International Journal of Soft Computing 3 (5): 359-367, 2008
ISSN: 1816-9503

Fast Training of Multilayer Perceptrons with Least Mean Fourth (LMF) Algorithm

Sabeur Abid and Farhat Fnaiech
Research Team (Signal, Image and Intelligent Control of Industrial Systems: SICISI),
Ecole Supérieure des Sciences et Techniques de Tunis (ESSTT),
5 Av. Taha Hussein, 1008, Tunis, Tunisia

Abstract: In this study, we propose a new fast training algorithm for multilayer perceptron (MLP). This new algorithm is based on the optimisation of Least Fourth (LF) criterion producing a modified form of the Standard Back-Propagation (SBP) algorithm. In this criterion, the Least Fourth error signal is appropriately weighed by the learning coefficient of the steepest descent method. To determine the updating rules for the hidden layers, a similar back propagation method used in the SBP algorithm is developed. This permits the application of the learning procedure to all the neural network layers. Several experiments was carried out indicate significant reduction in the total iteration number, in the convergence time and in the generalization capacities when compared to those of the SBP algorithm.

Key words: Multilayer Perceptron (MLP) Fast training algorithms, Standard Back Propagation (SBP) algorithm, least square error, least fourth error

INTRODUCTION

Neural networks viewed as nonlinear adaptive systems, have drawn great interest in the last two decades for multiple advantages such as: Capacities of learning and generalisation.

The back propagation learning algorithm is the standard algorithm used to train multilayer percpetrons. This algorithm suffers from many drawbacks and especially the slowlyness of the convergence. This algorithm is based on the first order optimisation methods. In the following we give a preview of main equations of this algorithm.

Assume that $E(\underline{w})$ is a cost function to minimize with respect to the parameter vector $\underline{w}$, $\nabla E(\underline{w})$ and is the gradient vector of $E(\underline{w})$ with respect to $\underline{w}$, then first order methods are based on the following rule:

$$\Delta \underline{w} = -\mu \frac{\partial E(\underline{w})}{\partial \underline{w}} = -\mu \nabla E(\underline{w}) \quad (1)$$

which is known in literature as the steepest descent algorithm (Abid *et al*., 2001; Fnaiech *et al*., 2002; Cichocki and Unbehauen, 1993; Charalambous, 1992). μ is called the learning coefficient.

Us we noted above, this algorithm suffers from multiple shortcomings. One of which is the slow rate at which the algorithm converges. To accelerate the convergence speed of the algorithm several methods and techniques based on this algorithm are developed (Abid *et al*., 2001; Fnaiech *et al*., 2002; Jacoob, 1988). These techniques use many ideas such as decreasing the learning coefficient, using a decreasing momentum, varying the objective function…

In this setting is located this present work that aims at proposing a new fast algorithm based on first order method and the SBP technique.

In Abid *et al*. (2001) we have proposed a new fast learning algorithm based on an optimization criterion formed by the sum of the linear and the nonlinear quadratic errors of a sigmoidal neuron. This has yielded to a modified form of the conventional SBP algorithm. We have shown that this can increases the convergence speed considerably. In this paper a new algorithm is presented, which is faster than the SBP algorithm. This algorithm is based on the optimization of a Least Mean Fourth Criterion (LMF).

The Least Square and Least Fourth algorithms belong to the family of Least Mean Squares (LMS) and Least Mean Fourth (LMF) algorithms. The LMS algorithm is the most widely used algorithm for adaptive filters in many applications (Haykin, 1991). On the other hand the LMF algorithm was proposed as a special case of general family of steepest descend algorithms (Zerguine, 2002; Walach and Widrow, 1984). The new proposed algorithm

Corresponding Author: Sabeur Abid, Research Team (Signal, Image and Intelligent Control of Industrial Systems: SICISI), Ecole Supérieure des Sciences et Techniques de Tunis (ESSTT), 5 Av. Taha Hussein, 1008, Tunis, Tunisia

consists at minimizing a criterion expressed by the fourths errors for all output neurons and for the current pattern. Training patterns are run through the network until convergence is reached. Experimental results show that this algorithm requires a lower number of iterations for convergence and less time in comparison with the SBP algorithm. In some examples we can reach a decrease of more than 40% in training iterations with better generalization capacities.

MULTILAYER PERCEPTRON DESCRIPTION

The used MLP, is formed by interconnections of neurons of type given by Fig. 1:

The linear and nonlinear actual outputs are, respectively given by:

$$u_j^{[s]} = \sum_{i=0}^{n_{s-1}+1} w_{ji}^{[s]} y_i^{[s-1]} \quad (2)$$

$$f(u_j^{[s]}) = \frac{1}{1+e^{-u_j^{[s]}}} = y_j^{[s]} \quad (3)$$

where $w_{ji}^{[s]}$, is the weight of the ith unit in the (s-1)th layer to the jth unit in the sth layer and $(n_{s-1}+1)$ is the number of inputs to the neuron (j).

The nonlinear error signal is given by:

$$e_j^{[s]} = d_j^{[s]} - y_j^{[s]} \quad (4)$$

where, $d_j^{[s]}$ and $y_j^{[s]}$ are the desired and the actual outputs, respectively.

The feedforward interconnections between these neurons lead to the feedforward MLP given in Fig. 2.

n_s is the number of the units in layer s and the subscript L denotes the output layer.

The steepest descent method is the standard one used to train these types of networks. This leads to the most known learning algorithm which is the backpropagation algorithm. In the sequel we will give a brief review of this algorithm.

Review of the Standard Backpropagation (SBP) algorithm: The SBP algorithm has become the standard algorithm used for training MLP. It is a generalized Least Mean Squared (LMS) algorithm that minimizes a criterion equals to the sum of the squares of the errors between the actual and the desired outputs. This criterion is equal to:

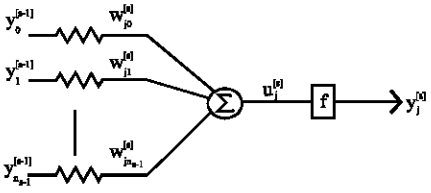

Fig.1: A neuron j in a layer [s] of a MLP

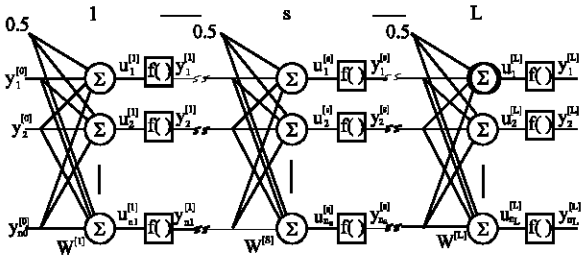

Fig. 2: Fully connected feedforward multilayer perceptron

$$E_p = \sum_{j=1}^{n_L} \left(e_j^{[L]}\right)^2 \quad (5)$$

where the nonlinear error signal is given by:

$$e_j^{[L]} = d_j^{[L]} - y_j^{[L]} \quad (6)$$

$d_j^{[L]}$ and $y_j^{[s]}$ are, respectively the desired and the current outputs, for the j^{th} unit. p denotes in (5) the p^{th} pattern and n_L is the number of output units. The application of the gradient descent method given by Eq. (1) to the criterion E_p leads to:

$$\Delta w_{ji}^{[s]} = -\mu \frac{\partial E_p}{\partial w_{ji}^{[s]}} \quad (7)$$

Since, the SBP algorithm is widely treated in literature (Cichocki, 1993), we summarize it by the following steps:

- Compute the local error signals for the output layer using:

$$\delta_j^{[L]} = f'(u_j^{[L]}) e_j^{[L]} \quad (8)$$

- Compute the local error signals for the hidden layers, i.e., for s = L-1 to 1, using:

$$\delta_j^{[s]} = f'(u_j^{[s]}) \sum_{r=1}^{n_{s+1}} \left(\delta_r^{[s+1]} w_{rj}^{[s+1]}\right) \quad (9)$$

where, f′ is the first derivative of f with respect to $w_{ji}^{[s]}$.

- Update the weights according to the following equation without momentum:

$$w_{ji}^{[s]}(k+1) = w_{ji}^{[s]}(k) + \mu\delta_j^{[s]}y_i^{[s-1]} \tag{10}$$

In the sequel we will study the different approaches that govern the learning speed of a MLP neural network.

DIFFERENT APPROACHES FOR INCREASING MLP TRAINING ALGORITHMS

As mentioned above, the learning of the MLP using the SBP algorithm is plagued by slow convergence. We have summarized the approaches for increasing the convergence speed onto seven cases:

The weight updating procedure: There are two methods used to present patterns to the network: the on-line method and the batch method. The first consists to present a pattern (a learning example) to the network inputs and then all weights are updated before the next pattern is presented, whereas the batch method consists to accumulate weight changes Δw (w is the weight of any neuron connection) over some number of the learning examples and then synaptic weights are actually changed.

The optimization criterion choice: lost functions: In some applications and especially when the output layer of neurons is large and the signals are contaminated by non-Gaussian noise a variety of loss functions can be used in order to reduce the noise influence.

From many loss functions known from robust statistics the following four functions are the most used ones (Hampel *et al*., 1987):

- The logistic function.

$$\sigma_L(e) = \beta^2 \ln\left(\cosh(e/\beta)\right) \tag{11}$$

- The Huber's function.

$$\sigma_H(e) = \begin{cases} e^2/2 & \text{for} \quad |e| \le \beta \\ \beta|e| - \beta^2/2 & \text{for} \quad |e| > \beta \end{cases} \tag{12}$$

- The Huber's function with saturation, also called Talvar's function.

$$\sigma_T(e) = \begin{cases} e^2/2 & \text{for} \quad |e| \le \beta \\ \beta^2/2 & \text{for} \quad |e| > \beta \end{cases} \tag{13}$$

- The Hampel's function.

$$\sigma_{Ha}(e) = \begin{cases} \dfrac{\beta^2}{\Pi}\left(1 - \cos\dfrac{\Pi e}{\beta}\right) & \text{for} \quad |e| \le \beta \\ 2\dfrac{\beta^2}{\Pi} & \text{for} \quad |e| > \beta \end{cases} \tag{14}$$

where, β> 0 is a problem-dependent control parameter.

Different forms of error functions have been proposed in order to increase the convergence speed and/or to improve the generalization capacities.

In Karayiannis and Venetesanopoulos (1992) Karayiannis and al developed the following generalized criterion for training MLP neural networks:

$$E_p = (1-\lambda)\sum_{j=1}^{n_L}\sigma_1(e_{jp}) + \lambda\sum_{j=1}^{n_L}\sigma_2(e_{jp}) \tag{15}$$

where $\sigma_1(e_{jp})$ and $\sigma_2(e_{jp})$ are suitable loss functions, which are everywhere convex and differentiable and $\lambda \in [0,1]$ is gradually (from 1 to 0) decreasing parameter during the learning process.

It has been proposed that the parameter λ is computed according to the following rule:

$$\lambda = \lambda(E) = \exp(-c/E^2) \tag{16}$$

where, c> 0 is a positive real number.

Based on Eq. (15), we have developed in (Abid *et al*., 2001) a new error function using the linear and nonlinear neuron outputs (MBP algorithm).

The use of adaptive parameters: In literature we have found that the use of an adaptive slope of the activation function or global adaptation of the learning rate can increase the convergence speed in many applications.

For example to find the updating rule for the slope of the activation function, we apply the gradient method with respect to a:

$$\Delta a(k) = -\mu_a \frac{\partial E}{\partial a} \tag{17}$$

where a and E are the slope of the activation function of a neuron and the optimization function, respectively.

For the learning parameter, Darken and Moody in (Darkin and Moody, 1991) have suggested a modification of the learning coefficient along the training phase (PRC algorithm) such as:

$$\mu(k) = \mu_0 \frac{1}{1+\frac{k}{k_0}} \tag{18}$$

or

$$\mu(k) = \mu_0 \frac{1+\frac{c}{\mu_0}\frac{k}{k_0}}{1+\frac{c}{\mu_0}\frac{k}{k_0}+k_0\left(\frac{k}{k_0}\right)^2} \tag{19}$$

Where, c and k_0 are positive constants and μ_0 is the initial value of the learning coefficient μ.

Estimation of optimal initial conditions: The general problem of iterative search methods lies in the initialisation point. In the SBP algorithm, we always start with random initial weight values. The convergence speed of these algorithms can be considerably improved by finding appropriate initial weights.

Pre-processing the problem before using MLP: The pre-processing of data (by employing future extraction algorithms or projection methods) allows to reduce the size of the problem and then the MLP size. This can in many cases increases the learning speed.

Optimisation of the MLP structure: One can think that, starting with optimal MLP structure i.e. optimal number of the hidden layers and their corresponding number of neuron, can improve the learning stage.

Use of more advanced algorithms: Numerous optimization algorithms have been proposed to improve the convergence speed of the SBP algorithm. Unfortunately, some of these algorithms are computationally very expensive, i.e. they require a large increase of storage and computational cost which can become unmanageable even for a moderate size of neural networks. Among these algorithms we state:

- The Recursive Least Square algorithm (RLS).
- The Marquardt Levenberg algorithm.

In the sequel, we propose a new fast algorithm based on the minimisation of a fourth order optimisation criterion.

THE NEW MEAN LEAST FOURTH (MLF) ALGORITHM

The new proposed minimizing criterion, based on fourth order errors is given by:

$$E_p = \frac{1}{4}\sum_{j=1}^{n_L}\sum_{j=1}^{n_L}\left(e_j^{[L]}\right)^4 \tag{20}$$

Next, we derive the updating equations for both output and hidden layers.

Learning of the output layer: The weight update rule for the output layer can be derived by applying the gradient descent method to E_p, we get then:

$$\Delta w_{ji}^{[L]} = -\mu\frac{\partial E_p}{\partial w_{ji}^{[L]}} = \mu\frac{\partial y_j^{[L]}}{\partial u_j^{[L]}}\frac{\partial u_j^{[L]}}{\partial w_{ji}^{[L]}}\left(e_j^{[L]}\right)^3 \tag{21}$$

$$\Delta w_{ji}^{[L]} = \mu f'(u_j^{[L]}) y_i^{[L-1]}\left(e_j^{[L]}\right)^3 \tag{22}$$

By analogy to Eq. (8), we define the local error signals for the output layer by:

$$\delta_{3j}^{[L]} = f'(u_j^{[L]})\left(e_j^{[L]}\right)^3 \tag{23}$$

The updating equation becomes then:

$$\Delta w_{ji}^{[L]} = \mu\delta_{3j}^{[L]} y_{1i}^{[L-1]} \tag{24}$$

Learning of the hidden layers: First we apply the gradient descent method to E_p for the layer [L-1] and then we generalize results for the other hidden layers.
For the $[L-1]^{th}$ hidden layer we can write:

$$\Delta w_{ir}^{[L-1]} = -\mu\frac{\partial E_p}{\partial w_{ir}^{[L-1]}} \tag{25}$$

The application of the chain rule derivation yields to:

$$\begin{aligned}\Delta w_{ir}^{[L-1]} &= -\mu\frac{\partial E_p}{\partial e_j^{[L]}}\frac{\partial e_j^{[L]}}{\partial y_j^{[L]}}\frac{\partial y_j^{[L]}}{\partial u_j^{[L]}}\frac{\partial u_j^{[L]}}{\partial y_i^{[L-1]}}\cdot \\ &\cdot\frac{\partial y_i^{[L-1]}}{\partial u_i^{[L-1]}}\frac{\partial u_i^{[L-1]}}{\partial w_{ir}^{[L-1]}} \\ &= \mu f'(u_i^{[L-1]}) y_r^{[L-2]}\sum_{j=1}^{n_L}\left(e_j^{[L]}\right)^3 f'(u_j^{[L]}) w_{ji}^{[L]}\end{aligned} \tag{26}$$

Let us define the local signal errors by:

$$\delta_{3i}^{[L-1]} = \sum_{j=1}^{n_L}\left(e_j^{[L]}\right)^3 f'(u_j^{[L]}) w_{ji}^{[L]} \tag{27}$$

Table 1 : Different steps of the new LMF algorithm

Step 1:	Initialization: *From layer s=1 to L, set all $y_0^{[s]}$ to values different from 0, (0.5 for example). *Randomize all the weights at $y_{ji}^{[s]}$ random values. *Choose a small positive value μ.
Step 2:	Select training pattern: Select an input/output pattern to be fed into the network.
Step 3:	Run selected pattern p through the network for each layer (s), (s = 1... L) and calculate for each node j the linear and nonlinear outputs: Eq. (2) and (3).
Step 4:	Error signals: *For the output layer L: calculate the local output errors: Eq. (14) and (15). *For the hidden layers: s =L-1 to 1 compute the local signal errors: Eq. (23) and (24).
Step 5:	Updating the synaptic coefficients: For the output layer L, update the weights: Eq. (16) For any node j of the layer s=1 to L-1 modify the synaptic coefficients using Eq. (22).
Step 6:	Testing for the ending of the running: Various criteria are tested for ending. We can use the mean squared error of the network output as a convergence test or we can run the program for a fixed number of iterations. If the condition is not verified, go back to Step 2.

The updating Eq. (24) for the $[L-1]^{th}$ layer becomes:

$$\Delta w_{ir}^{[L-1]} = \mu f'(u_i^{[L-1]}) y_r^{[L-2]} \delta_{3i}^{[L-1]} \quad (28)$$

This differentiation procedure may be performed layer by layer. Hence, for a given layer s the weight updating equation becomes:

$$\Delta w_{ji}^{[s]} = \mu f'(u_j^{[s]}) y_i^{[s-1]} \delta_{3j}^{[s]} \quad (29)$$

Where:

$$\delta_{3j}^{[s]} = \sum_{j=1}^{n_s} \left(e_j^{[s+1]}\right)^3 f'(u_j^{[s+1]}) w_{ji}^{[s+1]} \quad (30)$$

both for s = L-1 ... 1

Finally, Table 1 summarizes the different steps of the new algorithm.

COMPARISON OF THE COMPUTATION COMPLEXITY

Table 2 gives a comparison of the number of multiplication operations needed for each algorithm to compute the error signals and the updating equations for one pattern. Obviously, the proposed algorithm is slightly more complex than the SBP algorithm. However, it is shown below to have faster convergence behavior in terms of the number of iterations needed and in computation time.

EVALUATION OF THE SENSITIVITY TO SYNAPTIC WEIGHTS INITIALIZATION

To study the sensitivity to weights initialization, we have tested the convergence of each algorithm for a huge number of different random initialization trials. For all simulations we have taken the same number of trials

Table 2: Multiplication operation number of the LF/SBP algorithm

Algorithm	SBP	LF
Errors	$2n_L + \sum_{s=1}^{L-1} n_s(n_{s+1} + 2)$	$2n_L + \sum_{s=1}^{L-1} n_s(n_s^2 + n_{s+1} + 2)$
Updating	$\sum_{s=1}^{L} n_s(n_{s-1} + 2)$	$\sum_{s=1}^{L} n_s(n_{s-1} + 3)$

(1000). We mean by a trial one training phase with one weight random initialization. The ending criterion is the Mean Squared Error (MSE) defined by:

$$E = \frac{1}{N} \sum_{p=1}^{N} \sum_{i=1}^{n_L} (e_i^{(L)})^2 \quad (31)$$

where, N is the total number of training patterns.

Each training phase is stopped if the MSE reaches a threshold fixed beforehand. This threshold is selected depending on the application and it will be denoted in the sequel as: ending _ threshold.

Each learning trial must be stopped if the ending threshold is not reached after an iteration number fixed a priori. The choice of this number depends on the application too. This number is denoted by: Iter _ number. The convergence is assumed to be failed if Iter _ number is reached before the value of ending _ threshold.

The sensitivity to weight initialization is evaluated via:

$$S_w(\%) = 100.\left(1 - \frac{\text{number of convergent trials}}{\text{total number of trials}}\right) \quad (32)$$

Then: more S_w is small, less the algorithm is sensitive to weight initialization.

STUDY OF THE GENERALIZATION CAPACITIES

To study the generalization capacities (G_C) after the training phase, we should present new patterns (testing

patterns), that we know the desired outputs, to the networks and compare the neural outputs with respect to the desired ones. If the norm of the error between these two inputs is smaller than a threshold chosen beforehand (denoted in the sequel gen _ threshold) then the new pattern is assumed to be recognized with success.

The generalization capacity is evaluated via the following proposed formula:

$$G_c(\%) = 100.\frac{\text{recognized pattern number}}{\text{total testing patterns number}} \quad (33)$$

Then: More G_C is big, more the network have a good generalization capacity.

SIMULATION RESULTS AND PERFORMANCES COMPARISON

To compare performances of the new algorithm with respect to the conventional SBP one, both algorithms are used to train networks for the same problem. In this study, we present four examples, the 4-b parity checker (logic problem), the circle in the square problem (analog problem) and the brain diseases classification (a real medical problem). For both algorithms, learning parameters (such as μ...) are selected after a lot of trials (100) to maximize performances of each algorithm. However, an exhaustive search for best possible parameters is beyond the scope of this paper and other optimal values may exist for each algorithm. In order to make suitable comparison we keep the same neural network size for testing both algorithms.

The 4-b parity checker: The aim of this application is to determine the parity of 4-bit binary number. For more details see (Abid *et al.*, 2001). Figure 3 shows the average of the Mean Squared Error (MSE) over 100 different weight initializations (only convergent trials are considered for tracing these curves).

We note that the LMF algorithm is highly faster than the SBP. The new LMF algorithm remains below 510^{-4} after only 220 iterations as opposed to the SBP algorithm at 500 iterations. Based on these experiments, clearly we see that there is an improvement ratio, nearly 2.25, for the number of iterations. The computation time per iteration was calculated and found to be similar for the two algorithms. Consequently the improvement ratio for the convergence time is about 2.25.

We note that the new algorithm is very sensitive to the choice of μ and one should make many trials to find a good value to ensure the rapidity of the algorithm.

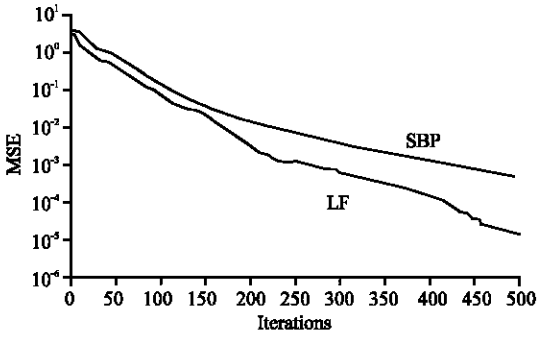

Fig. 3: Learning curve for the LF versus SBP algorithm for the 4-b parity checker (μ = 5)

Table 3: Performance comparison of the LMF algorithm with respect to the SBP one for the 4-b parity checker

	S_w	G_c	Improvement ratio on iterations	Improvement ratioon time
LF	42	98	22	22
SBP	81.8	82		

To evaluate the sensitivity to weights initialization S_W we have chosen iter _ number = 500 and ending _ threshold = 0.1.

For the generalization test G_C we have presented to the network a 4-bit distorted numbers. The distortion rate with respect to the exact 4-bit binary numbers is about 30% and gen _ thresholds. Table 3 resumes performances of the 2 algorithms for this application.

From these results we note that the LMF network is less affected by the choice of the initial weights and it has good generalization capacities with respect to the SBP one.

The circle in the square problem: In this application, the MLP have to decide if a point of coordinate (x, y) varying from -0.5 to +0.5 is in the circle of radius equals to 0.35. The input coordinates are selected randomly. If the distance of training point from the origin is less than 0.35, the desired output is assigned the value 0.1 indicating that the point is inside the circle. A distance greater than 0.35 means that the point is outside the circle and the desired output becomes 0.9. Training patterns are presented to the network alternating between the two classes (inside and outside the circle). In one iteration we present to the network 100 input/output patterns. The output of the network at various iteration numbers is shown in Fig. 4(a-c) for the SBP algorithm and Fig. 5(a-c) for the new algorithm. It is seen that the new algorithm forms a circle in 10-30 iterations whereas the SBP algorithm needs more than 100 iterations.

It should be also noted that this method of specifying the desired output forces the "training

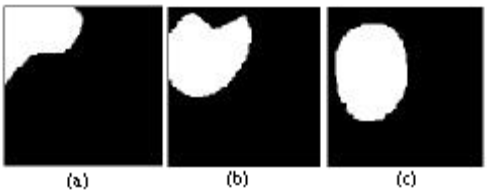

Fig. 4: SBP (μ = 0.95); (a) Iterations 2. MSE = 0.278. (b) Iterations 30. MSE = 0.106. (c) Iterations 100. MSE = 0.0198

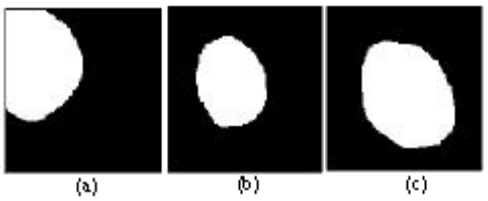

Fig. 5: LF. (μ = 0.95); (a) Iterations 2. MSE = 0.259. (b) Iterations 10. MSE = 0.0381. (c) Iterations 30. MSE = 0.0102

surface" to assume a top hat-like shape, a constant high value outside the circle and a constant low value inside the circle; which is not the best choice for the circle problem. Since, during the training period, fewer errors occur when the output values are closer to 0.1 and 0.9, the algorithms tend to sacrifice the transition region in order to flatten the inside and outside regions. The result is less than perfect circle. A training surface with continuous first derivatives and gradual transition region would improve the appearance of the circle.

Figure 6 shows the average of the MSE over 100 different good weight initializations versus the iteration number for both algorithms during training.

The new algorithm remains below 2×10^{-2} after 100 iterations as opposed to the SBP algorithm at 180 iterations. Notice that there is an improvement ratio of about 1.8 in the number of iterations and in the learning time.

To evaluate S_w we have chosen iter _ number = 200, ending _ threshold = 10^{-2}. For G_c we have presented to the network a new coordinate (x,y) and we have chosen gen _ threshold = 0.1.

Table 4 Resumes the performances of the two algorithms for this application.

For this analog problem we note again good performances of the new algorithm with respect to the SBP one.

Brain diseases c lassification: In this application, we use features from Contingent Negative Variation (CNV)

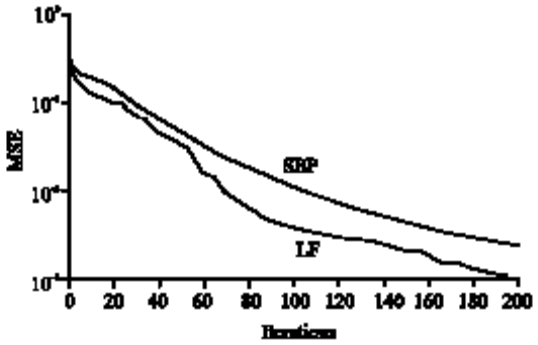

Fig. 6: Learning curve for the MLSLF versus SBP algorithm for the circle in the square problem

Table 4: Performance comparison of the LMF algorithm with respect to the SBP one for the circle in the square problem

	S_w	G_c	Improvement ratio on iterations	Improvement ratio on time
LF	463	97.1	1.8	1.8
SBP	69.7	76		

Table 5: Results of neural network training with the MLSLF and SBP algorithms (μ = 0.5; β = 1.25)

	S_w	G_c	Improvement ratio on iterations	Improvement ratio on time
LF	47	74	1.5	1.5
SBP	73.9	55.5		

waveforms of the electroencephalograms to train a MLP to classify four types of subjects: Huntington's disease, Parkinson's disease patients, patients with schizophrenia and normal subjects, Fig. 5. To study the redundancy between the CNV data used for the disease classification before the training step, we define a characterization capability as the ratio between "inter-class" and "intra-class" deviations of each feature (Fnaiech *et al*., 2004). The features are set into vectors $y^{[k]} = x_{k,n}$ of 17 elements, with k the disease index and n the example index. Let define N_k as the number of examples in disease class k (11, 16, 20 and 40, respectively for the Huntington's disease, Parkinson's disease, schizophrenia and normal cases). For classification purposes, the subjects were classified in the conventional way using the different features of CNV data. These features were used as input vectors to the MLP neural network Fig. 7.

Two networks were trained using the SBP and the LMF algorithms with 75% of the available data and tested with the remaining ones. The network weights were updated on each presentation of a feature vector. The trained networks have 17 input units, two hidden layers with 7 and 11 units, respectively and 4 output neurons. The ending threshold is equal to 0.01. The performances of the two algorithms are summarized in Table 5.

From these results, for this real medical problem, we note again that the new LMF algorithm has a good

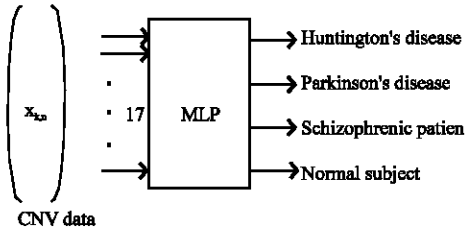

Fig. 7: Principle of brain disease classification with MLP, from CNV data

convergence speed, it is less affected by the choice of the initial weights and it has good generalization capacities with respect to the SBP one.

GRAPHIC INTERPRETATION OF THE HIGH CONVERGENCE RATE OF THE ALGORITHM

We give here a heuristic graphic interpretation of the high convergence of the LMF algorithm with respect to the SBP one.

Recall that the least mean fourth algorithm is based on the squared signal of the optimization criterion of the SBP algorithm. Indeed for the SBP algorithm we have:

$$E_{SBP} = \sum_{j=1}^{n_L} \left(e_j^{[L]}\right)^2$$

while, for the LMF algorithm we have:

$$E_{LMF} = \frac{1}{4} \sum_{j=1}^{n_L} \sum_{j=1}^{n_L} \left(e_j^{[L]}\right)^4$$

Assume that E_{SBP} admits a local (or global) minimum in any range of $\Re$ then E_{LMF} admits the same minimum but in this case the absolute value of the slope of the E_{LMF}'s curve, will be greater than that of E_{SBP} in the same point. Fig. (8) represents an example of a searching curve of a Squared Error (SE) near a local or global minimum ($|e_j^{[l]} < 1|$) and its corresponding fourth error (FE). For simplicity, we have assumed, for the representation, that E_{SBP} and E_{LMF} depend each one only on one variable w.

We know that the gradient method allows to follow the gradient in a descending direction. The convergence of the method is governed by the amplitude of the searching rate μ. If μ is big then we have a fast convergence but this can leads to oscillation phenomena or even a divergence. Whereas if μ is small then the algorithm becomes stable but this leads to a very slow convergence behavior.

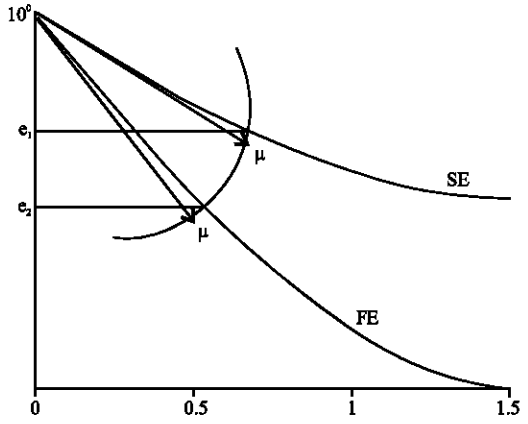

Fig. 8: Graphic interpretation of the higher speed of the fourth error algorithm with respect to the squared error one (SE: squared error's curve; FE: fourth error's curve). For the same amplitude of the searching rate μ, the error obtained in FE is less than that of SE.

Figure 8 shows that, for the same amplitude of μ and near the local or the global minimum, we reach a smaller error in the Fourth Error surface with respect to the Squared Error surface. And this is trivial because we have near the minimum (e.i. when) ($|e_j^{[l]} < 1|$):

$$\left(e_j^{[L]}\right)^4 < \left(e_j^{[L]}\right)^2$$

It becomes that for the same learning coefficient μ and for the same number of iterations, the error value reached following the curve of the mean fourth error is less than the one obtained when following the squared error's curve.

CONCLUSION

In this study, we have proposed a new fast algorithm for training multilayer neural network based on a new cost function of fourths errors. The convergence of the new LMF algorithm requires less iterations and less training time than the SBP for convergence and provides better generalization.

We have seen that for a real medical application, this new algorithm, in spite of its simplicity and its slight modification with respect to the SBP one, ensures a gain of convergence time near 40%. For other simple applications this gain is bigger. Furthermore this new algorithm has almost the same computation complexity of the SBP one.

Moreover, the convergence of the LMF algorithm (as in the case of the SBP one) is governed by the learning parameter μ. A bad choice of this parameter can cause the divergence. It is very important to develop the theory to find rules that help to choose this parameter. It will be important also to compare performances of this new algorithm with other fast algorithms such as the Modified Back propagation algorithm introduced in (Abid *et al*., 2001) and Recursive Least Square algorithm in (Azimi-Sadjadi and Liou, 1992).

REFERENCES

Abid, S., F. Fnaiech and M. Najim, 2001. A fast feed-forward training algorithm using. A modified form of the standard back propagation algorithm. IEEE. Trans. Neural Network, Vol. 12, No. 2.

Azimi-Sadjadi, M.R. and R.J. Liou, 1992. Fast learning process of multilayer neural network using recursive least squares method. IEEE. Trans. Signal Processing, Vol. 40, No. 2.

Berezinski, C., 1988. Algorithmique Numérique, Paris, France: Edition Marketing.

Cichocki, A.R., 1993. Unbehauen, Neural Network for optimization and signal processing, John Wiley and Sons Ltd. Baffins Lane, Chechester. West Sussex PO19 1UD, England.

Charalambous, C., 1992. Conjugate gradient algorithm for efficient training of artificial neural networks. IEEE. Proc., 139 (3): 301-310.

Darken, C. and J. Moody, 1991. Towards faster stochastic gradient search", in the book Advances in Neural information Processing systems 4, Morgan Kaufman, San Mateo, pp: 1009-1016.

Fnaiech, F., S. Abid, N. Ellala and M. Cheriet, 2002. A comparative study of fast neural networks learning algorithms. IEEE. International Conference on Systems, Man and Cybernetics, Hammamet, Tunisia.

Fnaiech, N., S. Abid, F. Fnaiech and M. Cheriet, 2004. A modified version of a formal pruning algorithm based on local relative variance analysis. First International Symposium on Control, Communications and Signal Processing: ISCCSP, Hammamet, Tunisia.

Graupe, D., 1984. Time series analisys, Identification Adaptive Filtering. Malabar, FL: Krieger.

Haykin, S., 1991. Adaptive Filter Theory, Prentice-Hall, Englewood Cliffs, NJ.

Hampel, F.R., E.N. Ronchetti, P. Rousser and W.A. Stachel, 1987. Robust statistics. The approach Based on Influence Functions, New York: John Wiley.

Jacobs, R.A., 1988. Increased rates of convergence through learning rate adaptation. Neural Networks, 1 (4): 295-308.

Karayiannis, N.B. and A.N. Venetsanopoulos, 1992. Fast learning algorithm for neural networks, IEEE Trans. Circuits and Syst., 39: 453-474.

Walach, E. and B. Widrow, 1984. The Least Mean fourth (LMF) Adaptive Algorithm and its family. IEEE. Trans. Inf. Theory IT., 30: 275-283.

Zerguine, A., 2002. A time-varying normalized mixed-norm LMS-LMF algorithm. 11th European Signal Processing Conference EUSIPCO, Paper coded #394. Toulouse, France.