A NEW FAST NEURAL NETWORK TRAINING ALGORITHM BASED ON A MODIFIED STANDARD BACK-PROPAGATION (MBP) CRITERION MINIMIZATION

S. Abid* : *Student Member IEEE* - F. Fnaiech* - M.Najim : *Follow IEEE***
Email: Farhat.Fnaiech@esstt.rnu.tn
***E.S.S.T.T. 5 Av. Taha Hussein 1008 Tunis - TUNISIA.**
Tel (216) 1 503 837 - Fax (216) 1 391 166
****Equipe Signal & Image: Avenue du Dr. Albert Schweitzer - Domaine Universitaire. BP 99.**
33402 Talence Cedex -FRANCE, Email : najim@goelette.tsi.u-bordeaux.fr

ABSTRACT

In this work a new approach for the learning process of multilayer perceptron Neural Networks (NN) is proposed. This approach minimizes a modified form of the criterion used in the standard back-propagation algorithm (SBP) formed by the sum of the linear and the nonlinear quadratic errors of the neuron. To determine the desired target in the hidden layers an analog back-propagation strategy used in the conventional learning algorithms is developed. This permits the application of the learning procedure to all the layers. Simulation results on the 4-byte parity checker and the circle in the square problem are obtained which indicate significant reduction in the total number of iterations when compared to those of the SBP algorithm.

1. INTRODUCTION

Speeding the NN training algorithms is coming a recent subject of research [2][5][6][8][12]. The aim of this work is to present a new algorithm which is considerably faster than the SBP algorithm. The new algorithm uses a modified form of the conventional SBP algorithm, it minimizes the sum of the linear and the nonlinear squares errors for all output units and for the current pattern. To find the linear output error we compute the desired output summation by inverting the output layer nonlinearity [5][12]. This paper is organized as follows : in section 2 we shall give a brief review of the SBP algorithm where in section 3 we present the new algorithm and we shall show the modifications done on the SBP algorithm and the effect of introducing the linear error in the updating equations. Following this, in section 4, experimental results are given showing comparisons between the two algorithms and finally section 5 presents the main conclusions of the paper. An appendix is given at the end of this work which highlight the different demonstrations of the new algorithm.

2. REVIEW OF THE SBP ALGORITHM

The SBP is the most used algorithm for training multilayer NN formed by interconnections of neurons of type of Fig.1. The linear and the nonlinear actual outputs are respectively given by :

$$u_j^{[s]} = \sum_{i=0}^{n_{s-1}} w_{ji}^{[s]} y_i^{[s-1]} \tag{1}$$

$$f(u_j^{[s]}) = \frac{1}{1+e^{-au_j^{[s]}}} = y_j^{[s]} \tag{2}$$

The subscript [s] denotes the number of the corresponding layer : s=1...L. n_s is the number of neuron of the s^{th} layer.
The SBP is based on the following optimization criterion defined for the p^{th} pattern as :

$$E = \sum_{j=1}^{n_L} \frac{1}{2}\left(e_{1j}^{[L]}\right)^2 = \sum_{j=1}^{n_L} \frac{1}{2}\left(d_j^{[L]} - y_j^{[L]}\right)^2 \tag{3}$$

$d_j^{[L]}$ and $y_j^{[L]}$ are the desired and the actual outputs respectively, for the j^{th} unit (j=1...n_L).
Since the SBP algorithm is treated in the literature [1][2], we shall only remind here its main equations defined for the p^{th} pattern with the following steps :
1) Initialize randomly the weight vectors: $W_j^{[s]} = \left[w_{j0}^{[s]}, w_{j1}^{[s]}, \ldots, w_{jn_{s-1}}^{[s]}\right]^T$ s=1...L; j=1...n_s
2) Run a training pattern through the network : equations (1) and (2).
3) Evaluate the error signals :
- For the output layer [L]:

$$e_j^{[L]} = f'(u_j^{[L]}) e_{1j}^{[L]} \tag{4}$$

- For the hidden layers [s] : (s=1... L-1)

$$e_j^{[s]} = f'(u_j^{[s]}) \sum_{r=1}^{n_{s+1}} \left(e_r^{[s+1]} w_{rj}^{[s+1]}\right) \tag{5}$$

4) Update the new weight vectors $W_j^{[s]}$:

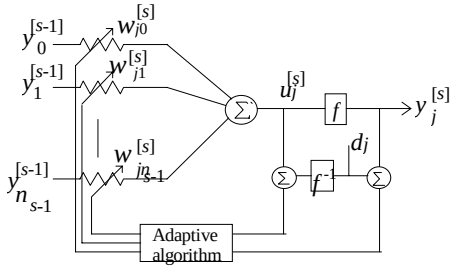


Fig.1: Implementation of the new MBP algorithm for a neuron *j* in a layer [*s*] of the NN.

$$w_{ji}^{[s]}(k+1) = w_{ji}^{[s]}(k) + \mu e_j^{[s]} y_i^{[s-1]} \tag{6}$$

μ is the learning coefficient, f' is the first derivative of f.
5) Go to step 2 if the algorithm has not converged.
In order to increase the convergence speed of the SBP, we shall use a new form of the signal error based on the linear and nonlinear error at the output of the NN.

3. NEW APPROACH

3.1. Learning of the single layer perceptron

Let us first develop the new algorithm for a neuron *j* located in any given layer [*s*] of the network. We assume that we know the desired nonlinear output of this neuron for the chosen pattern. Then the desired summation signal is directly computed by inverting the nonlinearity activation function.

Note that we have (n_{s-1}+1) inputs to this neuron (*j*). The nonlinear and linear errors are equal respectively to:

$$e_{1j}^{[s]} = d_j^{[s]} - y_j^{[s]} \tag{7}$$

$$e_{2j}^{[s]} = ld_j^{[s]} - u_j^{[s]} \tag{8}$$

Where $ld_j^{[s]}$ is the desired summation given by :

$$ld_j^{[s]} = f^{-1}(d_j^{[s]}) \tag{9}$$

Let us define now the new proposed optimization criterion *E* for the j^{th} neuron and for the p^{th} pattern :

$$E = \frac{1}{2}\left(e_{1j}^{[s]}\right)^2 + \frac{1}{2}\lambda\left(e_{2j}^{[s]}\right)^2 \tag{10}$$

By applying the gradient descent method to *E* we shall state :

$$\Delta w_{ji}^{[s]} = -\mu\frac{\partial E}{\partial w_{ji}^{[s]}} = \mu e_j^{[s]} y_i^{[s-1]} + \mu\lambda e_{2j}^{[s]} y_i^{[s-1]} \tag{11}$$

where $e_j^{[s]}$ is defined by equation (4).

Comparatively to the SBP, the new algorithm differ only by the term $\mu\lambda e_{2j}^{[s]} y_i^{[s-1]}$.

In the appendix it is shown that this algorithm converges faster than the SBP algorithm for the given nonlinearity activation function and also for a good choice of the weighting coefficient λ.

3.2. The new MBP algorithm for the multilayer NN

The new optimization criterion is given by:

$$E = \sum_{j=1}^{n_L}\frac{1}{2}\left(e_{1j}^{[L]}\right)^2 + \sum_{j=1}^{n_L}\frac{1}{2}\lambda\left(e_{2j}^{[L]}\right)^2 \tag{12}$$

we will derive now the updating equations for the output and the hidden layers .

3.2.1. Learning of the output layer by the MBP algorithm

In this layer the nonlinear outputs are known then we can compute their corresponding linear outputs, hence the linear and nonlinear errors are directly evaluated. The application of the gradient descent method to *E* gives :

$$\Delta w_{ji}^{[L]} = -\mu\frac{\partial E}{\partial w_{ji}^{[L]}} = \mu f'(u_j^{[L]}) e_{1j}^{[L]} y_i^{[L-1]} + \mu\lambda e_{2j}^{[L]} y_i^{[L-1]} \tag{13}$$

3.2.2. Learning of the hidden layers by the MBP algorithm

Since the desired linear and nonlinear outputs are unknown both of their corresponding errors should be estimated.

Let us first apply the gradient descent method to *E* for the layer [*L-1*] and then generalize the results for the other hidden layers :

$$\Delta w_{ir}^{[L-1]} = -\mu\frac{\partial E}{\partial w_{ir}^{[L-1]}} \tag{14}$$

Which leads after some calculation to :

$$\Delta w_{ir}^{[L-1]} = \mu y_r^{[L-2]} f'(u_i^{[L-1]})\left[\sum_{j=1}^{n_L} e_{1j}^{[L]} f'(u_j^{[L]}) w_{ji}^{[L]}\right] + \mu\lambda y_r^{[L-2]}\left[f'(u_i^{[L-1]})\sum_{j=1}^{n_L} e_{2j}^{[L]} w_{ji}^{[L]}\right] \tag{15}$$

By analogy to the updating equation for the output layer we can assume that the term: $\sum_{j=1}^{n_L} e_{1j}^{[L]} f'(u_j^{[L]}) w_{ji}^{[L]}$ is a nonlinear error and the term: $f'(u_i^{[L-1]})\sum_{j=1}^{n_L} e_{2j}^{[L]} w_{ji}^{[L]}$ is a linear error for this layer.

Define the nonlinear error estimation in the hidden layer [*L-1*] for the i^{th} neuron by :

$$e_{1i}^{[L-1]} = \sum_{j=1}^{n_L} e_{1j}^{[L]} f'(u_j^{[L]}) w_{ji}^{[L]} \tag{16}$$

and the linear error estimation by:

$$e_{2i}^{[L-1]} = f'(u_i^{[L-1]})\sum_{j=1}^{n_L} e_{2j}^{[L]} w_{ji}^{[L]} \tag{17}$$

Then the updating equation for the $[L-1]^{th}$ layer takes the same form of the one obtained for the output layer :

$$\Delta w_{ir}^{[L-1]} = \mu y_r^{[L-2]} f'(u_i^{[L-1]}) e_{1i}^{[L-1]} + \mu\lambda y_r^{[L-2]} e_{2i}^{[L-1]} \tag{18}$$

The procedure of derivation may be continued layer by layer. Hence, for a given layer [*s*] the updating equation (18) becomes:

$$\Delta w_{ji}^{[s]} = \mu y_i^{[s-1]} f'(u_j^{[s]}) e_{1j}^{[s]} + \mu\lambda y_i^{[s-1]} e_{2j}^{[s]}. \tag{19}$$

where :

$$e_{1j}^{[s]} = \sum_{r=1}^{n_{s+1}} e_{1r}^{[s+1]} f'(u_r^{[s+1]}) w_{rj}^{[s]} \tag{20}$$

and :

$$e_{2j}^{[s]} = f'(u_j^{[s]})\sum_{r=1}^{n_{s+1}} e_{2r}^{[s+1]} w_{rj}^{[s+1]} \tag{21}$$

then the learning will be faster than in the SBP algorithm.

In the sequel we summarize the new MBP algorithm.

Step 1 - Initialization:

* From layer s=1 to L, equalize all $y_0^{[s-1]}$ to a values different from 0, (0.5 for example).

* Randomize all the weights $w_{ji}^{[s]}$ at random values.

* Choose the weighting value λ or an initial value $\lambda(0)$.

Step 2 - Select training pattern:

Select an Input/ Output pair to be processed into the network.

The input vector is $y^{[0]}$ and its corresponding output is $d^{[L]}$

Step 3 :

Run a selected pattern through the network for each layer [s], (s = 1 ,..., L) and for each node (j) in this layer by calculating:

- The summation outputs: equation (1).
- The nonlinear outputs: equation (2).

Step 4 - Error signals:

* Calculate for the output layer (L) and for the p^{th} pattern:
- The desired summations : equation (9).
- The output errors: equation (7).
- The linear output errors: equation (8).

* For the hidden layers : s =1 to L-1
- Evaluate the nonlinear estimation errors: equation (20).
-Evaluate the linear estimation errors : equation (21).

Step 5 -Updating the synaptic coefficients :

For any node (j) of the layer s=1 to L modify the synaptic coefficients : equation (19).

Step 6 - Test for ending the running:

If the convergence condition is not verified, go back to Step2.

3.3. Comparison of the computation complexity

Table(3) gives a comparison of the number of the multiplication operations needed for each algorithm. Obviously, the proposed algorithm is slightly more complex than the SBP, while in the sequel, we will see that it has a faster convergence behavior than the SBP in number of training iterations and in computing time.

4. EXPERIMENTAL RESULTS

The 4-byte parity checker problem (logic problem) and the circle in the square problem (analog problem) are the benchmark used in this section.

The parameters of the sigmoidal function, the learning and the weighting coefficient for the new algorithm are chosen the same for both algorithms. Moreover we give the range of the initial values of the synaptic coefficients. For all applications we have used the same NN structure with 8 hidden neurons, one output neuron. The input neuron number vary according to the application : 4 for the 4-byte parity checker problem and 2 for the circle in the square problem. The sigmoidal threshold was hold constant: a=0.8.

4.1. The 4-byte parity checker problem

Table 1 shows that the new algorithm remains below a mean-squared error (MSE) equals to 10e-11 after only 150 iterations as opposed to the SBP algorithm at 460 iterations. From this we see that there is an improvement ratio nearly of 70 % for the number of iterations. We remark also that we reach a CPU time ratio of about 50 % with respect to the time requested for the SBP.

	4-byte parity checker problem (Threshold=10e-11)		
	Number of iterations	**CPU Time (s)/iteration**	**Total time of convergence(s)**
BP	**460**	**5.2210^{-3}**	**2.401**
MBP	**150**	**7.5810^{-3}**	**1.137**

Table 1 : Comparison of the CPU Time needed for the convergence of the MBP/SBP.

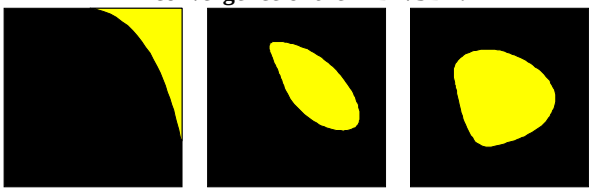
4.2. Circle in the square problem

In this application the NN have to decide if a point of coordinate x, y surrounded between -0.5 and +0.5 is in the circle of radius equal to 0.35.

Table 2 shows that the new algorithm remains below a MSE equal to 0.0018 after 135 iterations as opposed to the SBP algorithm at 400 iterations. From this we see that there is also an improvement ratio of about 66 % for the number of iterations and a CPU time ratio of about 50 % with respect to the time requested for the SBP.

	Circle in the square problem (Threshold=0.0018)		
	Number of iterations	**CPU Time (s)/iteration**	**Total time of convergence(s)**
BP	**400**	**29.6610^{-3}**	**11.864**
MBP	**135**	**45.9710^{-3}**	**6.206**

Table 2 : Comparison of the CPU Time needed for the convergence of the MBP/SBP.



(a) 2 iterations MSE=0.193 **(b) 30 iterations MSE=0.114** **(c) 100 iterations MSE=0.0259**

Fig.2 : SBP algorithm (μ=0.5);

(a) 2 iterations MSE=0.229 **(b) 10 iterations MSE=0.0358** **(c) 30 iterations MSE=0.0156**

Fig.3 : MBP algorithm (μ=0.5;λ=0.7);

The output of the network at various iteration numbers is shown in Fig. 2 (a, b, c) for the SBP and Fig. 3 (a, b, c) for the MBP. It is seen that the new algorithm forms a circle in 30 iterations whereas the SBP takes more than 100 iterations.

5. CONCLUSION

In this paper we have proposed a new fast algorithm for training neural networks based on a criterion taking into account the linear and nonlinear signal errors. The new MBP algorithm converges on fewer iterations when compared to the SBP algorithm for a suitable choice of the learning parameters. The optimal range values of the parameter λ is given in the appendix. Since this range become small during the learning procedure a decreasing type exponential of the weighted term is required.

6. REFERENCES

[1] V.J.MATHEWS, "Adaptive Polynomial Filters" *IEEE SP Magazine*.pp 10-26. July 1991.

[2] G.J. Wang. C.C. Chen " A fast multilayer neural network training algorithm based on the layer-by-layer optimization procedures " *IEEE Transactions on neural networks*. Vol. 7, No.3,pp768-775 May 1996.

[3] A.Cichocki . R.Unbehauen, " Neural Network for Optimization and Signal Processing " John Wiley & Sons Ltd. Baffins Lane, Chichester. West Sussex PO19 1UD, England 1993.

[4] S.Haykin, " Adaptive Filter Theory " Englwood Cliffs, NJ : Prentice-Hall, 1986.

[5] R. S. Scalero and N. Tepedelenlioglu, " A Fast New Algorithm for Training Feedforward Neural Networks " *IEEE Transactions on signal processing*. Vol 40. No, 1, pp 202-210,January 1992.

[6] M. R. Azimi-Sadjadi and R.J. Liou, " Fast Learning Process of multilayer Neural Network Using Recursive Least Squares Method " *IEEE Transactions on Signal Processing*, Vol 40 No, 2, February 1992.

[7] R.P.Lippman, " An Introduction to Computing with Neural Networks " *IEEE.ASSP*.Mag. Vol 4. No, 2 Apr 1987.

[8] F.Fnaiech. D.Bastard. V. Buzenac. R. Settineri and M. Najim, " A Fast Kalman Filter based new algorithm for training feedforward neural networks " Procedings of EUSIPCO,94,13st-16st septembre 1994 Eidumbrg, Scottland, UK.

[9] M.Sayadi, F.Fnaiech, A.Chaari et M.Najim, " Apprentissage rapide des réseaux de neurones multicouches : Application de l'algorithme des moindres carrées récursifs factorisés " Actes des conférences internationales: NEUROTECH'94 Les réseaux neuromimétiques et leurs application. pp.311-322. 15-16 Decembre 1994. Marseille,(France).

[10] Y. Liguni, H. Sakai, H. Tokumaru, " A Real-Time Algorithm for Multilayered Neural Network Based on the Extended Kalman Filter " *IEEE Transactions on Signal Processing*, Vol.40 No,4,pp 959-966,April 1992.

[11] N.B. Carayiannis and A.V. Venetsanopoulos, " Fast learning algorithm for neural networks " *IEEE Trans,on Circuits and systems* - 11, Vol 39, pp,453-474,1992.

[12] F.B. König and F.Barmann, " A learning Algorithm for multilayered neural networks based on linear squares problems " *Neural Networks*, Vol 6, pp 127-131 1993.

[13] C. Brezinski « Algorithmes d'accelération de la convergence ; étude numérique » Editions Technip. ISBN 2-7108-0341-0

APPENDIX

As mentioned in 3.1. we assume that we know the desired nonlinear output of this neuron for the chosen pattern. Then the desired summation can be directly computed by inverting the nonlinearity.

By deriving E given by (10) with respect to $w_{ji}^{[s]}$ it follows :

$$\frac{\partial E}{\partial w_{ji}^{[s]}} = -e_{1j}^{[s]} f'(u_j^{[s]}) y_i^{[s-1]} - \lambda e_{2j}^{[s]} y_i^{[s-1]} \tag{22}$$

For the SBP algorithm : λ=0

$$\begin{aligned} w_{ji}^{[s]}(k+1) &= w_{ji}^{[s]}(k) + \mu f'(u_j^{[s]}) e_{1j}^{[s]} y_i^{[s-1]} \\ &\overset{\Delta}{=} F_i(w_{ji}^{[s]}) \end{aligned} \tag{23}$$

For the new algorithm ($\lambda > 0$) then :

$$\begin{aligned} w_{ji}^{[s]}(k+1) &= w_{ji}^{[s]}(k) + \mu f'(u_j^{[s]}) e_{1j}^{[s]} y_i^{[s-1]} \\ &\quad + \mu\lambda e_{2j}^{[s]} y_i^{[s-1]} \overset{\Delta}{=} H_i(w_{ji}^{[s]}) \end{aligned} \tag{24}$$

Note that we have :

$$H_i(w_{ji}^{[s]}) = F_i(w_{ji}^{[s]}) + \mu\lambda y_i^{[s-1]} e_{2j}^{[s]} \tag{25}$$

Now, assume that the functions F_i and H_i have the same fixed point over a certain range in the real set $\Re$. Hence, the recursions (23) and (24) may be viewed as two series functions of values $\{t_{ji} = F_i(w_{ji})\}$ and $\{h_{ji} = H_i(w_{ji})\}$ which may converge to this point.

In order to compare the speed of convergence of both F_i and H_i which in other words corresponds to the comparison of the new proposed modified algorithm and the SBP algorithm, we shall compare their corresponding rank and also their asymptotic constant errors. First let us state these following theorems [13].

Theorem 1:

Assume that a series $\{x_k\}$ is generated by $x_{k+1}=F(x_k)$. If this series converges to (x) and if F is differentiable in the neighborhood of (x) then the rank of the series $\{x_k\}$ is the smallest integer number r verifying:

$F'(x)=...=F^{(r-1)}(x)=0$ and $F^{(r)}(x)\neq 0$

We have also : $\lim_{k\to\infty} \frac{x_{k+1}-x_k}{(x_k-x)^r} = \frac{F^{(r)}(x)}{r!}$

The asymptotic constant error of the series $\{x_k\}$ is defined by

$$c_{(x)} \overset{\Delta}{=} \left| \lim_{k\to\infty} \frac{x_{k+1}-x_k}{(x_k-x)^r} \right| = \left| \frac{F^{(r)}(x)}{r!} \right| \tag{26}$$

Theorem 2:

If $\{t_{ji}\}$ and $\{h_{ji}\}$ are two series of ranks r and p respectively and if $r>p$ then $\{t_{ji}\}$ converges faster than $\{h_{ji}\}$. In the case of $r= p$ and the asymptotic constant error $c_{t_{ji}}$ is less than $c_{h_{ji}}$ then the series $\{t_{ji}\}$ converges faster than $\{h_{ji}\}$.

a- Determination of the rank *"r_1"* of F_i :

For this matter, let us compute the first derivative of F_i with respect to $w_{ji}^{[s]}$:

$$\frac{\partial F_i(w_{ji}^{[s]})}{\partial w_{ji}^{[s]}} = 1 + \mu y_i^{[s-1]} \bullet \bullet \left[\frac{\partial f'(u_j^{[s]})}{\partial w_{ji}^{[s]}} e_{1j}^{[s]} - f'(u_j^{[s]}) \frac{\partial y_j^{[s]}}{\partial w_{ji}^{[s]}} \right] \quad (27)$$

After some calculation we obtain :

$$\frac{\partial F_i(w_{ji}^{[s]})}{\partial w_{ji}^{[s]}} = 1 - \mu \left(y_i^{[s-1]}\right)^2 f'(u_j^{[s]}) \bullet \bullet \left[f'(u_j^{[s]}) - (1 - 2 f(u_j^{[s]})) e_{1j}^{[s]} \right] \quad (28)$$

Case 1 :

* If $\left[f'(u_j^{[s]}) - (1 - 2 f(u_j^{[s]})) e_{1j}^{[s]} \right] = 0$ or $y_i^{[s-1]} = 0$ then $F_i'(w_{ji}^{[s]}) = 1$. Then r_1 =1.

Case 2 :

* If $\left[f'(u_j^{[s]}) - (1 - 2 f(u_j^{[s]})) e_{1j}^{[s]} \right] \neq 0$ and $y_i^{[s-1]} \neq 0$ then taking into account that :

$0 < f(u_j^{[s]}) < 1,\ 0 < f'(u_j^{[s]}) < 1$ and $0 < \left(y_i^{[s-1]}\right)^2 < 1$

we obtain :

$$-1 < -\left(y_i^{[s-1]}\right)^2 f'(u_j^{[s]}) \bullet \bullet \left[f'(u_j^{[s]}) - (1 - 2 f(u_j^{[s]})) e_{1j}^{[s]} \right] < 1$$

then $F_i'(w_{ji}^{[s]}) > 0$ while

$$0 < \mu < \frac{1}{y_i^{[s-1]^2} f'(u_j^{[s]})} \bullet \bullet \frac{1}{\left[f'(u_j^{[s]}) - (1 - 2 f(u_j^{[s]})) e_{1j}^{[s]} \right]}$$

then $r_1 = 1$**.** For both cases we have : $F_i'(w_{ji}^{[s]}) > 0$, then the rank of this series is equal to 1. The asymptotic constant error of F_i is then equal to :

$$c_{t_{ji}} = \left| F'_i(w_{ji}^{[s]}) \right| = F'_i(w_{ji}^{[s]}) \quad (29)$$

b-Determination of the rank *"r_2"* of H_i:

Using equation (25) we find easly that :

$$\frac{\partial H_i(w_{ji}^{[s]})}{\partial w_{ji}^{[s]}} = \frac{\partial F_i(w_{ji}^{[s]})}{\partial w_{ji}^{[s]}} - \mu\lambda y_i^{[s-1]} \frac{\partial u_j^{[s]}}{\partial w_{ji}^{[s]}}$$

$$H_i'(w_{ji}^{[s]}) = F_i'(w_{ji}^{[s]}) - \mu\lambda \left(y_i^{[s-1]}\right)^2 \quad (30)$$

Case 1:

* If $F_i'(w_{ji}^{[s]}) = \mu\lambda \left(y_i^{[s-1]}\right)^2$ i.e. $H'_i(w_{ji}^{[s]}) = 0$ then the rank of this series become greater or equal to 2 i.e. $r_2 \geq 2$. We obtain a convergence speed at least double of the one obtained by the SBP.

Case 2:

* If $F_i'(w_{ji}^{[s]}) \neq \mu\lambda \left(y_i^{[s-1]}\right)^2$ i.e. $H'_i(w_{ji}^{[s]}) \neq 0$ then $r_2 = 1$. To make the series H_i converges faster than F_i it will be enough that $c_{t_{ji}} < c_{h_{ji}}$ i.e.: $\left|H_i'(w_{ji}^{[s]})\right| < \left|F_i'(w_{ji}^{[s]})\right|$.

This always holds when $0 < \lambda < \frac{2F'_i(w_{ji}^{[s]})}{\mu \left(y_i^{[s-1]}\right)^2}$ for this condition we have assumed that $\left(y_i^{[s-1]}\right)^2 \neq 0$. Then the series H_i converges faster than F_i .

If $\left(y_i^{[s-1]}\right)^2 = 0$ then $H_i'(w_{ji}^{[s]}) = F_i'(w_{ji}^{[s]})$ then $c_{t_{ji}} = c_{h_{ji}}$. In this case the two series have a similar convergence behavior, but it must be noted that this case can be occur only in the input layer for null components vectors in the learning input pattern. For the hidden and output layers we have always $\left(y_i^{[s-1]}\right)^2 \neq 0$ and the convergence is always faster than the SBP.

In some simulation results, we find out that it is convenient to choose λ as :

$\lambda(k+1) = \frac{1}{c\lambda(k)}$ where c is a positive constant.

Table 3 : Complexity evaluation of the MBP/SBP algorithm

Algorithm	SBP	MBP
Errors	$2n_L + \sum_{s=1}^{L-1} n_s(n_{s+1}+2)$	$5n_L + \sum_{s=1}^{L-1} n_s(n_{s+1}+2) + \sum_{s=1}^{L-1} n_s(n_{s-1}+1)$
Updating	$\sum_{s=1}^{L} n_s(n_{s-1}+2)$	$\sum_{s=1}^{L} n_s(n_{s-1}+2) + \sum_{s=1}^{L} n_s(n_{s-1}+2)$