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

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.

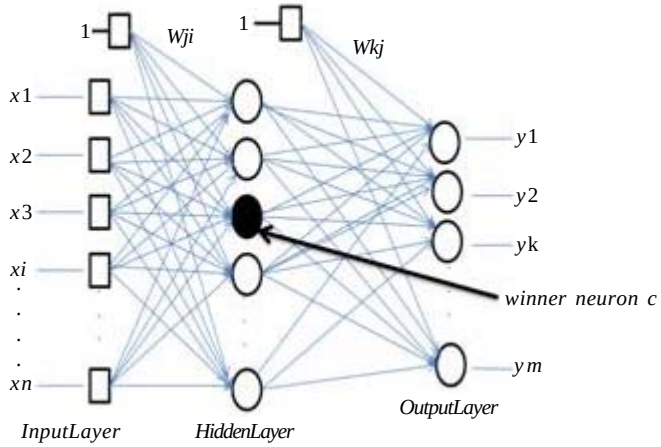

Figure 1. Feedforward neural network

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

$$E = \sum_{k=1}^{m} \frac{1}{2} [\, y_k^d - y_k]^2 \tag{1}$$

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).

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

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].

$$c = argmin_i \{ \| X - W_{ij}(t) \| \}, i \in \{1...n\} \tag{3}$$

This operator || . || 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:

$$W_{ij}(t+1) = \begin{cases} W_{ij}(t) + \alpha(t)\, h_{ci}(t)\, [X(t) - W_{ij}(t)]; i \in V_c(t) \\ W_{ij}(t) \qquad\qquad\qquad\qquad ; i \notin V_c(t) \end{cases} \tag{4}$$

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:

$$h_{ci}(t) = \begin{cases} 1 & \| r_c - r_i \| \le \sigma \\ 0 & otherwise \end{cases} \tag{5}$$

Where σ 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].

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

Where

$$\eta(t) = \mu\frac{\|\check{y}\|^2}{\|J^T\check{y}\|^2} = \mu\,.\,\frac{\|e\|^2}{\|f(a_j)\,.\,O_j\,.\,e\|^2} \tag{7}$$

Here μ 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 $\check{y}$ 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.

Where

$$\check{y} = y_k^d - y_k = e$$

And

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

Therefore

$$\Delta W_{kj} = -\eta(t)\,.\,\frac{\partial E(W)}{\partial W_{kj}}$$

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

$$W_{kj}(t) - \eta(t)\,(y_k^d - y_k)\,.\,y_k(1-y_k)\,.\,O_j \tag{8}$$

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:

$$F(x,y) = x\,.\,sin\,(x) + y.cos\,(y) - 0.75 \tag{9}$$

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.

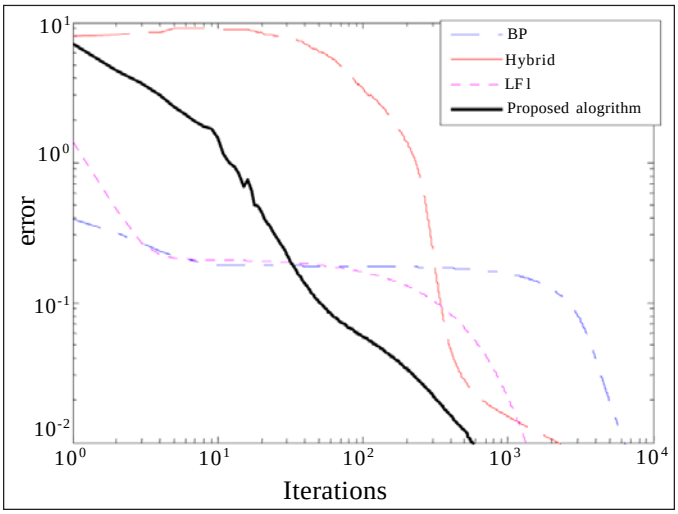

Figure 2. The evolution of the criterion for the proposed algorithm versus BP, hybrid and LF I (Example 1)

Learning methods	Number of iterations	Training error
BP	6328	0.01
Hybrid	2193	0.01
LF I	1332	0.01
Proposed	574	0.01

Table 1. Comparison Among Three Algorithm for Example 1

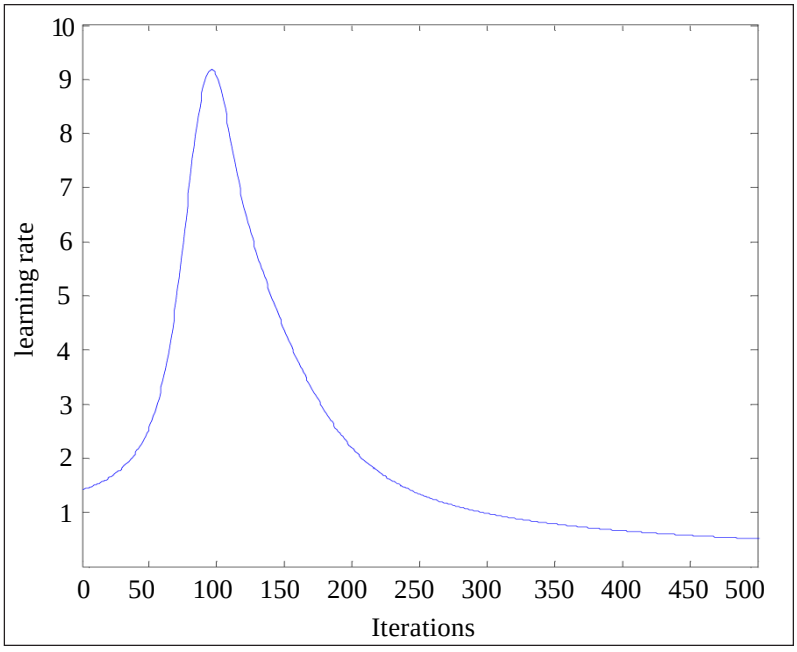

Figure 3. Adaptive learning rate (example 1)

3.2 Example 2

The nonlinear plant of the second example is described by the difference equation.

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \qquad (10)$$

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.

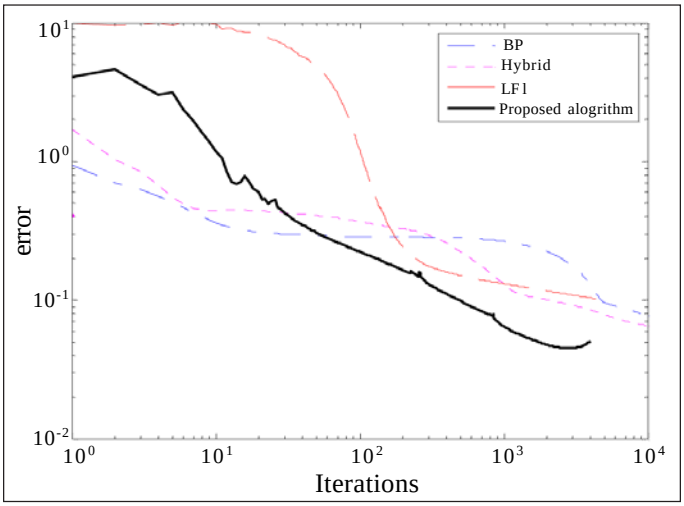

Figure 4. The evolution of the criterion for the proposed algorithm versus BP, hybrid and LF I (Example 2)

Learning methods	Number of iterations	Training error
BP	4237	0.1
Hybrid	3260	0.1
LF I	1366	0.1
Proposed	325	0.1

Table 2. Comparison Among Three Algorithm 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

[1] Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning Internal Representations by Error Propagation, Parallel Distributed Processing: Explorations in the Microstructures of Cognition. MIT. 1, p. 318-362.

[2] Park, D. J., June, B. E., Kim, J. H. (1992). Novel Fast Training Algorithm for Multilayer Feedforward Neural Networks, *Electronic Letters*, 28, p. 543-545.

[3] Qiu, G., Varley, M. R., Terrel, T. J. (1992). Accelerated Training of Backpropagation Networks by Using adaptive momentum Step, *Electronic Letters*, 28, p. 377-379.

[4] Chtourou, M. (1999). A Feedforward Neural Network Behaving as a Hierarchical System of Automata, *International Journal of Systems Science*, 30 (7) 689-695.

[5] Abid, S., Fnaiech, F., Najim, M. (2001). A fast Feedforward Training Algorithm Using a Modified form of the Standard Backpropagation Algorithm, *IEEE. on Trans. Neural Networks*, 12 (2) 424-430.

[6] Oscar, F. R., Deniz, E., Jose, C. P. (2003). Accelerating The Convergence Speed of Neural Networks Learning methods using least squares, *In*: Proceedings of the ESANN'2003, Belgium, p. 255-260.

[7] Liu, T. C., Li, R. K. (2005). A New ART-Counterpropagation Neural Network for Solving a Forecasting Problem, *Expert Systems with Application*, 28, p. 21-27.

[8] Zhang, N., Wu, W., Zheng, G. (2006). Convergence of Gradient Method with Momentum for Tow-Layer Feedforward Neural Networks, *IEEE. Trans. On Neural Networks*, 17 (2) 522-525.

[9] Behera, L., Kumar, S., Patnaik, A. (2006). On Adaptive Learning rate that guarantees convergence in feed forward Networks», *IEEE Trans. Neural Networks*, 17 (5) 1116-1125.

[10] Ben Nasr, M., Chtourou, M. (2006). A Hybrid training Algorithm for Feedforward Neural Networks, *Neural Processing Letters*, 24 (2) 107-117.

[11] Ben Nasr M., Chtourou M. (2009). A Fuzzy Neighborhood-based training Algorithm for Feedforward Neural Networks, *Neural Computing & Application*, 18 (2) 127-133.

[12] Kumar, T., Jeyasseli, S. (2009). Neighborhood based modified backpropagation algorithm using adaptive learning parameters for training Feedforward neural networks. *Neurocomputing*, 72, p. 3915-3921.

[13] Qiao, J. F., Zhang, Y., Han, H. G. (2008). Fast unit pruning algorithm for Feedforward Neural Network Design, *Applied Mathematics and Computation*, 205, p. 622-627.

[14] Hocenski, Z., Antunovic., Filko, D. (2010). Accelerated gradient learning algorithm for neural network weights update, *Neural Computing and Applications*, 19, p. 219-225.

[15] Gerstner, W. (2006). Laboratory of Computational Neuroscience (LCN) School of Computer and Communication Sciences, (4e edition. November 2004).

[16] Silva, F., Almeida L. (1990). Speeding-Up Backpropagation, *In*: R.Eckmiller, Advanced Neural Computers, North-Holland, Amsterdam, p.151-156.

[17] Jacobs, R. A. (1988). Increased Rates of Convergence Through Learning Rate Adaptation, *Neural Networks*, 1, p. 295-307.

[18] Parekh, R., Yang, J., Honavar, V. (2000). Constructive Neural Network Learning Algorithms for Neural Network Synthesis, *IEEE Trans. Neural Networks*., 11 (2) 436-451.

[19] Liu, D., Chang, T-S., Zhang, Y. (2002). A Constructive Algorithm for Neural Network Training, *IEEE Trans. Circuits Syst*., 49 (12) 1876-1879.

[20] Fahlman, S., Lebiere, C. (1990). The Cascade-Correlation Learning in Architecture, in Advances in Neural Information Processing System 2, Touretzky, D., Ed. SanMateo, CA: Morgan Kaufman, p. 524-532.

[21] Nickolai, S. R. (2000). The Layer-Wise Method and the Backpropagation Hybrid Approach to Learning a Feedforward Neural Network, *IEEE Trans. Neural Networks*., 11 (2) 295-305.

[22] Seghouane, A. K., Fleury, G. (2003). Apprentissage de Réseaux de Neurones à Fonctions Radiales de Base avec un jeu de Données à Entrée-Sortie Bruitées, Traitement de Signal, 20 (4) 413-421.

[23] Kohonen, T (1990). Self-Organized Maps, *In*: Proceedings in the IEEE, 78 (9) 1464-1480.

[24] Ben Nasr, M., Chtourou, M. (2011). A Self-organizing map-based initialization for hybrid training of feedforward neural networks, *Applied Soft Computing*, 11, p. 4458–4464.