The 5th International Conference on Electrical Engineering – Boumerdes (ICEE-B)
October 29-31, 2017, Boumerdes, Algeria.

A New Training Method For Solving The XOR Problem

LADJOUZI Samir
Department of Electrical Engineering University of Bouira
Bouira, Algeria
ladjouzi_samir2001@yahoo.fr

GROUNI Said
Department of Industrial Maintenance University of Boumerdes Boumerdes, Algeria
said.grouni@yahoo.com

KIRAT Abderrahmen Department of Industrial Maintenance University of Boumerdes Boumerdes, Algeria
kiratauto@gmail.com

SOUFI Youcef
Department of Electrical Engineering
University of Tébessa
Tébessa, Algeria
y_soufi@yahoo.fr

Abstract— **Training of Artificial Neural Networks (ANN) is an important step to make the network able to accomplish the desired task. This capacity of learning in such networks makes them applied in many applications as modeling and control. However, many of training algorithms have some drawbacks like: too many parameters to be estimated, important calculus time. In this paper, we propose a very simple method to train a Single Hidden Layer Perceptron (SHLP) based on replacing the traditional ANN's phase training by another approach called Neural Least Mean Square (NLMS) problem resolution. The key of this method is to compute some ANN's weights by the Least Mean Square (LMS) formula, and to leave others weights to their initial values. This new training method is applied to the classical XOR problem and the results are compared with the conventional Backpropagation algorithm. The obtained results were satisfactory and the comparison made with the classical algorithm revealed that our method allowed to reduce several parameters in the learning, namely: the computation time, the overall value of the error squared, the number of iterations and the number of weights to be adjusted.**

Keywords: *ANN's phase training - Single Hidden Layer Perceptron (SHLP) - Neural Least Mean Square (NLMS) - XOR problem - Backpropagation algorithm.*

I. INTRODUCTION

Among one of the interest areas of computer science widely used in many disciplines, we cite Artificial Neural Networks (ANN). A variety of ANN architectures are proposed by several researchers such Multi Layer Perceptron (MLP), Radial Basis Function (RBF), Hopfield network…etc., but the most used topology is the MLP, which is organized in layers where the first layer represent the inputs, the last is the outputs and between them there are hidden layers that transmit information. This emergent technique is used in various applications like: physics, signal processing, engineering…etc., but one of the classical tasks solved by ANN is Boolean problems especially the XOR problem.

This problem is enough difficult to solve because it is what we call a non linearly separable problem unsolved with a single perceptron. However, different solutions are proposed in literature for solving the XOR problem using ANN: in [1] a Genetic Algorithm was used to train an MLP, the authors proposed in [2] a new model of spiking perceptron capabilities to solve the XOR problem. A modified Back Propagation (BP) algorithm with adaptive learning rate and adaptive momentum was suggested in [3] to perform capacities of the standard BP and in [4, 5] author achieves an MLP minimum design based on using various forms of function activation of the hidden layer of the MLP. Although these solutions provide good results, but they need additional computations and complex frameworks. In our paper, we propose a very simple and fast solution to solve the XOR problem by using an MLP and assuming some constraints on the MLP characteristics. Our idea is based on this concept: instead to use the MLP's phase training, we bring some MLP's weights arbitrary and we try to find the rest of the weights by using the Least Mean Square algorithm.

The paper's organization is given as follows: Section II presents a general presentation of an SHLP and its different equations. In section III we explain in detail the new approach and its associated weight's computation. Simulation and obtained results are shown in Section IV, and finally, conclusion is shown in Section V.

II. SINGLE HIDDEN LAYER PERCEPTRON

An MLP is an association of multiple neurons fully connected and structured as layers. There are three kinds of layers: one input layer which receives external inputs of the network, a single output layer which represents the final outputs of the MLP and multiple hidden layers disposed between the previous layers (input and output) which transmit data from the first and the last layer. In both hidden and output layers there are a different number of neurons interconnected with associated weights. In practice, it is shown that a Single

Hidden Layer Perceptron (SHLP) with a sufficient number of neurons is adequate to do the approximation of any complex function like the XOR problem [6, 7, 8]. In our work, we use the SHLP architecture, but the study can also be extended to network with more hidden layers. Fig. 1 shows the architecture of an SHLP with *p* inputs and *m* outputs.

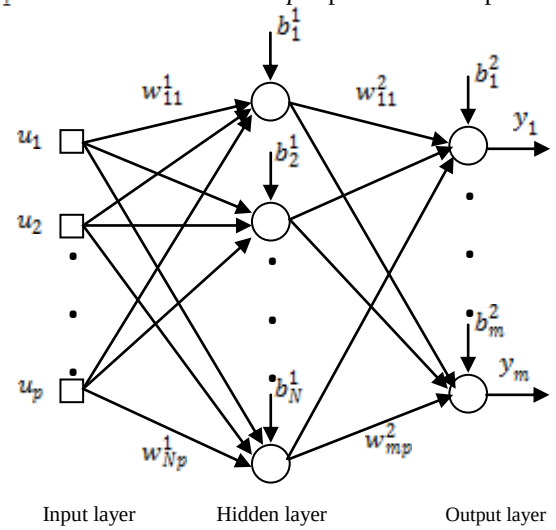

Fig. 1. Architecture of an MLP with a single hidden layer

We will use the following notation for the rest of the paper:
u_i is the i^{th} input presented to the network $(1 \le i \le p)$.
O_j is the output of the j^{th} neuron in the hidden layer $(1 \le j \le N)$
y_j is the j^{th} output of the network $(1 \le j \le m)$.
N is the number of neurons in the hidden layer.
w_{ji}^1 is the synaptic weight connecting the j^{th} neuron of the hidden layer with the i^{th} input of the network $(1 \le i \le p,\ 1 \le j \le N)$
w_{ji}^2 is the synaptic weight connecting the j^{th} neuron of the output layer with the i^{th} neuron of the hidden layer $(1 \le i \le N,\quad 1 \le j \le m)$
b_i^1 is the bias applied to the i^{th} neuron of the hidden layer $(1 \le i \le N)$
b_j^2 is the bias applied to the j^{th} neuron of the output layer $(1 \le j \le m)$
f_1 is the activation function of neurons in the hidden layer
f_2 is the activation function of neurons in the output layer.
The flow propagation of data in the MLP is given by these equations:
The j^{th} neuron output in the hidden layer is given by this equation:

$$O_j(k) = f_1\left(\sum_{i=1}^{p} w_{ji}^1(k)\,u_i(k) + b_j^1(k)\right) \qquad (1)$$

$(1 \le j \le N)$

And the j^{th} output of the network is computed by:

$$y_j(k) = f_2\left(\sum_{i=1}^{N} w_{ji}^2(k)\,O_i(k) + b_j^2(k)\right) \qquad (2)$$

$(1 \le j \le m)$.

For the XOR problem resolution, we take two inputs u_1 and u_2 with a single output y. We recall that the XOR problem can be summarized by this table:

TABLE I. TRUTH TABLE FOR THE XOR FUNCTION

u_1	u_2	y_d
0	0	0
0	1	1
1	0	1
1	1	0

The objective is to find optimal weight values of the network so his output y follows the desired output y_d.

III. NEURAL LEAST MEAN SQUARE APPROACH

For convenience to our work, we assume that our network has an input layer with two inputs and a single layer of N neurons with a sigmoid activation function like:

$$y = f_1(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \qquad (3)$$

For the output layer there is one neuron having a linear activation function:

$$y = f_2(x) = x \qquad (4)$$

Our approach called Neural Least Mean Square (NLMS) is based on this idea: we calculate the outputs of the nonlinear part of the MLP network (from the input layer to the outputs of the hidden layer). After this we use the linear part of the network (from theoutputs of the hidden layer to the final output of the network) in order to compute the weights of this linear portion using the Least Mean Square formula. Note that the real output of the MLP is not computed, but we use the desired output at its place, allowing us to try to find ideal weights that provide us the optimal MLP output.

Equations (1) can be rewritten as :

$$O_j(k) = \frac{1 - e^{-h_j(k)}}{1 + e^{-h_j(k)}} \qquad (5)$$

With

$$h_j(k) = \sum_{i=1}^{p} w_{ji}^1(k)u_i(k) + b_j^1(k) \qquad (6)$$

$(1 \le j \le N)$

Equation (2) is transformed as:

$$y(k) = \sum_{i=1}^{N} w_{ji}^2(k)O_i(k) + b_1^2(k) \qquad (7)$$

Our idea is to bring randomly first all the weights of the network, then we compute the output neurons of the hidden layer $O_j(k)$for each pattern using (5) and (6). We substitute the real output y of the network by the desired output y_d , then by rearranging (7) we will have the following equation:

$$y_d(k) - b_1^2(k) = \sum_{i=1}^{N} w_{ji}^2(k)\, O_j(k) \qquad (8)$$

By replacing the four patterns of the network inputs depicted in table I, we get:

for pattern 1:

$$y_d(1) - b_1^2 = w_{11}^2 O_1(1) + w_{12}^2 O_2(1) + ... + w_{1N}^2 O_N(1) \qquad (9)$$

for pattern 2:

$$y_d(2) - b_1^2 = w_{11}^2 O_1(2) + w_{12}^2 O_2(2) + ... + w_{1N}^2 O_N(2) \qquad (10)$$

for pattern 3:

$$y_d(3) - b_1^2 = w_{11}^2 O_1(3) + w_{12}^2 O_2(3) + ... + w_{1N}^2 O_N(3) \qquad (11)$$

for pattern 4:

$$y_d(4) - b_1^2 = w_{11}^2 O_1(4) + w_{12}^2 O_2(4) + ... + w_{1N}^2 O_N(4) \qquad (12)$$

By writing these last equations in a matrix form, we obtain:

$$\underbrace{\begin{bmatrix} O_1(1) & O_2(1) & \cdot & \cdot & O_N(1) \\ O_1(2) & O_2(2) & \cdot & \cdot & O_N(2) \\ O_1(3) & O_2(3) & \cdot & \cdot & O_N(3) \\ O_1(4) & O_2(4) & \cdot & \cdot & O_N(4) \end{bmatrix}}_{O} \underbrace{\begin{bmatrix} w_{11}^2 \\ w_{12}^2 \\ \cdot \\ \cdot \\ w_{1N}^2 \end{bmatrix}}_{W} = \underbrace{\begin{bmatrix} y_d(1) - b_1^2 \\ y_d(2) - b_1^2 \\ y_d(3) - b_1^2 \\ y_d(4) - b_1^2 \end{bmatrix}}_{b} \qquad (13)$$

The above matrix form can be written by this equation:

$$OW = b \qquad (14)$$

where:

O : is a (4xN) matrix corresponding to the output neurons of the hidden layer for all patterns.

W : is a vector column containing N elements corresponding to the weights connections between hidden and output layers $\left(W = \left[w_{11}^2, w_{12}^2, ..., w_{1N}^2\right]^T\right)$.

b : is a vector column containing N elements corresponding to the difference between desired outputs for each pattern and the bias of the output neuron of the MLP network.

Our purpose is to find optimal values of the weights w_{1i}^2 $(1 \le i \le N)$, thus (14) can be transformed as a least square problem of the form $Ax = b$ which the objective is to minimize :

$$min_x \|Ax - b\| \qquad (15)$$

Where : A = O and x = W

To approximate the solution x, we can apply the backslash operator (\ ) like:

$$W = O \setminus b \qquad (16)$$

Or this simple formula:

$$W = O^T \left(OO^T\right)^{-1} b \qquad (17)$$

The proposed NLMS algorithm is summarized in the different steps below:

Step 1
Initialize randomly the weights w_{ji}^1 , $b_i^1$and b_1^2
$(1 \le i \le 2, \quad 1 \le j \le N)$

Step 2
For k=1 to 4
Present the k^{th} pattern and calculate outputs $O_j(k)$ of hidden neurons
$(1 \le i \le N)$
End For

Step 3
Construct the matrix O and the vector b from (13)

Step 4
Compute weights w_{1i}^2
$(1 \le i \le N)$
from (16) or (17)

Note that we will not update the weights:w_{ji}^1 , $b_i^1$and b_1^2 but only compute w_{1i}^2

IV. SIMULATIONS AND RESULTS

In order to show the effectiveness of our approach we have compared it with the Standard BackPropagation (SBP). Weights and biases were initialized to random values which were taken in the range of [-1, 1]. The same initial values of weights and biases are used in each algorithm in order to compare between them. Simulations are done with various numbers of hidden neurons. The learning rate is taken equal to the value 0.1 in the SBP algorithm. The goal training network is defined to achieve a value of Mean Squared Error (MSE) equal to 10^{-10}. Results of the simulations are shown in table II.

TABLE II. OBTAINED RESULTS FOR THE XOR PROBLEM

		SBP		NLMS	
		MSE	epochs	MSE	epochs
Number of hidden neurons	04	**$1.29*10^{-11}$**	**262**	**$5.68*10^{-29}$**	**1**
	06	**$1.16*10^{-11}$**	**220**	**$3.64*10^{-30}$**	**1**
	08	**$1.02*10^{-11}$**	**203**	**$3.16*10^{-30}$**	**1**
	10	**$9.97*10^{-12}$**	**168**	**$1.54*10^{-29}$**	**1**
	12	**$8.39*10^{-12}$**	**149**	**$1.52*10^{-31}$**	**1**

From results of table II we can conclude that all algorithms successfully solve the XOR problem, but there are some differences between them resumed as:

MSE performance

The SBP algorithm reaches a very small value of MSE for any number of neurons in the hidden layer. Furthermore, we can see that for the NLMS approach, the MSE value is extremely reduced compared with other algorithms.

Iterations

Concerning the number of iterations (epochs) needed to achieve the desired values; we note that for the SBP algorithm it needs an important number, although it decreases with the number augmentation of neurons. But for the NLMS approach and for any number of neurons selected, we will still need only one iteration.

Weights adaptation

If the SHLP network has N neurons in the hidden layer, then the number of weights to be adapted will be equal to (4N + 1), and their adjustment is done at each iteration for each algorithm. While for the NLMS algorithm, there is no weight adjustment, it only calculates, in a single iteration, N weights connecting neurons of the hidden layer to the output neuron, the rest of the weights being unchanged.

Calculation Time

As we said in the previous point, adaptation of all the weights and for all iterations requires a relatively large time calculation. On the other hand the NLMS approach calculations are performed for a single iteration and a reduced number of weights.

According to the obtained results, we can conclude that our proposed method for solving the XOR problem is better compared to the conventional learning algorithm SBP. However, we have to underline that to assure good results the choice of neuron number that must not be less than four (the number of examples), because in this case we would have an overdetermined system.

Our work in this paper has dealt with a well-known problem that is the approximation of the XOR function. Nevertheless, the idea presented in this paper can be extended to other problems such as the parity-N problem, and so we must adapt the architecture of the neural network according to our problem

V. CONCLUSION

The famous XOR problem is among the most problems solved by neural networks. Many training algorithms have been used in literature to find optimal weight values of the network, so this last can approximate the XOR function. In this paper, we proposed a new and very simple procedure to solve the XOR problem and guarantee better results compared with classical training algorithms. The main advantage of this new approach is based on the fact that we didn't adapt all network's weights, but we keep unchanged some weights of the network and compute the remaining weights by using the Least Mean Square formula. Furthermore, our proposed algorithm does not require many epochs to achieve the goal, but only one iteration is sufficient.

REFERENCES

[1] U. Seiffert, "Multiple layer perceptron training using Genetic Algorithms", in *Proc. 9*th *European Symposium on Artificial Neural Networks ESANN*, pp. 159-164, *2001*.

[2] J.Yang, W.Yang, W. Wu, "A novel spiking perceptron that can solve XOR problem", Neural Network World, pp. 45-50, 2011.

[3] M.A.W. Saduf, "Improving learning efficiency by Adaptively Changing Learning Rate and Momentum", International Journal of Advance Foundation and Research in Science & Engineering IJAFRSE, vol. 1, no. 3, pp. 32-39, 2014.

[4] K. S. Vaibhav, "One Solution to XOR problem using Multilayer Perceptron having Minimum Configuration", International Journal of Science and Engineering, vol. 3, no. 2, pp. 32-41, 2015.

[5] K. S. Vaibhav, "Proposing Solution to XOR problem using minimum configuration MLP", International Conference on Computational Modeling and Security, Procedia Computer Science vol. 85, pp. 263-270, 2016.

[6] G. Panchal, A. Ganatra, Y. P. Kosta, D. Panchal, "Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers", International Journal of Computer Theory and Engineering, vol. 3, no. 2,pp. 332–337, 2011.

[7] Qamar, H. B. Wagas, A. Jamil, "The Impact of Training Iterations on ANN Applications Using BPNN Algorithm", International Journal of Future Computer and Communication, vol. 2, no. 6, pp. 567-569, 2013.

[8] K. Funahashi, "On the approximate realization of continuous mappings by neural networks", *Neural Networks*, vol. 2, pp. 183-192, 1989.