The 2-D Gabor function is shown in Fig. 7. For this problem, 10 000 training data are generated where input variables x_1 and x_2 are sampled randomly with uniform distribution in the range $[-0.5, +0.5]$. However, 10 000 test data are generated using a regular pattern. In this case, for each value of x_1, 100 values of x_2 are selected sequentially in the range of $[-0.5, +0.5]$ with an interval of 0.01. We consider a radial basis function network for approximating this function. The centers and weights are trained using BP and LF algorithms. After training, the rms error was computed over the 10 000 test data. This rms error is averaged over 50 different runs. The results are summarized in Table IV. It can be seen that LF I provides better error convergence as compared to BP for half the number of centers. The performance still improves with LF II. The improvement in error convergence with LF II becomes more apparent in Fig. 8, where it is observed that as the number of training data is increased LF II gives better accuracy as compared to LF I.

TABLE IV
PERFORMANCE RESULTS FOR GABOR FUNCTION

Algorithm	No. of Centers	rms error/run	parameters
BP	40	0.0847241	$\eta_{1\text{-}2} = 0.2$
BP	80	0.0314169	$\eta_{1\text{-}2} = 0.2$
LF-I	40	0.0192033	$\mu = 0.8$
LF-II	40	0.0186757	$\mu = 0.8, \lambda = 0.3$

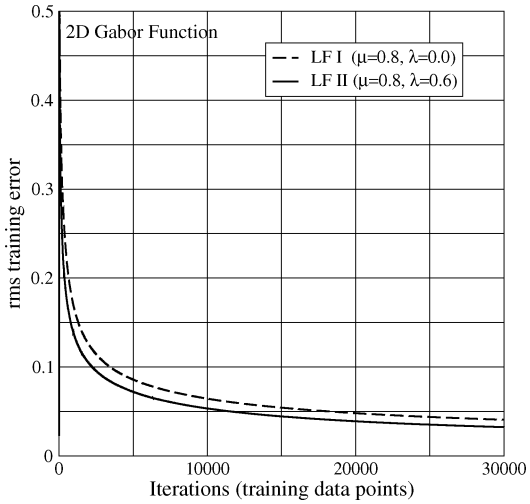


Fig. 8. Performance comparison between LF I and LF II for 2-D Gabor function.

V. CONCLUSION

We have proposed a novel algorithm for weight update in feedforward networks using Lyapunov function approach. The key contribution of the paper is to show a parallel between proposed LF I and II algorithms and popular BP algorithm. It is shown that the proposed algorithms have the same structure as that of popular BP algorithm with the difference that the fixed learning rate in BP is replaced by an adaptive learning rate. In LF I, the adaptive learning rate is found to be $\eta_a = \left(\mu(\|\tilde{y}\|^2/\|\mathbf{J_p}^T\tilde{y}\|^2)\right)$, while in LF II we have both an adaptive learning rate as well as an adaptive acceleration term. Through analysis, it is shown that this additional acceleration term tends to avoid local minima and, thus, increases the chances of attaining global minimum. Detailed analysis of convergence properties of these two algorithms for feedforward networks is carried out which provides better understanding of the working of feedforward networks. It shows how one can avoid local minima by properly choosing the network architecture. This paper is first of its kind to provide an exact expression for adaptive learning rate. Through simulation results on three bench mark problems, we establish that the proposed algorithms LF I and LF II outperform both popular BP and EKF algorithms in terms of convergence speed. When the proposed algorithm is tested for approximation of 2-D Gabor function using radial basis function network, LF algorithm achieves better accuracy than that of BP

$$\delta_j(n) = \begin{cases} e_j(n)\psi_j'(v_j(n)), & \text{if neuron } j \text{ is an output node} \\ \psi_j'(v_j(n))\sum_k \delta_k(n) w_{kj}(n), & \text{if neuron } j \text{ is a hidden node} \end{cases} \quad (34)$$

algorithm with 50% reduction in the number of centres. This proves the efficacy of proposed algorithms. Although the simulation results for LF II are based on difference equation model of weight update (20), the results have also been verified by integrating the original differential equation (17) using Euler's method.

Since BP algorithm is very popular among users of feedforward networks, readers will benefit from knowing that a proper adaptive learning rate can be found that may transform a locally convergent BP algorithm into a globally convergent one.

APPENDIX

A. BP Algorithm

BP algorithm is based on GD method in which the weight is updated in such a direction as to reduce the error. The weight update rule [6] is given by

$$\triangle w_i(t) = -\eta \frac{\partial E}{\partial w_i} \quad (32)$$

$$\triangle w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad (33)$$

where η is the learning rate parameter of the BP algorithm and the local gradient $\delta_j(n)$ is given by (34), as shown at the top of the page, where k refers to the next layer.

B. EKF

The EKF can be characterized as an algorithm for computing the conditional mean and covariance of the probability distribution of a nonlinear dynamic system with uncorrelated Gaussian process and measurement noise. The conditional mean is the unique unbiased estimate. This algorithm uses second-order process training that processes and uses information about the shape of the training problem's underlying error surface. This is a method of estimating the state vector. Here, the weight vector $a(t)$ is considered as the state vector to be estimated

$$a = [(a^1)^T, (a^2)^T, \ldots, (a^{M-1})^T]^T, \qquad (L \times 1)$$

$$a^n{}_i = [a^n{}_{i,1}, a^n{}_{i,2}, \ldots, a^n{}_{i,N_n}]^T, \qquad (N_n \times 1)$$

$$a^n = [(a^n{}_1)^T, (a^n{}_2)^T, \ldots, (a^n{}_{N_{n+1}-1})^T]^T, \qquad (N_n(N_{n+1}-1) \times 1). \quad (35)$$

In addition, the total number of the link weights is defined by $L = \sum_{n=1}^{M-1} N_n(N_{n+1}-1)$. The MLP is then expressed by the following nonlinear system equations:

$$a(t+1) = a(t) \quad (36)$$

$$y^d(t) = h_t(a(t)) + v(t)) = y^M(t) + v(t). \quad (37)$$

Note here that $y^M(t)$ is the output vector of the nodes in the output layer for pattern t. The input to the NN for pattern t combined with the structure of the NN is expressed by a nonlinear time-variant function h_t. $v(t)$ is assumed to be a white noise vector with covariance matrix $R(t)$ regarded as modeling error. Given $h_s, R(s), y(s) | 1 \le s \le t$, the application of EKF gives the following algorithm [10]:

$$\hat{a}(t) = \hat{a}(t-1) + K(t)[y^d(t) - \hat{y}^M(t)], \qquad (L \times 1) \quad (38)$$

$$K(t) = P(t-1)H(t)^T[H(t)P(t-1)H(t)^T + R(t)]^{-1}, \qquad (L \times N_M) \quad (39)$$

$$P(t) = P(t-1) - K(t)H(t)P(t-1), \qquad (L \times L) \quad (40)$$

where $K(t)$ is the Kalman gain vector. $H(t)$ is expressed by

$$H(t) = \left(\frac{\partial x^M(t)}{\partial a} \right)_{a=\hat{a}(t-1)}. \quad (41)$$

In our case, $R(t)$ is assumed to be a diagonal matrix with a diagonal elements λ which is updated as follows:

$$\hat{\lambda}(t) = \hat{\lambda}(t-1) + \mu(t) \left[\frac{(y^d(t) - \hat{x}^M(t))^T (y^d(t) - \hat{x}^M(t))}{N_M} - \hat{\lambda}(t-1) \right], \qquad (1 \times 1). \quad (42)$$

However, $\hat{\lambda}(t)$ tends to zero as t tends to infinity. Thus for t greater than some chosen $T_{\max}$, it is helpful to fix $\mu(t) = 1/T_{\max}$. This ensures that the error correcting term in (42) does not tend to zero.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their useful comments and suggestions which improved the quality of this paper. They also acknowledge contribution of P. Kumar, a Postgraduate Student at Indian Institute of Technology (IIT), Kanpur, India, for his technical suggestions.

REFERENCES

[1] R. P. Lippmann, "An introduction to computing with neural networks," *IEEE Acoust. Speech, Signal Process. Mag.*, vol. 4, no. 2, pp. 4–22, Apr. 1987.

[2] K. S. Narendra and K. Parthasarathy, "Gradient methods for optimisation of dynamical systems containing neural networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 252–262, Mar. 1991.

[3] K. C. Tan and H. J. Tang, "New dynamical optimal learning for linear multilayer fnn," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1562–1568, Nov. 2004.

[4] W. Wei, F. Guorui, L. Zhengxue, and X. Yuesheng, "Deterministic convergence of an online gradient method for bp neural network," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 533–540, May 2005.

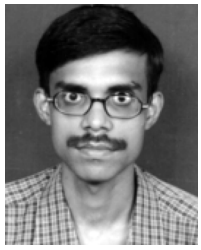
[5] X. Yu, M. O. Efe, and O. Kaynak, "A general backpropagation algorithm for feedforward neural networks learning," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 251–254, Jan. 2002.

[6] S. Haykin, *Neural Networks, A Comprehensive Foundation*, S. Haykin, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[7] D. Sarkar, "Methods to speed up error back propagation learning algorithm," *ACM Comput. Surv.*, vol. 27, no. 4, pp. 519–544, 1995.
[8] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," in *Inst. Electr. Eng. Proc.*, 1992, vol. 139, pp. 301–310.
[9] S. Osowski, P. Bojarczak, and M. Stodolski, "Fast second order learning algorithm for feedforward multilayer neural network and its applications," *Neural Netw.*, vol. 9, no. 9, pp. 1583–1596, 1996.
[10] Y. Iiguni, H. Sakai, and H. Tokumaru, "A real-time learning algorithm for a multilayered neural netwok based on extended Kalman filter," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 959–966, Apr. 1992.
[11] J. Bilski and L. Rutkowski, "A fast training algorithm for neural networks," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 6, pp. 749–753, Jun. 1998.
[12] M. T. Hagan and M. B. Mehnaj, "Training feedforward networks with Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.
[13] G. Lera and M. Pinzolas, "Neighborhood based Levenberg-Marquardt algorithm for neural network training," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1200–1203, Sep. 2002.
[14] B. M. Wilamowski, S. Iplikci, O. Kaynak, and M. O. Efe, "An algorithm for fast convergence in training neural networks," in *Proc . IEEE Int. Joint Conf. Neural Netw. (IJCNN '01)*, Jul. 15–19, 2001, vol. 3, pp. 1778–1782.
[15] A. Toledo, M. Pinzolas, J. J. Ibarrola, and G. Lera, "Improvement of the neighborhood based Levenberg-Marquardt algorithm by local adaptation of the learning coefficient," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 988–992, Jul. 2005.
[16] W. Yu, A. S. Poznyak, and X. Li, "Multilayer dynamic neural networks for non-linear system on-line identification," *Int. J. Contr.*, vol. 74, no. 18, pp. 1858–1864, 2001.
[17] L. Behera, M. Gopal, and S. Choudhury, "On adaptive trajectory tracking of a robot manipulator using inversion of its neural emulator," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1401–1414, Nov. 1996.
[18] T. P. Vogl, J. K. Mangis, A. K. zigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the backpropagation method," *Biol. Cybern.*, vol. 59, pp. 256–264, Sep. 1988.
[19] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Netw.*, vol. 1, no. 4, pp. 295–308, 1988.
[20] T. Tollenaere, "Supersab: Fast adaptive back propagation with good scaling properties," *Neural Netw.*, vol. 3, no. 5, pp. 561–573, 1990.
[21] G. Qiu, M. R. Varley, and T. J. Terrel, "Accelerated training of bp using adaptive momentum steps," *Inst. Electr. Eng. Electronics Lett.*, vol. 28, no. 4, pp. 377–379, Feb. 1992.
[22] M. Krstic and I. Kanellakapoulos*, Non Linear and Adaptive Control Design*, S. Haykin, Ed. New York: Wiley, 1995.
[23] X. H. Yu, G. A. Chen, and S. X. Cheng, "Acceleration of bp learning using optimized learning rate and momentum," *Inst. Electr. Eng. Electronics Lett.*, vol. 29, no. 14, pp. 1288–1290, Jul. 1993.
[24] C. T. Chen*, Linear System Theory and Design*, 3rd ed. New York: Oxford Univ. Press, 1999.
[25] C.-K. Li, "A sigma-pi-sigma neural network (spsnn)," *Neural Process. Lett.*, vol. 17, pp. 1–19, 2003.

Laxmidhar Behera (S'92–M'03–SM'03) was born in January 1967. He received the B.S. and M.S. degrees in electrical engineering from Regional Engineering College (REC), Rourkela, India, in 1988 and 1990, respectively, and the Ph.D. degree from Indian Institute of Technology (IIT), Delhi, India, in 1995.

He was a Lecturer at REC from 1990 to 1991 and an Assistant Professor at Birla Institute of Technology and Science (BITS), Pilani, India, from 1996 to 1999. He was a member of the faculty at the Bhakti Vedanta Institute, Mumbai, India, in 1999–2000. He worked as a Scientist at the Institute of Autonomous Intelligent System, Gesellschaft für Mathematik und Datenverarbeitung (GMD), Sankt Augustin, Germany, in 2000–2001. He is currently an Associate Professor at the Department of Electrical Engineering at IIT, Kanpur, India. His areas of interests are intelligent control, neural computation, Robotics and quantum neural networks.



Swagat Kumar was born in January 1980. He received the B.S. degree in electrical engineering from Orissa School of Mining Engineering (OSME), Keonjhar, Orissa, India, in 2001, and the M.S. degree in control system engineering from Indian Institute of Technology (IIT), Kanpur, India, in 2002. He is currently working towards the Ph.D. degree at the Department of Electrical Engineering, IIT Kanpur.

His areas of interests are nonlinear optimization, neural networks, and robotics.

Awhan Patnaik was born in 1979. He received the B.S. degree in electrical engineering from B. P. Poddar Institute of Management and Technology (BPPIMT), Kolkata, India, in 2003. He is currently working towards the Ph.D. degree at the Department of Electrical Engineering, Indian Institute of Technology (IIT), Kanpur, India.

His areas of interests are mathematical control theory and optimization.