Improvements to the Conventional Layer-by-Layer BP Algorithm*

Xu-Qin Li[1], Fei Han[1,2], Tat-Ming Lok[3], Michael R. Lyu[4], and Guang-Bin Huang[5]

[1] Institute of Intelligent Machines, Chinese Academic of Sciences,
PO Box 1130 Hefei Anhui, China 230031
[2] Department of Automation, University of Science and Technology of China
Hefei 230027, China
[3] Information Engineering Dept., The Chinese University of Hong Kong, Shatin,
Hong Kong
[4] Computer Science & Engineering Dept., The Chinese University of Hong Kong, Shatin,
Hong Kong
[5] School of Electrical and Electronic Engineering, Nanyang Technological university,
Singapore
{xqli, hanfei1976}@iim.ac.cn, tmlok@ie.cuhk.edu.hk,
lyu@cse.cuhk.edu.hk, EGBHuang@ntu.edu.sg

Abstract. This paper points out some drawbacks and proposes some modifications to the conventional layer-by-layer BP algorithm. In particular, we present a new perspective to the learning rate, which is to use a heuristic rule to define the learning rate so as to update the weights. Meanwhile, to pull the algorithm out of saturation area and prevent it from converging to a local minimum, a momentum term is introduced to the former algorithm. And finally the effectiveness and efficiency of the proposed method are demonstrated by two benchmark examples.

1 Introduction

The error back propagation algorithm (EBP) was a major breakthrough in neural network research [1][2][3][4][5][6]. However, the basic algorithm is too slow for most practical applications. So researchers have proposed several variations of error back propagation that provide significant speedup and make the algorithm more practical [7][8]. To accelerate the EBP algorithm, some modified error functions, which are different from popular mean-squared errors (MSE's), have been proposed [9].

In 1995, Ergezinger and Thomsen [10] proposed a layer-by-layer algorithm (LBL) which was based on a linearization of the nonlinear processing elements and the optimization of the EBP layer-by-layer. And in order to limit the introduced linearization error, a penalty term was added to the cost function. Commonly the

* This work was supported by the National Natural Science Foundation of China (Nos.60472111 and 60405002), and RGC Project No.CUHK 4170/04E, RGC Project No. CUHK4205/04E and UGC Project No.AoE/E-01/99.

D.S. Huang, X.-P. Zhang, G.-B. Huang (Eds.): ICIC 2005, Part II, LNCS 3645, pp. 189 – 198, 2005.

proposed layer-by-layer algorithms were decomposed into two parts: a linear one and a nonlinear one. The linear part of each layer was solved through the least-square errors (LSE's) or mean-squared errors (MSE's). But the nonlinear parts were different in a certain extent, with some of which have to assign the desired input to the hidden targets, while some of them not, or some of which use a heuristic rule to define the learning rate, while some of them use an optimal one to define the learning rate [11][12].

Although these methods have showed a fast convergence through decreasing the possibility to a premature saturation [13][14], sometimes they still result in some inevitable problems. They may not converge to the desired accuracy or involve huge computational complexity due to target assignments at hidden layer. Essentially, these methods were used to define the learning rate so as to adapt the weights [15].

This paper proposes a new prospective to the conventional proposed layer-by-layer method. This method tends to overcome the stalling problem of the layer-by-layer algorithm with a heuristic method. And also the momentum terms are introduced to both the output layer and the hidden layer in order to accelerate convergence when the conjugate gradient is moving in a consistent direction.

This paper is organized as follows. The following Section gives a brief review to the conventional layer-by-layer method. Section III introduces a new prospective to the learning rate, and the momentum method is also integrated into the algorithm. In Section IV, the improvement is demonstrated by two benchmark problems. Finally, Section V concludes this whole paper.

2 Layer-by-Layer BP Algorithm

We consider a single hidden-layer perceptron for the sake of simplicity. The activation functions for output layer and hidden layer are linear function and sigmoid function, respectively. The training patterns are described as $x^{(p)} = [x_1^{(p)}, x_2^{(p)}, \cdots, x_N^{(p)}]^T$ (p=1,2,…,P) with associated target vectors of output layer $t^{(p)} = [t_1^{(p)}, t_2^{(p)}, \cdots , t_{\mathrm{N}}^{(p)}]^T$ (p=1,2,…,P). So the network can be depicted as:

$$\hat{h}_j^{(p)} = \sum_{i=0}^{N} w_{ji} x_i^{(p)} \ (x_0 = 0). \tag{1}$$

$$h_j^{(p)} = \tanh(\hat{h}_j^{(p)} / 2). \tag{2}$$

$$y_k^{(p)} = \hat{y}_k^{(p)} = \sum_{j=0}^{H} v_{kj} h_j^{(p)} \ (h_0 = 0). \tag{3}$$

The weights should be optimized in order to minimize the MSE at the output layer defined as:

$$E^{out} = \frac{1}{2}\sum_{p=1}^{P}\sum_{k}^{M}(t_k^{(p)} - \sum_{j=0}^{H} v_{kj}h_j^{(p)})^2 \cdot \tag{4}$$

2.1 Optimization of the Output Layer Weights

With a fixed W and the desired output $t^{(p)}$, optimize V for minimizing the cost function E^{out} :

$$\Delta v_{kj} = \eta_k^{out}\alpha_{kj} . \tag{5}$$

$$\alpha_{kj} = -\frac{\partial E^{out}}{\partial v_{kj}} = \sum_{p=1}^{P}\hat{d}_k^{(p)}h_j^{(p)} . \tag{6}$$

$$\eta_k^{out} = \frac{\sum_{j=0}^{H}\alpha_{kj}{}^2}{\sum_{p=1}^{P}(\sum_{j=0}^{H}\alpha_{kj}h_j^{(p)})^2} = \frac{\vec{\alpha}_k^T\vec{\alpha}_k}{\vec{\alpha}_k^T C_h\vec{\alpha}_k} . \tag{7}$$

where $\hat{d}_k^{(p)} = -\partial E^{out} / \partial \hat{y}_k^{(p)} = t_k^{(p)} - \hat{y}_k^{(p)}$, $\vec{\alpha}_k = [\alpha_{k0}, \alpha_{k1} \cdots \alpha_{kH}]^T$ and $C_h = \{\sum_{p=1}^{P} h_j^{(p)} h_j^{(p)'}\}_{(H+1)\times(H+1)}$.

2.2 Assign the Hidden Targets

With the updated V, we assign the hidden targets denoted by $z_j^{(p)}$:

$$z_j^{(p)} = h_j^{(p)} + \varsigma_p\beta_j^{(p)} . \tag{8}$$

$$\beta_j^{(p)} = -\frac{\partial E^{out}}{\partial h_j^{(p)}} = \sum_{k=1}^{M}\hat{d}_k^{(p)}v_{kj} \cdot \tag{9}$$

$$\varsigma_p = \frac{\sum_{j=1}^{H}\beta_j^{(p)2}}{\sum_{k=1}^{M}(\sum_{j=1}^{H}v_{kj}\beta_j^{(p)})^2} = \frac{\vec{\beta}^{(p)T}\vec{\beta}^{(p)}}{\vec{\beta}^{(p)T}C_v\vec{\beta}^{(p)}} \cdot \tag{10}$$

where $\vec{\beta}^{(p)} = [\beta_1^{(p)} \beta_2^{(p)} \cdots \beta_H^{(p)}]^T$ and $C_v = \{\sum_{k=1}^{M} v_{kj} v_{kj}'\}_{H \times H}$, and assign $\hat{z}_j^{(p)} = f^{-1}(z_j^{(p)})$ after truncating $z_j^{(p)}$ to stay in (-1, 1).

2.3 Optimization the Hidden Layer Weights

We use the training patterns $x_i^{(p)}$ and $\hat{z}_j^{(p)}$ to define a new cost function at the hidden layer [9]:

$$E^{hid} = \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{H} (\hat{z}_j^{(p)} - \sum_{i=0}^{N} w_{ji} x_i^{(p)})^2 [f'(\hat{z}_j^{(p)})]^2 . \tag{11}$$

and optimize W for minimizing E^{hid} as follows:

$$\Delta w_{ji} = \eta_j^{hid} \gamma_{ji} . \tag{12}$$

$$\gamma_{ji} = -\frac{\partial E^{hid}}{\partial w_{ji}} = \sum_{p=1}^{P} \hat{e}_j^{(p)} [f'(\hat{z}_j^{(p)})]^2 x_i^{(p)} . \tag{13}$$

$$\eta_j^{hid} = \frac{\sum_{i=0}^{N} (\gamma_{ji})^2}{\sum_{p=1}^{P} (\sum_{i=0}^{N} \gamma_{ji} x_i^{(p)})^2 [f'(\hat{z}_j^{(p)})]^2} = \frac{\hat{\gamma}_{ji}{}^T \hat{\gamma}_{ji}}{\hat{\gamma}_{ji}{}^T C_x \hat{\gamma}_{ji}} . \tag{14}$$

where $\hat{e}_j^{(p)} = -\partial E^{hid} / \partial \hat{h}_j^{(p)} = \hat{z}_j^{(p)} - \hat{h}_j^{(p)}$ $\hat{\gamma}_{ji} = [\gamma_{j0} \gamma_{j1} \cdots \gamma_{jN}]^T$ and $C_x = \{\sum_{p=1}^{P} x_i^{(p)} x_i^{(p)'} [f'(\hat{z}_j^{(p)})]^2\}_{(N+1) \times (N+1)}$ [9].

3 Modifications to the Conventional Algorithm

We would like to make the learning rate larger at the initial stage, since then we will be taking large steps and would expect to converge faster. However, if we make the learning rate too large, the algorithm will become unstable. It is impossible for us to predict the maximum allowable learning rate for arbitrary functions, but fortunately for quadratic functions we can set an upper limit.

Considering the output layer cost function:

$$E^{out} = \frac{1}{2}\sum_{p=1}^{P}\sum_{k}^{M}(t_k^{(p)} - \sum_{j=0}^{H} v_{kj}h_j^{(p)})^2 \ . \tag{15}$$

It can be transformed to a quadratic function with respect to v_{kj} :

$$F(v_{kj}) = \frac{1}{2} v_{kj}{}^T A v_{kj} + d^T v_{kj} + c \ . \tag{16}$$

The gradient of this quadratic function is $\nabla F(v_{kj}) = Av_{kj} + d$. Then A is called Hessian matrix of this quadratic function. Using a constant learning rate α , we obtain this expression according to the steepest descent algorithm:

$$v_{kj}(epoch+1) = v_{kj}(epoch) - \alpha(A * v_{kj}(epoch) + d) \ . \tag{17}$$

Or

$$v_{kj}(epoch+1) = [I - \alpha * A]v_{kj}(epoch) - \alpha d \ . \tag{18}$$

This linear dynamic system will be stable if the eigenvalues of the matrix $[I - \alpha * A]$ are less than one in magnitude. We can express the eigenvalues of this matrix in terms of the eigenvectors of the Hessian matrix A. Suppose $\{\lambda_1, \lambda_2, \cdots \lambda_n\}$ and $\{z_1, z_2, \cdots z_n\}$ to be the eigenvalues and eigenvectors of the Hessian matrix. Then $[I - \alpha A]z_i = z_i - \alpha A z_i = z_i - \alpha \lambda_i z_i = (1 - \alpha \lambda_i) z_i$.

So the eigenvectors of $[I - \alpha * A]$ are the same as the eigenvectors of A. Similarly, the eigenvalues of $[I - \alpha * A]$ are $(1 - \alpha\lambda_i)$. Naturally, we can obtain the following expression:

$$|(1 - \alpha\lambda_i)| < 1 \ . \tag{19}$$

The eigenvalues must be positive so that the quadratic function can be guaranteed to converge to a stable minimum point. So Equ. (20) will be reduced to: $\alpha < \frac{2}{\lambda_i}$.

Since it must be true for all the eigenvalues of the Hessian matrix, we have $\alpha < \frac{2}{\lambda_{\max}}$. So the maximum stable learning rate is inversely proportional to the matrix curvature of the quadratic function ($\alpha < \frac{2}{\lambda_{\max}}$) [16][17].

In fact, the optimal learning rate always changes during different applications which makes it difficult to be set optimally. So the so-called optimal learning rate in the algorithm aforementioned does not always perform best. As long as we can find out the Hessian matrix, then calculate the eigenvalues so as to define the maximum stable learning rate, the algorithm tends to converge most quickly in the direction of the eigenvector corresponding to this largest eigenvalue.

It is well known that backpropagation with momentum updating is one of the most popular modifications to the standard algorithm. When a momentum term is added to the EBP algorithm, in which the weights change is a combination of the new steepest decent step and the previous one, the weight trajectory will be much smoother and the convergence will be faster. Intuitively, the momentum term can be also added to the layer-by-layer BP algorithm.

Taking the output layer for example:

When the momentum term is added to the algorithm, the weights are updated according to the description in literature [18].

$$v_{kj}(epoch+1) = v_{kj}(epoch) + (1-\alpha)\Delta v_{kj}(epoch) + \alpha\Delta v_{kj}(epoch-1)\,. \tag{20}$$

Similarly, the hidden layer weights are updated according to the following formula:

$$w_{ji}(epoch+1) = w_{ji}(epoch) + (1-\alpha)\Delta w_{ji}(epoch) + \alpha\Delta w_{ji}(epoch-1)\,. \tag{21}$$

4 Simulation and Results

4.1 Function Approximation

In this section, a function approximation example was trained by a 1-3-1networks with 3 nodes in the hidden layer. The input data is denoted as P= -2,-1.6,-1.2…1.2, 1.6,2, and the desired output data as T=sin (3.14*p/4), which are assigned to the input and output layer of the network respectively [16].

4.1.1 Fixed Learning Rate

Fig. 1 and 2 show the MSE for the training patterns of the two methods for this function approximation example.

It can be seen from Fig. 1 and 2 that our improved method can reduce the MSE dramatically than the previous method, and the previous method tends to reach a certain MSE and remains there for the rest iterations with little or no improvement, while the proposed algorithm can make huge progress to reduce the MSE throughout the entire training process. So our improved method can decrease the MSE to an acceptable level when the training process for the previous method traps into the saturation area.

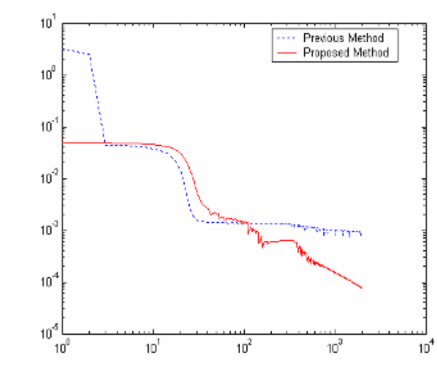

Fig. 1. The MSE cures of previous and our improved methods for a fixed learning rate (0.11) at the output layer

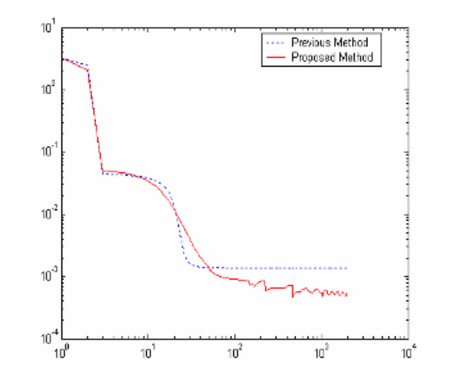

Fig. 2. The MSE cures of previous and our improved methods for a fixed learning rate (0.44) at the hidden layer

4.1.2 Incorporation of the Momentum Term

In this subsection, we further demonstrate the efficiency and effectiveness of another momentum term method. This method is derived by incorporating the momentum term, α into the weight updating formulae. Assume that the coefficient of α for the output layer and the hidden layer were set to 0.88 and 0.70, respectively.

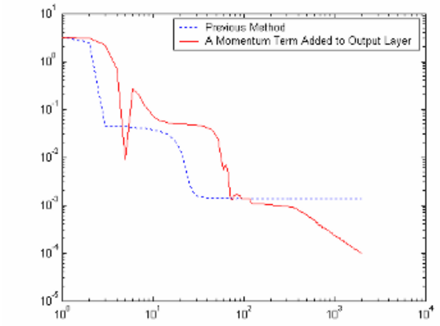

Fig. 3. Learning curves of the MSE for the previous and the output momentum term methods for a function approximation example

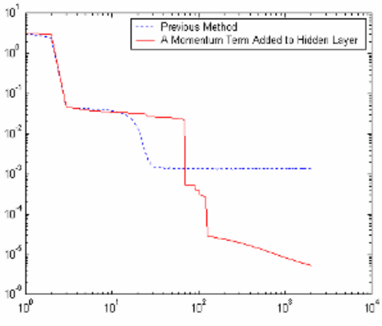

Fig. 4. Learning curves of the MSE for the previous and the hidden momentum term methods for a function approximation example

Fig. 3 and 4 illustrate the improvement of the introduction of momentum term to the original method. Although the previous method converges at the earlier stage, it stops to a local minimum after a certain number of iteration. The proposed algorithm can prevent the training process from falling into the flat regions, and make the MSE decrease dramatically until the MSE converges to a noticeable small level

4.2 XOR Problem

For an XOR problem, the network consists of two input nodes, there hidden nodes, and one output node.

4.2.1 Fixed Learning Rate

In the case of fixed learning rate, we use XOR to conduct some computer simulations. It can be found that the sum-square-errors of the previous algorithm cannot converge to an acceptable level but our proposed method can do. Fig.5 depicts a comparison of the MSEs for the standard EBP algorithm and the proposed algorithm, where the learning rate in the standard EBP method was set to 0.02 and the one for the hidden layer of the proposed method was set to 0.45.

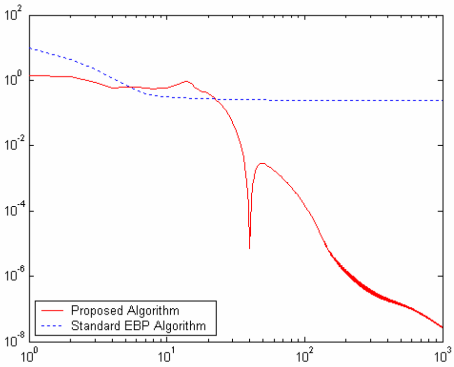

Fig. 5. Learning curves of the MSE for the standard EBP and our proposed methods for the XOR problem

We observe that the training process is easily trapped into saturation area for the standard EBP algorithm, while the proposed method of a fixed learning rate can converge over the whole learning procedure until it reaches to an acceptable level.

4.2.2 Incorporation of the Momentum Term

In addition, we also use XOR problem to verify our proposed momentum term method. Figs. 6 and 7 demonstrate the improvement of the proposed method over the standard EBP algorithm, where the coefficient of α for the output layer (Fig.6) and the hidden layer (Fig.7) were set to 0.80 and 0.60, respectively.

Fig. 6 and 7 show that our proposed method can prevent the training process falling into the flat regions, and make the MSE decrease dramatically until the MSE converges to a noticeable small level, while the standard EBP method stops to a local minimum after a certain number of iterations in spite of convergence at the early stage. So the proposed algorithm with a momentum term can meet a stringent demand in accuracy.

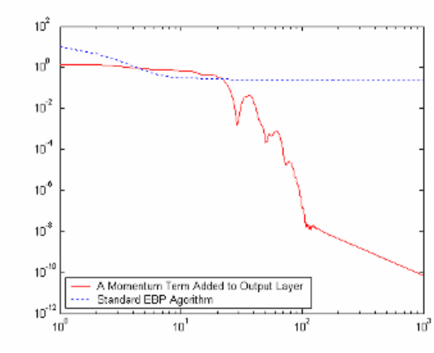

Fig. 6. Learning curves of the MSE for the momentum term and the standard EBP methods for XOR problem

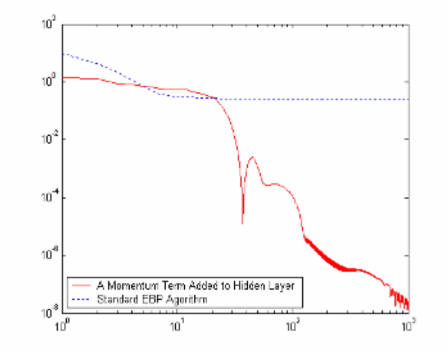

Fig. 7. Learning curves of the MSE for the momentum term and the standard EBP methods for XOR problem

5 Conclusions

This paper proposed some modifications to the conventional layer-by-layer BP algorithm so as to accelerate the learning process and reduce the possibilities to be trapped into the saturation area. To prove the efficiency and effectiveness of the proposed method, we trained a MLP network with a function approximation example and the XOR problem. In all the experiments, the proposed algorithm had demonstrated its improvement with orders of magnitude less than the previous algorithm in MSE. The modified method has also validated that the heuristic rule is sometimes better than the so-called optimal learning rate. So, much more research works about the optimal learning rate for a specific algorithm has to be done. What's more, when the momentum term is added into the algorithm, it can lead to the convergence from a local minimum to a global one, and reduce the MSE significantly. So the momentum term functions are well suitable not only to the standard BP algorithm but also to the layer-by-layer algorithm.

References

1. Huang, D.S., Horace, H.S.Ip and Zheru Chi: A Neural Root Finder of Polynomials Based on Root Moments. Neural Computation, Vol.16, No.8, pp.1721-1762, 2004
2. Huang D.S., Horace H.S.Ip, Law Ken, C. K. and Zheru Ch: Zeroing Polynomials Using Modified Constrained Neural Network Approach. IEEE Trans. On Neural Networks, vol.16, no.3, pp.721-732, 2005
3. Huang, D.S.: A Constructive Approach for Finding Arbitrary Roots of Polynomials by Neural Networks," IEEE Transactions on Neural Networks□Vol.15, No.2, pp.477-491, 2004
4. Huang, D.S.: Systematic Theory of Neural Networks for Pattern Recognition. Publishing House of Electronic Industry of China, Beijing, 1996

5. Rumelhart ,D.E. and McClelland, J.L.: Parallel Distributed Processing. Cambridge, MA: MIT Press, 1986
6. Huang, D.S., Horace, H.S.Ip and Zheru Chi: A Neural Root Finder of Polynomials Based on Root Moments.Neural Computation, Vol.16, No.8, pp.1721-1762, 2004
7. Huang, D.S.: The Local Minima Free Condition of Feedforward Neural Networks for Outer-supervised Learning. IEEE Trans on Systems, Man and Cybernetics, Vol.28B, No.3, 1998,477-480
8. Huang, D.S.: Radial Basis Probabilistic Neural Networks: Model and Application. International Journal of Pattern Recognition and Artificial Intelligence, 13(7), 1083-1101,1999
9. Sang-Hoon Oh and Soo-Young Lee: A New Error Function at Hidden Layers for Fast Training of Multilayer Perceptrons. IEEE Trans. Neural Networks, vol. 10, pp. 960–964, 1999
10. Ergezinger,S., and Thomsen,E.: An Accelerated Learning Algorithm for Multilayer Pereceptrons: Optimization Layer by Layer. IEEE Trans. Neural Networks, vol. 6, pp. 31–42, 1995
11. Wang,G.-J. and Chen,C.-C.: A fast Multilayer Neural Networks Training Algorithm Based on the Layer-by-layer Optimizing Procedures. IEEE Trans. Neural Networks, vol. 7, pp. 768–775, 1996
12. B.Ph.van Milligen, V.Tribaldos,J.A, Jimenez, and C.Santa Cruz: Comments on: An Accelerated Algorithm for Multilayer Perceptrons: Optimization Layer by Layer. IEEE Trans. Neural Networks, vol. 9, pp. 339–341, 1998
13. van Ooyen and Nienhuis, B.: Improving the Convergence of the Backpropagation Algorithm. Neural Networks, vol. 5, pp. 465–471, 1992
14. Huang, D.S.: The Bottleneck Behavior in Linear Feedforward Neural Network Classifiers and Their Breakthrough. Journal of Computer Science and Technology, Vol.14, No.1, 34-43,1999
15. Oh, S.-H.: Improving the Error Backpropagation Algorithm with a Modified Error Function. IEEE Trans. Neural Networks, vol. 8, pp.799–803, 1997
16. Martin T. Hagen and Howard B. Demuth: Neural Network Design. USA, PWS publishing company, 1996
17. Yann LeCun, Leon Botton, Genevieve, B.Orr and Klause-Robert Muller: Efficient Backprop. Neural Networks, LNCS 1524,pp.9-50,1998
18. Fredric M.Ham and Ivica Kostanic: Principles of Neuaocomputing for Science and Engineering,USA, McGraw-Hill Companies, Inc.2001