Neural Process Lett (2011) 33:201–214
DOI 10.1007/s11063-011-9173-1

AVLR-EBP: A Variable Step Size Approach to Speed-up the Convergence of Error Back-Propagation Algorithm

Arman Didandeh · Nima Mirbakhsh · Ali Amiri · Mahmood Fathy

Published online: 22 February 2011

Abstract A critical issue of Neural Network based large-scale data mining algorithms is how to speed up their learning algorithm. This problem is particularly challenging for Error Back-Propagation (EBP) algorithm in Multi-Layered Perceptron (MLP) Neural Networks due to their significant applications in many scientific and engineering problems. In this paper, we propose an Adaptive Variable Learning Rate EBP algorithm to attack the challenging problem of reducing the convergence time in an EBP algorithm, aiming to have a high-speed convergence in comparison with standard EBP algorithm. The idea is inspired from adaptive filtering, which leaded us into two semi-similar methods of calculating the learning rate. Mathematical analysis of AVLR-EBP algorithm confirms its convergence property. The AVLR-EBP algorithm is utilized for data classification applications. Simulation results on many well-known data sets shall demonstrate that this algorithm reaches to a considerable reduction in convergence time in comparison to the standard EBP algorithm. The proposed algorithm, in classifying the IRIS, Wine, Breast Cancer, Semeion and SPECT Heart datasets shows a reduction of the learning epochs relative to the standard EBP algorithm.

Keywords Neural networks · MLP · EBP · Algorithm · Classification

A. Didandeh · N. Mirbakhsh (✉)
Department of Computer Science and IT, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan, Iran
e-mail: nmirbakhsh@iasbs.ac.ir

A. Didandeh
e-mail: a_didandeh@iasbs.ac.ir

A. Amiri
Computer Engineering Group, Engineering Department, Zanjan University, Zanjan, Iran
e-mail: a_amiri@iust.ac.ir

M. Fathy
Computer Engineering Department, Iran University of Science and Technology, Tehran, Iran
e-mail: mahfathy@iust.ac.ir

1 Introduction

It is obvious that Multi-Layered Perceptron (MLP) is one of the most important machine learning tools nowadays. It is a method that uses Gradient-based algorithms and specifically the Error Back-Propagation (EBP) algorithm [5], aims to amend its error in order to move towards a better learning.

In general, the process of learning in the EBP algorithm is described as a step-by-step and iterative weight correction, in the negative gradient of Mean-Squared Errors (MSE) function. The error function in the MSE is calculated as the difference between the actual value and the returning value from the output layer. This error is back-propagated to the neurons of the preceding layers.

The performance of a learnt model based on MLP using the EBP algorithm depends on several parameters including the initializations, the learning rates selection, the topology of the network, the number of hidden layers, the amount of training data, etc.

Many researches are executed for increasing the convergence speed of EBP algorithm in MLP Neural Network see [2,3,8,7,13,12,14]. In [2,3] Abid and Fnaiech summarize the approaches for increasing the convergence speed of EBP onto seven cases including the weight updating procedure, the optimization criterion choice, the use of adaptive parameters, estimation of optimal initial conditions, pre-processing the problem before using MLP, optimization of MLP structure and the use of more advanced algorithms. Also they changed the error function and used mean least fourth to make his algorithm better.

In this paper we concentrate on the dynamic learning rate of the learning approach in order to update the weights of the networks similar to what was implemented in [7,8,12]. Thus we implemented a Variable Step Size (VSS) method to increase the convergence acceleration of the algorithm by reducing the learning epochs. In addition to demonstration of the convergence in a mathematical proof, several experimentations on the famous and most popular datasets from UCI Machine Learning Repository [4] showed that the convergence has a distinct increase in speed.

We would also like the readers of this paper to take *"Adaptive Variable Learning Rate"* and *"Variable Step Size"* to point to the same concept, whereas the former comes from adaptive filtering territory and the latter is familiar among neural network researchers.

This paper is organized as follows: In Sect. 2 and 3, we will have an overview on Multilayer Neural Networks and our approach. This includes a slight study of the *SVSS* and the *GNGD* methods. Then in Sect. 3, we will mathematically prove that our approach converges. Experimental results will follow these matters in Sect. 4 and we will conclude the paper in Sect. 5.

2 Multilayer Neural Networks

Multilayer neural networks are a powerful algorithms to model complex functions especially the ones that are linearly inseparable. A typical architecture of MLP is illustrated in Fig. 2 showing a 3 layered network of perceptrons including an input layer, one hidden layer and an output layer. This is also called feed-forward because data is fed into the input layer and is moved through the output layer for prediction. Learning a multilayer neural network is similar to a single perceptron learning approach. The error of MLP's predicted output is propagated to the previous layers by an EBP algorithm. The error for j^{th} data vector including input vector $X^j = [x_{j1},\ x_{j2},\ \ldots,\ x_{jn}]^T$ and target output d_j on the last layer is computed as the difference between the output result and data target output:

$$e_O^j = d_j - y_O^j$$

The propagated error for r^{th} node of hidden layer are calculated as below:

$$e_r^j = \sum_{k=1}^{K} w_{rk} e_k^j,$$

where K is the number of nodes in the next layer. The weights among MLP's neurons are usually randomly initialized and are improved during the learning phase. This improvements perform as below for the weight between node r and s in iteration n^{th}:

$$w_{rs}(n+1) = w_{rs}(n) + \mu(n)\, y_r(n)\, e_s(n),$$

where $y_r(n)$ is the r^{th} nod's output value and $\mu(n)$ is a small value called learning rate or step size. The process will be iterated until the MLP's error is greater than a specific predefined value (E_{max}).

3 Approach Overview

If we model the transfigurations of the error function of a MLP network during the learning phase, we can watch its sharp changes of value against time. In order to soften these sharp changes, we use a method for adaptive learning rate, or so-called VSS. This can be shown as:

$$w_{ji}(n) = -\mu^*(n) \frac{\partial \xi(n)}{\partial v_{j(n)}} y_i(n) \tag{1}$$

We have studied almost any standard version of MLP and a wide variety of adaptive filtering methods that use a VSS. The result of this study has been the election of two semi-similar adaptive learning rates, and a comparison between these two methods and the standard MLP method, which was the constant learning rate. We were motivated to go through these two VSS methods in parallel and then merge the results in this paper. So we call our general approach as ***AVLR-EBP*** which stands for "Adaptive Variable Learning Rate Error Back Propagation" and deal with the details of the elected methods in coming subsections, which are called the ***SVSS-EBP*** and the ***GNGD-EBP*** algorithms.

An applicable remark on this adaptive kind of learning rate was studied in [1,9,10]. In this notice, the dynamic learning rate is posed to have lower and upper bounds, in order to control the sequence of affects on the learning phase. We also use this notification in both methods as below:

$$\mu^*(n) = \begin{cases} \mu_{\max} & if(\mu^*(n) > \mu_{\max}) \\ \mu_{\min} & if(\mu^*(n) < \mu_{\min}) \quad (ii) \\ \mu^*(n) & otherwise \end{cases} \tag{2}$$

The following subsections will deal with these two elected methods of VSS.

3.1 Method 1- Simple Variable Step Size Algorithm (SVSS) for Error Back Propagation

The idea of the SVSS method comes from [9]. As Kwong and Johnston address in this paper (with a slight change to the original paper): "The LMS type adaptive algorithm is a gradient search algorithm which computes a set of weights that seeking to minimize the error of

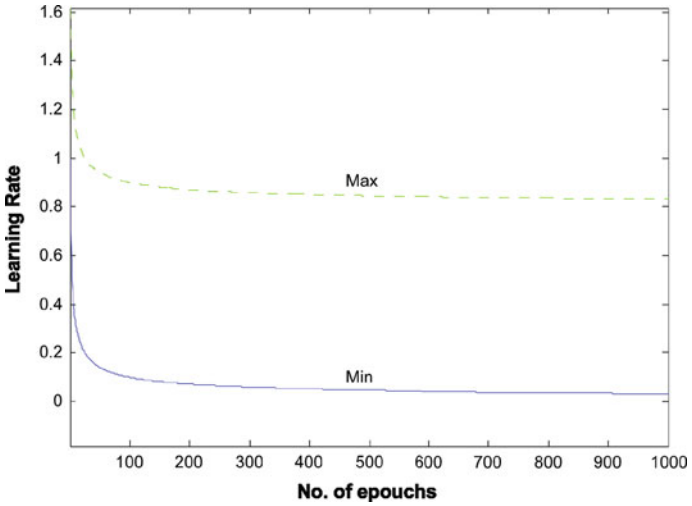

Fig. 1 The adaptive upper and lower bound for the AVLR

system, which is defined as the squared expected value of the difference between the desired target value and the value that the system has calculated for the input data." This leads us to have the merged algorithms named as ***SVSS-EBP***, using the coming formulas to implement, as a modified version of EBP algorithm.

Kwong and Johnston applied their idea for improving the standard LMS algorithm. We have taken their idea and applied it on EBP algorithm.

The adaptive learning rate—as told before—is going to have upper and lower bounds. We will first take μ_0 equal to $\mu_{\max}$ and with time going on and on, we will revise the learning rate due to the system error modifications. This will make a great outcome according to the experiments we have done. The experimental result in the coming sections will show this outcome in comparison with the standard ***EBP*** algorithm. The following formula represents the mathematics of this method:

$$\mu^*(n+1) = \alpha\mu^*(n) + \gamma e_p^2(n) \quad [0 < \alpha < 1,\ \gamma > 0] \tag{3}$$

With full respect to the original paper, we took the weights of the above formula as below:

$$\alpha = 0.97 \quad \gamma = 4.8 * 10^{-4} \tag{4}$$

We also made a slight change to the original idea by applying an idea of varying bounds for each step of the modification, which helped us in the improvement goal. This was done by applying a sluggish move on the upper and the lower bound of adaptive learning rates as below:

$$\mu_{\max}(n+1) = \frac{\mu_{max}(n)}{n^\sigma} \quad \mu_{\min}(n+1) = \frac{\mu_{min}(n)}{n^\sigma} \tag{5}$$

By taking $\sigma = 0.1$, the reduction move of our bounds was sluggish enough towards zero, hence making a good effect on the convergence time of the MLP network.

Figure 1 shall give a clue on what kind of modification may this idea make on the adaptive learning rate election during time:

Another notice should be mentioned on that the initial μ_{min} value is better to be taken equal to the value that the static learning rate of the system is suggested to take by experts.

3.2 Method 2—Generalized Normalized Gradient Descent Algorithm (GNGD) for EBP

This method originates its basis from [6]. Here, the learning rate is calculated for any step of learning phase to have the modification of weights softer than before. To do such a modification, we try to implement a series of rules to utilize an *Estimated Posterior Error* function based on step size and to generate ***GNGD-EBP*** as a modified version of EBP algorithm. what comes, we represent the main idea of an *Estimated Posteriori Error*.

The most certain modification of all in the EBP algorithm is here done on the error calculation phase. As told before, an estimated posteriori error is used in the update phase of our EBP algorithm. Besides, we will use a dynamic learning rate. These two modifications are being shown in the formula below:

$$w\,(n+1) \;=\; w\,(n) \;+\; \mu^{*}\,(n)\,y\,(n)\,e_{p}\,(n) \tag{6}$$

The basic point of our modifications is the calculation of these two parameters. In standard LMS weight update approach from which we borrow the idea of our hybrid approach, a set of most popular ways are dynamic time-varying approaches. Various forms of this dynamic learning rate are discussed in several papers and letters. We use mainly the idea discussed in [6] here. This is represented in the following formulas:

$$\mu^{*}\,(n+1) \;=\; \mu(n+1)/(\|\;y\,(n)\;\|^{2} \;+\delta) \tag{7}$$

$$\mu\,(n+1) \;=\; \alpha\mu\,(n) + \gamma e_{p}^{2}\,(n) \quad [0<\alpha<1,\;\gamma>0] \tag{8}$$

$$e_{p}\,(n) \;=\; e(n)/(1+\mu\,(n)\;\|\;y\,(n)\;\|^{2}) \tag{9}$$

There could be another add-on to this algorithm which was again discussed in [6], in which the parameter δ could be also taken in a dynamic mode. This is implementable using the below formulas:

$$\delta\,(n\;+\;1) \;=\;\; \delta\,(n) - \rho\nabla_{\delta(n-1)}E(n) \tag{10}$$

$$E\,(n) = e_{p}(n)^{2} \tag{11}$$

In the coming section, we try to give a proof of convergence over our approach. This approach is generalized to contain both methods which are dealt in the previous sections.

4 Convergence Proof

As illustrated in Fig. 2, we discuss the architecture of our MLP Neural Network as below:

- Input vector: $X^{j} \;=\; [x_{j1},\; x_{j2},\; \ldots,\; x_{jn}]^{T}$
- Target output: d_{j}
- Weight set: $W^{\dagger} \;=\; \{W_{O},\; W_{1},\; \ldots,\; W_{L}\}$
- Output layer weights vector: $W_{O} \;=\; [w_{O1}, \ldots,\; w_{On}]$
- Hidden layer weights vector:

$$W_{i} \;=\; [w_{i1},\; \ldots,\; w_{in}]\;\; for\; i \;=\; 1, \ldots, L \tag{12}$$

For each layer, we consider an activation function (also called a green function). To represent these functions, we note them as below:

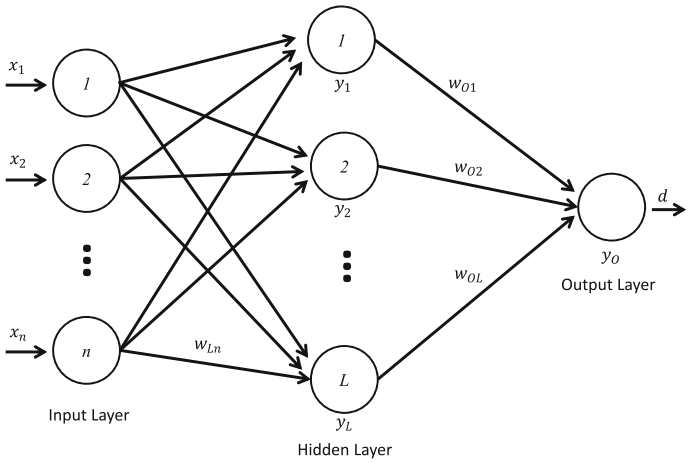

Fig. 2 A simple three-layer MLP

– Hidden layer activation function:

$$\Phi(x) = [\varphi(x_1), \ldots, \varphi(x_L)] \tag{13}$$

– Output layer activation function: φ

This activation function is used to calculate the output value of any neuron as below:

– Hidden layer output: $Y_j = [y_1, \ldots, y_L]$

$$y_i = \Phi(\mathrm{W_i X_j}) \Rightarrow Y_j = \Phi(\mathrm{W}X^j) \tag{14}$$

– Output layer desired output:

$$y_O^j = \varphi\left(W_O Y_j\right) = \varphi\left(W_O \Phi(\mathrm{W}X^j)\right) \tag{15}$$

We define the full error of our network as a function of input set as below, where N is the number of input samples:

$$\xi = \frac{1}{2}\sum_{j=1}^{N}(d_j - y_O^j)^2 \tag{16}$$

Now we define the below function:

$$\eta_j(t) = \frac{1}{2}(d_j - \varphi(t))^2 \tag{17}$$

So we can write the full error of the network regarding (16) and (17) as below:

$$\xi = \sum_{j=1}^{N} \eta_j\left(W_O \Phi(\mathrm{W}X^j)\right) \tag{18}$$

Now we calculate the derivation of the full error to all weights:

$$\frac{\partial \xi}{\partial W^\dagger} = \xi_{W^\dagger} = \xi_{W_O}, \xi_{W_1}, \ldots, \xi_{W_L} \tag{19}$$

So we have for hidden layers:

$$\xi_{W_i} = \sum_{j=1}^{N} \eta_j^{'} \left(W_O \Phi(\mathrm{W} X^j) \right) X^j \Phi^{'} \left(\mathrm{W_i} X^j \right) w_{Oi} \tag{20}$$

And for output layer:

$$\xi_{W_O} = \sum_{j=1}^{N} \eta_j^{'} \left(W_O \Phi(\mathrm{W} X^j) \right) \Phi \left(\mathrm{W} X^j \right) \tag{21}$$

While we should have our weight manipulations in a way that leads to reduction of the gradient of error, and with the below assumption:

$$W^{(n+1)} = W^{(n)} + \Delta W^{(n)} \tag{22}$$

We calculate the chain of $W^{(n)}$ s with a start on $W^{(0)}$. So we will have the weight modifications as below:

$$\Delta W_i^{(n)} = -\mu(n) \frac{\partial \xi}{\partial W_i} (\textit{Hidden layer}) \tag{23}$$

$$\Delta W_i^{(n)} = -\mu(n) \frac{\partial \xi}{\partial W_O} \; for \; i = 1, \ldots, L (\textit{Output layer}) \tag{24}$$

$$\Delta W_i^{(n)} = -\mu(n) \sum_{j=1}^{N} \eta_j^{'} \left(W_O \Phi(\mathrm{W} X^j) \right) X^j \Phi^{'} \left(\mathrm{W_i} X^j \right) w_{Oi} (\textit{Hidden layer}) \tag{25}$$

$$\Delta W_i^{(n)} = -\mu(n) \sum_{j=1}^{N} \eta_j^{'} \left(W_O \Phi(\mathrm{W} X^j) \right) \Phi \left(\mathrm{W} X^j \right) \; for \; i = 1, \ldots, L (\textit{Output layer}) \tag{26}$$

From now on, we assume that for our MLP network, the following assumptions are established:

Assumption A0 $\mu(n)$ is bounded, which means that there exists μ_{min} and μ_{max} that:

$$\mu_{min}/n^{\delta} \leq \mu(n) \leq \mu_{max}/n^{\delta}; \quad 0 < \delta \leq 1 \tag{27}$$

Assumption A1 The activation functions and their first and second derivation are uniformly bounded.

Assumption A2 The points of the sequence for output layer weights are placed in a closed bounded region, or better to say:

$$\left\{ W_O^{(n)} \right\} \subset \mathrm{R}^L \tag{28}$$

We also denote the followings to simplify the matters:

$$\Delta W_i^{(n)} = W_i^{(n+1)} - W_i^{(n)} \tag{29}$$

$$\Phi^{\mathrm{n,j}} = \Phi(\mathrm{W^{(n)}} X^j) \tag{30}$$

$$\Psi^{n,j} = \Phi^{(\mathrm{n}+1),\mathrm{j}} - \Phi^{\mathrm{n,j}} \tag{31}$$

Lemma 1 *With Assumptions A0, A1 and A2 valid, there exist* $C_i > 0$ *such that:*

$$\| \Phi(x) \| \leq C_1 \quad x \in R^L \tag{32}$$

$$\| \Psi^{n,j} \|^2 \leq C_2 \sum_{i=1}^{L} \| \Delta W_i^{(n)} \|^2 \quad n = 0, 1, \ldots; j = 1, 2, \ldots, N \tag{33}$$

$$\sum_{i=0}^{L} \| \Delta W_i^{(n)} \|^2 \leq C_3 / n^{\delta} \tag{34}$$

Proof (32) and (33) are proved in [13] so we just prove (34) here. With *A1* valid and regarding to (18), $\| \eta_j(t) \|$, $\| \eta_j'(t) \|$ and $\| \eta_j''(t) \|$ are uniformly bounded. If we mix *A0* and *A2* with (22), we will have:

$$\Delta W_i^{(n)} = -\mu(n) \xi_{W_i} \Rightarrow \| \Delta W_i^{(n)} \|^2 = \| \mu(n) \xi_{W_i} \|^2 \leq \mu_{max}^2 \| \xi_{W_i} \|^2 \tag{35}$$

With (20), (21) and *A2* we know that ξ_{W_i} is bounded. We consider C_3 the upper bound for it be. So we will have:

$$\| \Delta W_i^{(n)} \|^2 \leq \mu_{max}^2 \| \xi_{W_i} \|^2 \leq \mu_{max}^2 C_3^2 \tag{36}$$

□

Lemma 2 *Assume that the series* $\sum_{n=1}^{\infty} \frac{a_n^2}{n^{\delta}}$ *converge and all* $a_n \geq 0$ *where* $\delta \in (0, 1]$ *are fixed constants. Suppose that there exists a constant* $\mu > 0$ *such that* $|a_{n+,1} - a_n| < \frac{\mu}{n^{\delta}}$, $n = 1, 2, \ldots$ *Then we have:*

$$\lim_{n \to \infty} a_n \to 0 \tag{37}$$

Lemma 3 *Let* $F : \Omega \subset R^k \to R^m$ $(k, m \geq 1)$ *be continuous* ($\Omega \subset R^k$ *is a closed bounded region) and* $\Omega_0 = \{x \in \Omega : F(x) = 0\}$ *be finite. Suppose the sequence* $\{x_n\} \subset \Omega$ *is such that* $\lim_{n \to \infty} F(x_n) = 0$ *and* $\lim_{n \to \infty} \| x_{n+1} - x_n \| = 0$. *Then there exists* $x^* \in \Omega_0$ *such that* $\lim_{n \to \infty} x_n = x^*$.

Theorem 1 *Let* ξ *defined as* (18) *be the full error function of the MLP network in which weights are being updated as* (22). *With Assumptions* **A0, A1** *and* **A2** *valid, we will have:*

$$\lim_{n \to \infty} \xi_W(W^{(n)}) = 0 \tag{38}$$

Proof For each input X^j and in the n^{th} step, using the Taylor extension formula we have:

$$\varphi\left(W_i^{(n+1)} X^j\right) = \varphi\left(W_i^{(n)} X^j\right) + \varphi'\left(W_i^{(n+1)} X^j\right)\left(\Delta W_i^{(n)} X^j\right) + \frac{1}{2}\varphi''\left(\mathrm{t}_{i,j,n}\right)(\Delta W_i^{(n)} X^j)^2, \tag{39}$$

in which:

$$\mathrm{t}_{i,j,n} \in \mathrm{R} \quad (1 \leq i \leq L) \tag{40}$$

is a number between $W_i^{(n)} * X^j$ and $W_i^{(n+1)} * X^j$.

Since:

$$W_O^{(n)} \Psi^{n,j} = \sum_{i=1}^{L} w_{O,i}^{(n)} (\Phi\left(\mathrm{W}^{(\mathrm{n}+1)} X^j\right) - \Phi(\mathrm{W}^{(\mathrm{n})} X^j)), \tag{41}$$

the combination of (22), (39) and (40) yields:

$$\sum_{j=1}^{N} \eta_j^{'} \left(W_O^{(n)} \Phi^{\mathrm{n,j}} \right) W_O^{(n)} \Psi^{\mathrm{n,j}} = -\frac{1}{\mu\,(n)} \sum_{i=1}^{L} \| \Delta W_i^{(n)} \|^2 + \gamma_1, \tag{42}$$

in which:

$$\gamma_1 = \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{N} w_{O,i} \eta_j^{'} \left(W_O^{(n)} \Phi^{\mathrm{n,j}} \right) \varphi^{''} \left(\mathrm{t}_{i,j,n} \right) (\Delta W_i^{(n)} X^j)^2 \tag{43}$$

Regarding the update formula for weights in (22) and the mean value theorem, we can infer the following:

$$\begin{aligned} &\xi \left(W^{(n+1)} \right) - \xi \left(W^{(n)} \right) \\ &= \sum_{j=1}^{N} \eta_j^{'} \left(W_O^{(n)} \Phi^{\mathrm{n,j}} \right) \left(W_O^{(n+1)} \Phi^{(\mathrm{n}+1),\mathrm{j}} - W_O^{(n)} \Phi^{\mathrm{n,j}} \right) + \gamma_2 \end{aligned} \tag{44}$$

After placing the Taylor sequence for $\Phi^{(\mathrm{n}+1),\mathrm{j}}$, we will have:

$$\begin{aligned} &\xi \left(W^{(n+1)} \right) - \xi \left(W^{(n)} \right) \\ &= \sum_{j=1}^{N} \eta_j^{'} \left(W_O^{(n)} \Phi^{\mathrm{n,j}} \right) \Phi^{\mathrm{n,j}} \Delta W_O^{(n)} + \sum_{j=1}^{N} \eta_j^{'} \left(W_O^{(n)} \Phi^{\mathrm{n,j}} \right) W_O^{(n)} \Psi^{n,j} \\ &\quad + \gamma_2 + \gamma_3 \\ &= -\frac{1}{\mu\,(n)} \sum_{i=0}^{L} \| \Delta W_i^{(n)} \|^2 + \gamma_1 + \gamma_2 + \gamma_3, \end{aligned} \tag{45}$$

where:

$$\gamma_2 = \frac{1}{2} \sum_{i=0}^{L} \eta_j^{''} \left(\mathrm{t}_{i,j,n} \right) (W_O^{(n+1)} \Phi^{(\mathrm{n}+1),\mathrm{j}} - W_O^{(n)} \Phi^{\mathrm{n,j}})^2, \tag{46}$$

in which:

$$\mathrm{t}_{i,j,n} \in \mathrm{R} \quad (1 \le i \le L) \tag{47}$$

is a number between $W_i^{(n)} \Phi^{\mathrm{n,j}}$ and $W_i^{(n+1)} \Phi^{(\mathrm{n}+1),\mathrm{j}}$, and

$$\begin{aligned} \gamma_3 = &\sum_{j=1}^{N} \eta_j^{'} \left(W_O^{(n)} \Phi^{\mathrm{n,j}} \right) \Phi^{\mathrm{n,j}} \Delta W_O^{(n)} \\ &+ \sum_{j=1}^{N} \eta_j^{'} \left(W_O^{(n)} \Phi^{\mathrm{n,j}} \right) \Psi^{\mathrm{n,j}} \Delta W_O^{(n)} \end{aligned} \tag{48}$$

We can derive this from Assumptions **A1** and **A2** that:

$$\gamma_1 \le C_4 \sum \| \Delta W_i^{(n)} \|^2, \tag{49}$$

in which:

$$\begin{aligned} C_4 \;&=\; \tfrac{N}{2}\; SUP_{n\geq 0} \parallel W_0^{(n)} \parallel SUP_{t\in \mathrm{R}} \left|\varphi^{''}\,(\mathrm{t})\right| \\ SUP_{t\in \mathrm{R}} &\left|\eta_j^{'}\,(\mathrm{t})\right| MAX_{1\leq j\leq N} \parallel X^j \parallel^2 \end{aligned} \tag{50}$$

Similarly we can estimate the terms γ_2 and γ_3. Thus there exists $C_5 \geq 0$ such that:

$$\xi\left(W^{(n+1)}\right) - \xi\left(W^{(n)}\right) \leq \left(-\frac{1}{\mu\,(n)} - C_5\right) \sum_{i=0}^{L} \parallel \Delta W_i^{(n)} \parallel^2 \tag{51}$$

With regard to Assumption ***A0***, there exists a positive integer $N_0 \in \mathrm{N}$ such that for $n \geq N_0$ we have $\frac{1}{N_0} > C_5$. So we can consider for sufficiently large values of n that:

$$\xi\left(W^{(n+1)}\right) \leq \xi\left(W^{(n)}\right) \tag{52}$$

Now if we consider another assumption that $\xi\left(W^{(n)}\right) \geq 0$ for $n = 0,\ 1,\ \ldots$ then we can have:

$$\exists \xi^* \geq 0 \ni \lim_{n\to\infty} \xi(\,W^{(n)}) = \xi^* \tag{53}$$

Due to (50) we will have:

$$\xi\left(W^{(n+1)}\right) \leq \ldots \leq \xi\left(W^{(0)}\right) - \sum_{k=0}^{n}\left(\frac{1}{\mu\,(n)} - C_5\right) \sum_{i=0}^{L} \parallel \Delta W_i^{(k)} \parallel^2 \tag{54}$$

If we set $n \to \infty$, it leads to:

$$\sum_{n=0}^{\infty}\left(\frac{1}{\mu\,(n)} - C_5\right) \sum_{i=0}^{L} \parallel \Delta W_i^{(k)} \parallel^2 \leq \xi\left(W^{(0)}\right) \leq \infty \tag{55}$$

With Assumption ***A0*** valid, and (22), we will have:

$$\begin{aligned} \sum_{n=1}^{\infty} \frac{1}{n^{\delta}} \parallel \xi_W\left(\,W^{(n)}\right) \parallel^2 &\leq \sum_{n=1}^{\infty} \frac{1}{n^{\delta}\mu\,(n)^2} \sum_{i=0}^{L} \parallel \Delta W_i^{(n)} \parallel^2 \\ &< \frac{1}{\mu_{min}} \sum_{n=1}^{\infty} \frac{1}{\mu\,(n)} \sum_{i=0}^{L} \parallel \Delta W_i^{(n)} \parallel^2 < \infty \end{aligned} \tag{56}$$

Using the mean value rule for $\eta_j^{'}$ and $\varphi^{'}\,(\mathrm{t})$, and due to ***A1*** and ***A2***, and also using (34) in ***Lemma 1*** , we will gain:

$$\begin{aligned} |\parallel \xi_W\left(W^{(n+1)}\right) \parallel - \parallel \xi_W\left(W^{(n)}\right) \parallel| &\leq \parallel \xi_W\left(W^{(n+1)}\right) \parallel - \parallel \xi_W\left(W^{(n)}\right) \parallel \\ &\leq C_6 \sum_{i=0}^{L} \parallel \Delta W_i^{(n)} \parallel < C_6\, C_3\, / n^{\delta} \end{aligned} \tag{57}$$

So ***(Theorem)*** is provable by combining (56), (57) and ***Lemma 2.***:

$$\lim_{n\to\infty} \xi_W(\,W^{(n)}) = 0 \tag{58}$$

Table 1 Convergence comparison among three algorithm

Data set	layers	*EBP*	*SVSS-EBP*	*GNGD-EBP*
Wine	4 8	501	454	268
Wine	5 5	367	360	285
Wine	6 4	477	487	311
Breast cancer	4 8	445	364	292
Breast cancer	5 5	435	332	313
Breast cancer	6 4	539	415	212
Iris	4 8	768	788	292
Iris	5 5	769	591	288
Iris	6 4	1189	667	428
Semeion	4 8	170	143	62
Semeion	5 5	338	301	268
Semeion	6 4	315	214	169
SPECT heart	4 8	498	239	211
SPECT heart	5 5	170	159	161
SPECT heart	6 4	212	178	233

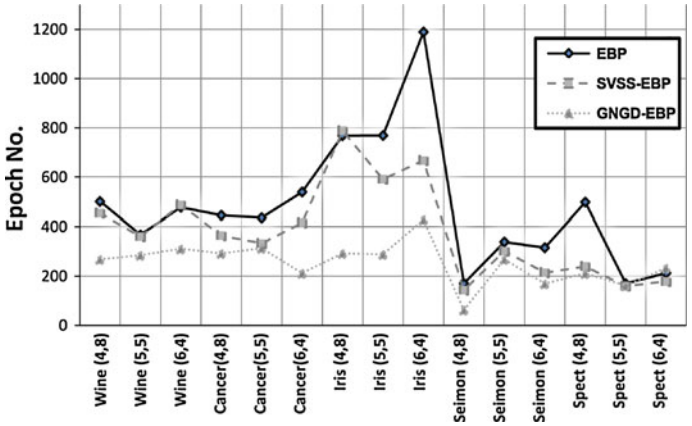

Fig. 3 Convergence comparison among three algorithm

In view of (34), (58) and

$$\frac{\partial \xi}{\partial W^{\dagger}} = \xi_{W^{\dagger}} = [\xi_{W_O}, \xi_{W_1}, \ldots, \xi_{W_L}], \tag{59}$$

we have:

$$\lim_{n\to\infty} \| \Delta W_O^{(n)} \|, \qquad \lim_{n\to\infty} \xi_{W_O}(W^{(n)}) = 0 \tag{60}$$

Notice that the error function is twice differentiable. With $x = w_o$ and $f(x) = \xi_{W_O}(W)$ in ***Lemma 3***, the finiteness of Φ_O with (60) and ***Lemma 3*** leads to strong convergence of ***(Theorem)***.

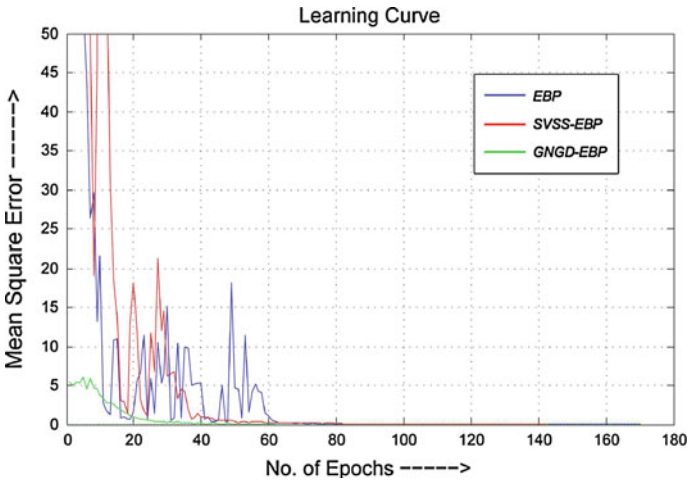

Fig. 4 The result of implementing tree algorithms to the Semeion dataset in 4–8 topology

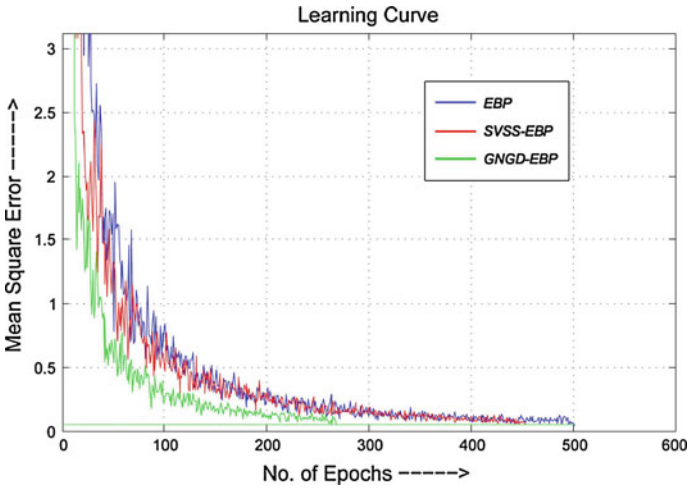

Fig. 5 A part of the result of implementing tree algorithms to the Wine dataset in 4–8 topology

5 Experimental Results

In this section, we will bring some pieces of experimental results using *MATLAB* over the datasets that were put under our *AVLR-EBP* algorithm. In order to compare the convergence of this algorithm to the previous standard one, we have use the five famous and most popular datasets from UCI Machine Learning Repository [4]:

– "Iris" containing 3 classes of 50 instances each, where each class refers to a type of iris plant.
– "Wine" that these data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

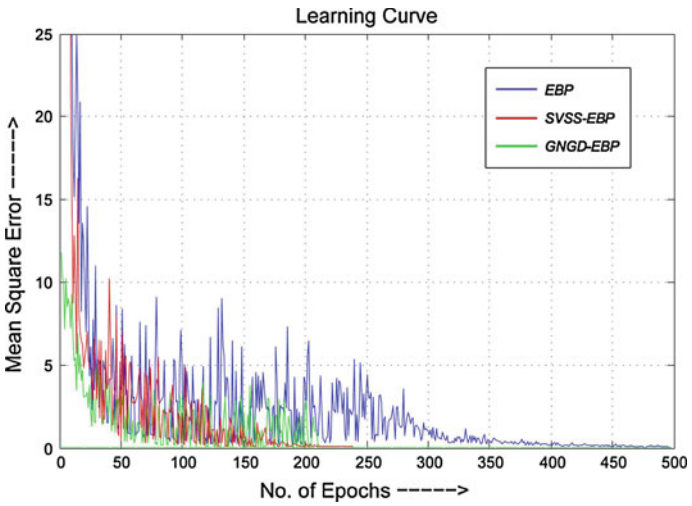

Fig. 6 The result of implementing tree algorithms to the SPECT Heart dataset in 4–8 topology

– "Breast Cancer Wisconsin" that the features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.
– "SPECT Heart", The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images.
– "Semeion Handwritten Digit Data Set" [11] includes 1593 handwritten digits from around 80 persons were scanned, stretched in a rectangular box 16 × 16 in a gray scale of 256 values.

We setup our experiments in three topologies for our 4-layered MLP network, except the input and output layers including a 4–6, 5–5 and 8–4 for number of neurons in hidden layers.

As illustrated in Table 1, SVSS-EBP and specially GNGD-EBP have obviously better results in comparison with an ordinary EBP algorithm. Also these results are illustrated in Fig. 3.

The convergence graph for the algorithmes for Seimeion, Wine and SPECT Heart is illustrated in Figs. 4, 5, 6. As it can be seen, the modifications have made some great impacts in the convergence of the EBP learning algorithm.

6 Conclusions

In this paper, we have proposed a hybrid method of generating MLP Neural Networks with the idea originated from adaptive filtering. The idea was about using an adaptive learning rate (also called as *VSS*) in the weight update formula of the *EBP* in order to control the variance of weight adjustment and to lead the adaptation of network weights into a passage that reduced the convergence time and speeded up the learning algorithm. This problem is crucially important while *MLP* is widely utilized in many scientific and engineering problems of nowadays research. The *AVLR-EBP* approach was discussed using two semi-similar methods called as *SVSS-EBP* and *GNGD-EBP* which have their ideas originated from certain previous works in adaptive filtering realm. The mathematical notice on this approach shows the substantial lessening in epoch count in comparison to the standard *EBP* algorithm. Experimental results

are driven over some popular datasets, the IRIS ,Wine, Breast Cancer, Semeion and SPECT Heart datasets, for each of the methods of our approach and were represented.

References

1. Amiri A, Fathy M, Amintoosi M, Sadoghi-Yazdi H (2007) Modified quantized input variable step size LMS, QX-VSS LMS algorithm applied to signal prediction. Proceeidings of 4th IEEE GCC Conference, pp 162–168
2. Abid S, Fnaiech F, Jervis BW, Cheriet M (2005) Fast training of multilayer perceptrons with a mixed norm algorithm. IEEE International Joint Conference on Neural Networks Proceedings, pp 1018–1022
3. Fnaiech F, Abid S (2008) Fast training of multilayer perceptrons with least mean fourth (LMF) algorithm. Int J Soft Comput 3:359–367
4. Frank A, Asuncion A (2010) UCI machine learning repository. [http://archive.ics.uci.edu/ml]. University of California, School of Information and Computer Science, Irvine, CA
5. Haykin S (1998) Neural networks, a comprehensive foundation, 2nd edn. Prentice Hall International Inc., McMaster University, Hamilton, Ontario, Canada
6. Hur M, Choi JY, Baek J-S, Seo JS (2008) Generalized normalized gradient descent algorithm based on estimated a posteriori error. Proceeding of the 10th International Conference on Advanced Communication Technology, pp 23–26
7. Magoulas GD, Vrahatis MN, Androulakis GS (1999) Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. Neural Comput 11:1769–1796
8. Plagianakos VP, Vrahatis MN, Magoulas GD (1999) Nonmonotone methods for backpropagation training with adaptive learning r. International Joint Conference on Neural Networks, pp 1762–1767
9. Reymond HK, Edward WJ (1992) A variable step size LMS algorithm. IEEE Transaction on Signal Processing 40(7):1633–1642
10. Sadoghi-Yazdi H, Lotfizad M, Fathy M (2006) Car tracking by quantized input LMS, QX-LMS algorithm in traffic scenes. IEE Proceeding on Vision, Image and Signal Processing 153(1):37–45
11. Semeion research center of sciences of communication, via Sersale 117, 00128 Rome, Italy Tattile Via Gaetano Donizetti, 1-3-5,25030 Mairano (Brescia), Italy
12. Shao H, Wu W (2006) Convergence of BP algorithm with variable learning rates for FNN training. Proceedings of the 5th Mexican international Conference on Artificial intelligence, pp 245–252
13. Wu W, Feng G, Li X (2002) Training multilayer perceptrons via minimization of sum of ridge functions. Adv Comput Math 17:331–347
14. Zhao X-Y, Lai K-S, Dai D-M (2007) An improved BP algorithm and its application in classification of surface defects of steel plate. Int J Iron Steel Res 14(2):52–55