Convergence of Hybrid Algorithm with Adaptive Learning Parameter for Multilayer Neural Network

Fadwa DAMAK, Mounir BEN NASR, Mohamed CHTOUROU
Department of Electrical Engineering
ENIS
Sfax, Tunisia
{fadwa_damak, nasr_mounir}@yahoo.fr, Mohamed.Chtourou@enis.rnu.tn

ABSTRACT: A new learning algorithm suited for training multilayered neural network is proposed that we have named hybrid is hereby introduced. With this algorithm the weights of the hidden layer are adjusted using the Kohonen algorithm. While the weights of the output layer are trained using a gradient descent method with adaptive learning parameter based Lyapunov function. The effectiveness of the proposed approach is shown by the simulation results.

Keywords: Feedforward, Neural Network, Hybrid Training, Adaptive Learning Rate, Lyapunov Theory

Received: 1 March 2013, Revised 18 April 2013, Accepted 27 April 2013

© 2013 DLINE. All rights reserved

1. Introduction

A multilayer neural network trained with the backpropagation (BP) algorithm has been successfully applied to solve diverse problems [1]. The principal disadvantage behind this algorithm is its slow convergence. Therefore, several algorithms are proposed in order to study the convergence performance of the learning network layers [2] [3] [4] [5] [6] [8] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]. Many other algorithm with the emphasis on hybrid techniques have been developed to accelerate the training method [21] [22] [7] [9] [10] and [24].

The approach proposed here is inspired by previously proposed approach [10]. In [10], the authors proposed a hybrid training algorithm for multilayer neural network which combines unsupervised and supervised learning. They used a Kohonen algorithm with a bubble neighborhood function for training the weights between input and hidden layers. The weights between hidden and output layers are adjusted by a gradient descent method.

The main contribution of this work is based on modified gradient descent method by replacing fixed learning parameter with adaptive learning parameter using Lyapunov function (LFI) described by Behera and al., [9] [12].

This work is organized in the following structures: the proposed training method is presented in section 2. Section 3 describes the simulation results and comparison between other well known algorithms. Finally, in section 4 we present the conclusions.

2. Proposed Learning Algorithm

We consider a multilayer neural network with three layers. An input layer with n neurons, a hidden layer with p neurons and an
output layer with m neurons as show in figure 1.

![Feedforward neural network diagram](image)

Figure 1. Feedforward neural network

$E$ is the criterion to be minimized defined by the equation (1).

$$E = \sum_{k=1}^{m} \frac{1}{2} [y_k^d - y_k]^2$$  \hspace{1cm} (1)

Where $y_k^d$ and $y_k$ are the desired and actual outputs.

The activation function chosen by the most researchers is the sigmoid function which is presented by the equation (2).

$$f(x) = \frac{1}{1 + e^{-x}}$$  \hspace{1cm} (2)

The proposed algorithm is described as follows: After initializing the weight vectors with various small random values, every input vector is submitted to the neural network by selecting the nearest Euclidean similarity measure [23].

$$c = \text{argmin}_i \{ ||X - W_i(t)|| \}, i \in \{1...n\}$$  \hspace{1cm} (3)

This operator $|| \cdot ||$ defines the Euclidean distance between two points in a space of dimension $n$.

Thereafter update the weights of both the winner neuron and its neighbors as:

$$W_{ij}(t+1) = \begin{cases} W_{ij}(t) + \alpha(t) h_{ci}(t) [X(t) \cdot W_i(t)]; i \in V_c(t) \\ W_{ij}(t) \quad ; i \notin V_c(t) \end{cases}$$  \hspace{1cm} (4)

Where $V_c(t)$ defines the topological neighborhood around the winner neuron and allows defining the area update of the weight vectors when the winner neuron is found. The neighborhood is generally great for the first few iterations and decreases.

$\alpha(t)$ is a learning rate between 0 and 1. The value of $\alpha(t)$ is generally high enough for the first few iterations and decreases gradually to 0.

The neighborhood function defined as:

$$h_{ci}(t) = \begin{cases} 1 \quad ||r_i - r_c|| \leq \sigma \\ 0 \quad \text{otherwise} \end{cases}$$  \hspace{1cm} (5)

Where $\sigma$ is a parameter defining the radius of the neighborhood function. $r_i$ and $r_c$ are the positions of a neuron $i$ and a winner neuron $c$.

The weights leaving from the winner neuron and its neighbors are adjusted by gradient descent algorithm with adaptive
learning rate using Lyapunov function [9].

\[ W_{kj}(t+1) = W_{kj}(t) - \eta(t) \frac{\partial E(w)}{\partial w_{kj}} \]  
(6)

Where

\[ \eta(t) = \mu \frac{\| \hat{y} \|^2}{\| J^T \hat{y} \|^2} = \mu \frac{\| e \|^2}{\| f(a_j) \cdot O_j \cdot e \|^2} \]  
(7)

Here \( \mu \) is a constant which is selected heuristically. We can add a very small constant to the denominator of the previous equation to avoid numerical instability when error \( \hat{y} \) goes to zero.

\( a_j \): Activation of neuron \( j \) in the hidden layer and \( O_j \) is the output of \( j^{th} \) neuron in the hidden layer.

\[ \hat{y} = y^d_k - y_k = e \]

And

\[ J^T = \frac{\partial y}{\partial W_{kj}} \]

Therefore

\[ \Delta W_{kj} = -\eta(t) \cdot \frac{\partial E(W)}{\partial W_{kj}} \]

So the law of updating the weights of the output layer is \( W_{kj}(t+1) = \)

\[ W_{kj}(t) - \eta(t) \cdot (y^d_k - y_k) \cdot y_k (1 - y_k) \cdot O_j \]  
(8)

We repeat the previous steps until the criterion is less than the desired minimum error or the number iterations are less than a maximum number.

3. Simulation Results

In this section, the performance of the proposed training method has been presented. Two examples are considered, approximation function and system identification. The proposed algorithm has been compared with hybrid algorithm [10], LFI [9] [12] and the BP [1].

3.1 Example 1

We consider the function given by the following expression:

\[ F(x, y) = x \cdot \sin(x) + y \cdot \cos(y) - 0.75 \]  
(9)

Where \( x \) and \( y \) are randomly chosen from the interval \([-2, 2]\).

The data set used consists of 70 observations.

Figure 2 shows the evolution of the learning criterion for the different algorithms.

As observed from table 1, we see clearly that there is an improvement ratio: the proposed algorithm is 11 times faster than BP algorithm, 4 times faster than Hybrid algorithm and 2 times faster than LF I.

The adaptive learning rates based Lyapunov function is shown in Figure 3. It can be seen that the adaptive learning rate becomes zero as the network get trained.
3.2 Example 2
The nonlinear plant of the second example is described by the difference equation.

\[ y(k+1) = \frac{y(k)}{1 + y^2(k)} + u^3(k) \]  

(10)
The model has two inputs $u(k)$ and $y(k)$ and a single output $y(k+1)$.
The input $u(k)$ is a random uniform distribution in the interval $[-2, 2]$.
A data base of 70 examples was created using (10).
Figure 4 shows the evolution of the learning criterion for the different algorithms.

![Graph showing the evolution of the criterion for the proposed algorithm versus BP, hybrid and LF I (Example 2)]

<table>
<thead>
<tr>
<th>Learning methods</th>
<th>Number of iterations</th>
<th>Training error</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>4237</td>
<td>0.1</td>
</tr>
<tr>
<td>Hybrid</td>
<td>3260</td>
<td>0.1</td>
</tr>
<tr>
<td>LF I</td>
<td>1366</td>
<td>0.1</td>
</tr>
<tr>
<td>Proposed</td>
<td>325</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 2. Comparison Among Three Algorithms for Example 2

As observed from table 2, we see clearly that there is an improvement ratio: the proposed algorithm is 13 times faster than BP algorithm, 10 times faster than Hybrid algorithm and 4 times faster than LF I.

4. Conclusion

The proposed algorithm trains multilayer neural networks by training the weights of the hidden layers using Kohonen algorithm. The weights of the output layer are trained using gradient descent method with adaptive learning rate. In the examples considered, the simulations results show that, the proposed algorithm is better compared to other algorithms in terms of speed and convergence.

References


