Metaplasticity Artificial Neural Networks Model Application to Radar Detection

Diego Andina and Juan Fombellida
Universidad Politécnica de Madrid, Spain
d.andina@gc.ssr.upm.es
j.fombellida@gc.ssr.upm.es
Grupo de Automatización en Señal y Comunicaciones
GASC/UPM

ABSTRACT—Many Artificial Neural Networks design algorithms or learning methods imply the minimization of an error objective function. During learning, weight values are updated following a strategy that tends to minimize the final mean error in the Network performance. Weight values are classically seen as a representation of the synaptic weights in biological neurons and their ability to change its value could be interpreted as artificial plasticity inspired by this biological property of neurons. In such a way, metaplasticity is interpreted in this paper as the ability to change the efficiency of artificial plasticity giving more relevance to weight updating of less frequent activations and resting relevance to frequent ones. Modeling this interpretation in the training phase, the hypothesis of an improved training is tested in the Multilayer Perceptron with Backpropagation case. The results show a much more efficient training maintaining the Artificial Neural Network performance.[1]

Key Words: Neural Networks, Backpropagation Training Algorithm, Metaplasticity, Binary Detection

1. INTRODUCTION

The idea proposed is to improve the basic error minimization algorithm used to train an Artificial Neural Network (ANN) [1] manipulating the error objective function in order to give more relevance to the less frequent training patterns generated around the threshold value and to subtract relevance to the frequent ones. So, if the objective is to minimize an expected error E_M defined by the following expression:

$$E_M = \varepsilon\left\{E(x)\right\} \qquad (1)$$

where X is a random variable of the training input vectors $x = (x_1, x_2, ..., x_n)$, $(x \in R^n)$, where R^n is the n-dimensional space and *E(x)* is the expression of a given error criterion as a function of the inputs applied in ANN training, then

$$E_M = \int_{R^n} E(x)\, f(x)\,\partial x = \int_{R^n} e(x)\,\partial x \qquad (2)$$

$$E_M = \int_{R^n} \frac{e(x)}{f_X^*(x)}\, f_X^*(x)\,\partial x \qquad (3)$$

From statistical inference theory applied to eq. (3), an estimator of E is given by:

$$\hat{E}_M = \frac{1}{M}\sum_{k=1}^{N} \frac{e(x_k)}{f_X^*(x_k)} \qquad (4)$$

where $x_k^*, k = 1, 2, ..., N$, are independent sample vectors whose *pdf* is $f_X^*(x)$, that we call Weighting Function and $f_X^*(x)$ can be arbitrarily chosen by the designer if $f_X^*(x) \neq 0$ wherever $e(x) \neq 0, \forall x \in R^n$. Note that from eq. (3) $f_X^*(x)$ is ideally given by [1]:

$$\left(f_X^*(x)\right)_{opt} = \frac{1}{E_M} e(x) \qquad (5)$$

2. WEIGHTING OPERATION

What lies in eq. (4) is that an error objective function $E(x_k)$ can be weighted by a proper function $w^*(x_k)$ without affecting the final error objective. In fig. 1 we present a block diagram for the Weighted Training.

On the hypothesis that by giving more relevance in weight update to less frequent activations and resting relevance to frequent ones, Metaplasticity [2][3]is being modelled and therefore training can be improved making it faster and reducing the number of patterns necessary to complete the training. We decide to train a classifier and test for the case of a Multilayer Perceptron (MLP) the following weight functions:

$$w_{\overline{X}}^*(x) = A\sqrt{(2\pi)^{N_1}} \cdot e^{B\sum_{i=1}^{8} x_i^2} \qquad (6)$$

[1] [1]This research has been supported by the National Spanish Research Institution "Comisión Interministerial de Cienciay Tecnología-CICYT"as part of the project **AGL2006-12689/AGR**

and

$$w^*_{\bar{X}}(x) = C\hat{y} \qquad (7)$$

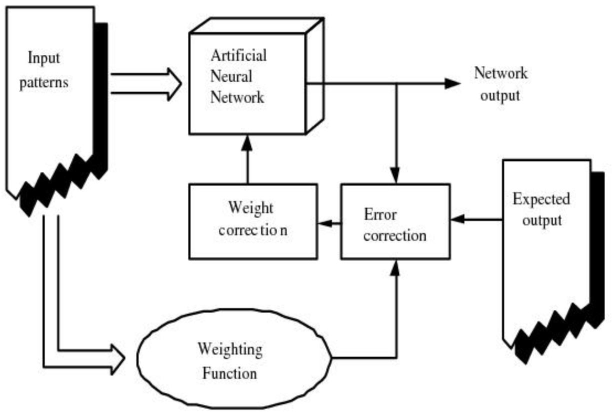

Fig 1. Weighted training cycle

In eq. (6) an inverse Gaussian function is proposed as weighting function as a standard assumption for the weighting function. In eq. (7) the network output is used and advantage is taken from the inherent a posteriori probabilities estimation for each input class of MLP outputs, so the statistical distribution of training patterns is used to quantify how frequent a pattern is. *A*, *B*, and *C* are parameters to be adjusted according to the training to converge, this parameters should be optimized depending on the application of the classifier, in our case we want to use the neural networks to classify Radar patterns. N_1 represents the number elements in the hidden layer of the MLP.

3. COMPUTER RESULTS

Experiments have been carried out in order to evaluate the Backpropagation with Weighting (BPW) algorithm [4]. We present the results obtained from training groups of 100 MLPs applying a BPW algorithm consisting in Least Mean Square (LMS) criterion modified by the proposed weighting functions.

3.1 General Characteristics of the Experiments

The ANNs applied are MLPs with structure 16/8/1 (that is 16 inputs, and one hidden layer of 8 units). The choice of the structure and the rest of the parameters of the network was the optimal solution for the given example application [1][5]. The activation function is sigmoidal logarithmic with scalar output in the range (0,1) and it is the same for all the neurons.

For the training of the network we used balanced patterns of two classes, being class H_0 noise patterns and being class H_1 signal received with additive Gaussian noise. These patterns configure the problem of signal detection noise and the ANN acts as a binary detector. The application of the ANN is an elemental radar detection problem [5][6] when the basic parameter for the patterns is the Signal-to-Noise ratio, *SNR*, and the performance of the detectors is evaluated in terms of the Neyman-Pearson criterion. That is, maximizing probability of detection, P_d, for a fixed false alarm probability, P_{fa}. In the radar literature, performance is evaluated through the Detection curves (P_d vs. *SNR*), so we use the comparison between these detection curves to present the results of our method.

In each experiment 100 networks were trained in order to achieve mean results that does not depend on initial random value of the weights of the ANN. Fig. 2 shows the error evolution comparison of the network trained with BPW and classical BP training. Error is calculated as the rate of misclassified patterns of the test set out of the total number of patterns. The BPW training algorithm requires much less iterations to consider an ANN trained than the classical BP does, which shortens the total time of training.

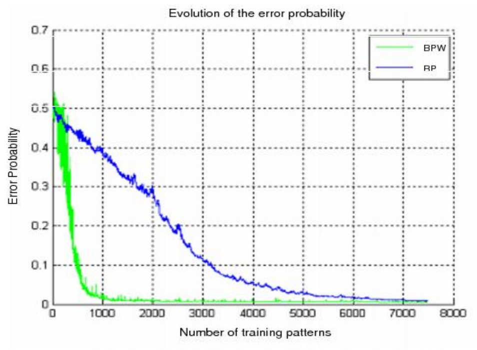

Fig 2. Classification error in training phase of BP and BPW

In the following experiments, two different criterions were applied to stop the training: in one case it was stopped when the classification error reached zero (denoted as is_m) and in the other the training was conducted with a fixed number of 3000 patterns ($3is_m$). When inverse Gaussian function (6) was applied as Weighting function, the training was conducted as usual BP training, maintaining the expression of the error function to be minimized from the beginning to the end of the training. Fig. 3 shows an example of NN training using only weighting function (6). But in the case of (7) the weighting function is not valid until the output of the network is a sufficiently good approximation of the *a posteriori* probabilities of the inputs. In the first iterations $\hat{y}$ can tend to values very close to zero, the training curve evolution freezes around error classification value of 0.1 or 0.2 and the MLP does not learn. So, in this second case, function (6) was applied till the error probability achieved a value in the range of 0.1-0.2 and then switched to function (7) till the end of training. Fig. 4 shows the error evolution during the network training phase for the second case. As usual, three set of patterns have been used to design the network. A training set (composed of patterns of $SNR = 13.2dB$ for class H_1), a test set to calculate the error during training and a validation set to obtain the detection curves.

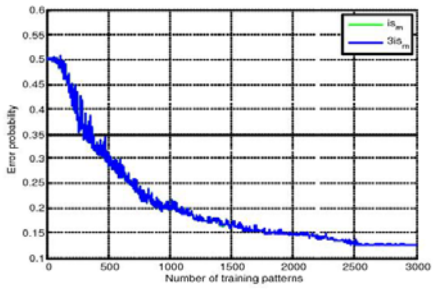

Fig 3. Classification error in training phase with only one weighting function. 3is$_m$ and is$_m$ have the same evolution.

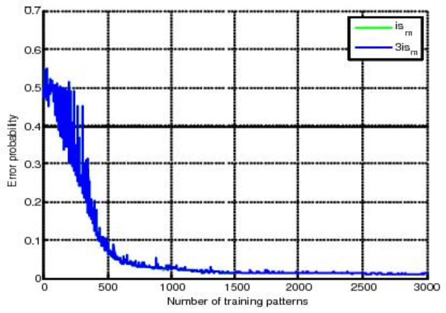

Fig 4. Classification error in training phase, threshold 0.2. 3is$_m$ is$_m$ have the same evolution.

The detection probability for three different false alarm probability (probability of "decide H_0 when input corresponds to H_1") values related to the SNR are shown in fig. 5, 6 and 7, respectively.

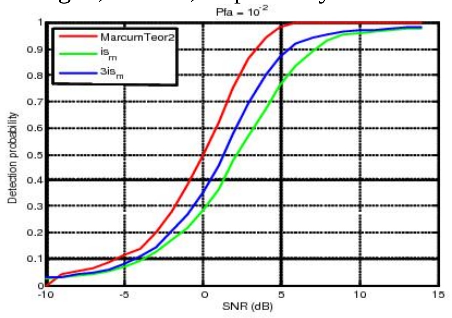

Fig 5. Detection probability, $P_{fa}=10^{-2}$, threshold 0.2.

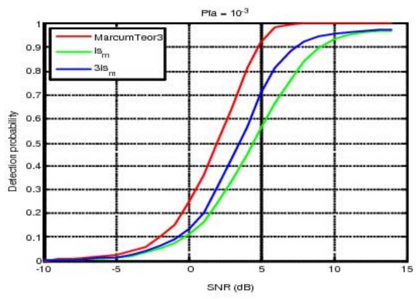

Fig 6. Detection probability, $P_{fa}=10^{-3}$, threshold 0.2.

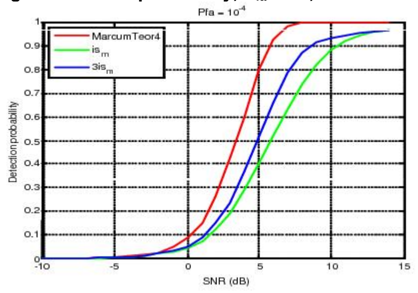

Fig 7. Detection probability, $P_{fa}=10^{-4}$, threshold 0.2.

The red line represents the theoretical maximum curve by Marcum theorem [6]. The green line represents average performance for the networks that were trained until the error probability reached zero and the blue line is used for the networks trained with the fixed number of patterns. False alarm probabilities, $P_{fa,}$ of 10^{-2}, 10^{-3} and 10^{-4} have been considered. For the detection probability that corresponds to the false alarm probability of 0.01, we find that the results are noticeably better if the NNs were trained with the fixed number of patterns (3000) for all the values in relation to the SNR between 0 and 8 dB. In the case of false alarm probability of 0.001 and 0.0001 we also get better results for training a network with the fixed number of patterns and the curve (blue) is much closer to the theoretical one (red). For the high SNR values the results could be improved, which could make a part of the future lines of investigation for this application.

Fig. 8 shows the results obtained for setting the threshold for changing the weighting functions at 0.15. Again, we considered two criterions for stopping the training of a network. We can see that the decision to change the weighting function when the threshold 0.15 was reached gave also satisfactory results. Again, the experiments were carried out for false alarm probability values 10^{-2}, 10^{-3} and 10^{-4}. The results obtained are better in the case of training a network with the fixed number of patterns, as it was with the threshold of 0.2. This seems to show that the optimum point to switch weighting functions is a matter of study.

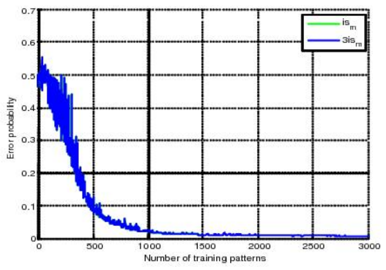

Fig 8. Classification error in training phase, threshold 0.15. 3is$_m$ is$_m$ have the same evolution.

3.2 The Best Obtained Network

The error probability evolution of the best network obtained is shown in fig. 9. Only 355 iterations were needed to reach the zero classification error. We can see that the network has a rapid error evolution to the zero value, with a low number of iterations. This allows us to save time and computing resources. The threshold for changing the weighting function was in this case set to 0.2.

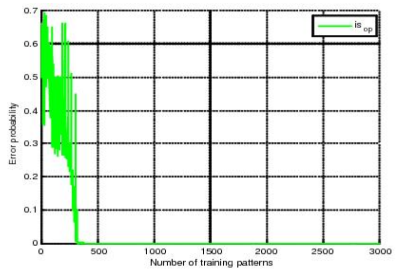

Fig 9. Classification error in training phase, threshold 0.2, the best case.

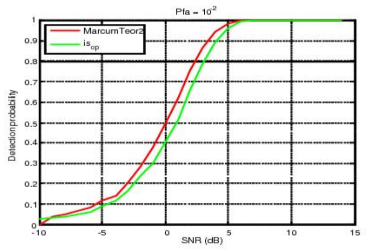

Fig 10. Detection probability, P$_{fa}$=10^{-2}, threshold 0.2, the best case.

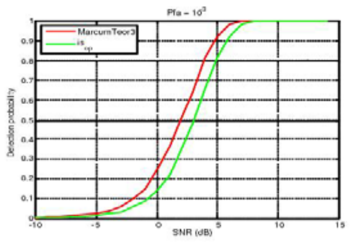

Fig 11. Detection probability, P$_{fa}$=10^{-3}, threshold 0.2, the best case.

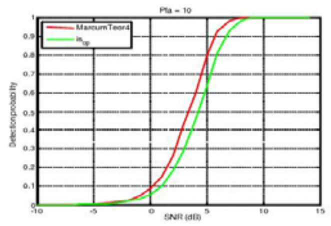

Fig 12. Detection probability, P$_{fa}$=10^{-4}, threshold 0.2, the best case

Fig. 10, 11 and 12 show the characteristics of trained network for false alarm probabilities, P_{fa} , of 10^{-2}, 10^{-3} and 10^{-4} ,respectively. We can see that the distance between two curves is less than 1 *dB*. Even though the number of iterations used was small, we could continue the training with a fixed number of patterns and obtain values even closer to the theoretical maximum. These results support, one more time, the superiority of the performance of **NNs** trained applying **BPW** criterion with two weighting functions.

4. CONCLUSIONS

We test the hypothesis that weighting the error objective function giving more relevance to less frequent training patterns and subtracting relevance to frequent ones is a way to model of metaplasticity biological properties in artificial neurones. We apply the statistical distribution of training patterns to quantify how frequent a pattern is in an application of MLP with error Backpropagation training, finding that Weighting training requires much less training patterns maintaining the ANN performance.

REFERENCES

1. Andina D. Pham D.T. (Eds)., "Computational Intelligence for Engineering and Manufacturing". Springer-Verlag, Germany, March 2007.
2. Godoy Simoes M. and Ropero Pelaez J., "A computational model of synactic metaplasticity", *Proceedings of the International Joint Conference on Neural Networks*, 1999.
3. P. Friedich and P. Tompa, "Synaptic metaplasticity and the local charge effect in postsynaptic densities", *Trends in Neuroscience, 3 (21):pp 97-101*, May 1998.
4. Andina D. Jectic A., "Improved multilayer perceptron design by weighted learning", *Proc. Of the 2007 IEEE International Symposium on Industrial Electronics ISIE 2007, Vol.(15):6pp.*, Vigo Spain, June 2007.
5. Alarcón M.J., Torres-Alegre S. and Andina D. "Behaviour of a binary detector based on

an ann digital implementation in the presence of seu simulations", *Proc. of III World Multiconfernce on Systemics, Cybernetics and Informatics (SCI'99) and 5th International Conference on Information Systems Analisis and Synthesis (ISAS'99), Vol.(3):pp. 616-619*, Orlando, Florida, USA, July-August 1999.

6. Marcum J. "A statistical theory of target detection by pulse radar", *IEEE Transactons on Information Theory April 1960 Vol.(6,Issue2):pp.59-26 7 April 1960.*