# MODIFIED HEBBIAN LEARNING RULE FOR SINGLE LAYER LEARNING

J. M. Zurada, A. Malinowski, P. Przestrzelski†

Department of Electrical Engineering, University of Louisville, Louisville, Kentucky 402922
e-mail: jmzura02@ulkyvx.louisville.edu

†Faculty of Electronics, Technical University of Gdansk, 80-395 Gdansk, Poland

**Abstract - The paper introduces a new learning method based on the supervised Hebbian learning of encoded associations. The learning is performed with weight initialization of any values and with invariable learning constant. The method is suitable for training of single-layer networks and compares favorably with other conventional learning rules discussed in the paper.**

## I. INTRODUCTION

Single-layer neural networks are usually trained using generalized delta training rules, Hebbian learning rule or competitive unsupervised winner-take-all training [1-2]. This paper introduces a new form of supervised Hebbian learning formula which is characterized by rather quick convergence. This feature can be helpful especially for large networks and for large amount of input data. In addition, initial weights in the proposed rule do not have to be assumed as in the case in the Hebbian rule near zero. In the paper, the rule is implemented for case of three training point clusters and compared with other rules suitable for providing the classification of the training set data.

The modified supervised Hebbian learning rule is based on the supervised Hebbian (correlation) learning which will be reviewed first [3].

## II. SUPERVISED HEBBIAN LEARNING

Let us revise the Hebbian learning rule [1-3] for a single continuous neuron described with the activation function

$$o = f(net) = \frac{2}{1 + e^{-\lambda net}} - 1 \qquad (1)$$

where the activation and input and weights vectors are, respectively

$$net = w^T x$$

$$x = [x_1, x_2, ..., x_N]^T$$

$$w = [w_1, w_2, ..., w_N]^T$$

In the classical Hebbian rule learning neuron is taught by modifying its weight vector w after every input pattern using the formula

$$\Delta w = \eta x o \qquad (2)$$

where $\eta$ denotes a positive learning constant. Initial weights cannot all be equal to zero but should be close to it. The response of the neuron to the present input pattern needs to be evaluated every time its weights are updated. Neurons taught that way with patterns from several classes or clusters usually learns how to recognize one of them, but because of the random initial weights it cannot be determined which particular set of patterns is recognized. We thus can only obtain learning of the data without their proper labeling. To achieve training with labeling, a supervised Hebbian learning of the neuron is performed, and its weights change according to the following formula

$$\Delta w = \eta x d \qquad (3)$$

where d denotes the desired output to the input pattern x, and $\eta$ is the learning constant. While using this method we no longer need to evaluate neuron's responses to the training patterns no matter if we train a single neuron or single-layer network.

Let a neuron be trained by a set of P patterns $\mathbf{x}_1,...,\mathbf{x}_P$, and the desired outputs for those pattern be known as $\mathbf{d}_1,...,\mathbf{d}_P$. The learning cycle consists then of steps as follows :

$$w^0 \text{ are initial weights}$$

$$w^1 = w^0 + \eta x_1 d_1$$

$$w^2 = w^1 + \eta x_2 d_2 = w^0 + \eta x_1 d_1 + \eta x_2 d_2$$

$$...$$

$$w^P = w^{P-1} + \eta x_P d_P = w^0 + \eta \sum_{k=1}^{P} x_k d_k \qquad (4)$$

After completing the n-th learning cycle the weights become

$$w^{nP} = w^0 + n\eta \sum_{k=1}^{P} x_k d_k \qquad (5)$$

0-7803-1254-6/93$03.00 © 1993 IEEE

Let us see that for large enough n we can usually neglect w for

$$|w_i^0| \ll n\eta \left| \sum_{k=1}^{P} x_{k,i} d_k \right|, \qquad i = 1, \ldots N$$

what is true if only the sum is not equal to zero. However, for the same reason there is a possibility that weights will increase until an overflow occurs during learning. To prevent this, the learning constant $\eta$ should be decreased after each cycle, e.g.

$$\eta^{(n)} = \alpha^{n-1}\eta^{(0)}, \quad \alpha \text{ is a constant } \varepsilon\ (0,1) \qquad (6)$$

In this case we can evaluate the limit of the weights :

$$lim_{n\to\infty} w^{nP} = w^0 + \frac{\eta}{1-\alpha}\sum_{k=1}^{P} x_k d_k \qquad (7)$$

It can now be seen that the initial weights of non-zero value may introduce a significant error in the final value independent of learning process. However, since we use formula (3) instead of (2) non-zero weights are no longer necessary, because we no longer use the network response for its training.

## III. Modified Supervised Hebbian Learning

We now propose the modification of the supervised learning rule as described above. Our modification causes improvement of convergence of the learning process so that we can terminate it after shorter period. Let us consider the following formula for weights adjustment

$$\Delta w = \frac{P-1}{P} w + \frac{1}{P}\eta x d \qquad (8)$$

where P is total number of input patterns which neuron shell recognize after the training. As previously, the learning cycle consists of P steps as follows

$$w^0 \text{ are initial weights}$$

$$w^1 = \frac{P-1}{P} w^0 + \frac{\eta}{P} x_1 d_1$$

$$w^2 = \frac{P-1}{P} w^1 + \frac{\eta}{P} x_2 d_2 =$$

$$w^2 = \left(\frac{P-1}{P}\right)^2 w^0 + \frac{P-1}{P}\frac{\eta}{P} x_1 d_1 + \frac{\eta}{P} x_2 d_2$$

$$w^3 = \frac{P-1}{P} w^2 + \frac{\eta}{P} x_3 d_3 =$$

$$w^3 = \left(\frac{P-1}{P}\right)^3 w^0 + \left(\frac{P-1}{P}\right)^2 \frac{\eta}{P} x_1 d_1 + \frac{P-1}{P}\frac{\eta}{P} x_2 d_2 + \frac{\eta}{P} x_3 d_3$$

$$\cdots$$

$$w^P = \frac{P-1}{P} w^{P-1} + \frac{\eta}{P} x_P d_P =$$

$$w^P = \left(\frac{P-1}{P}\right)^{P-1} w^0 + \frac{\eta}{P}\sum_{k=0}^{P-1}\left(\frac{P-1}{P}\right)^k x_{P-k}\, d_{P-k} \qquad (9)$$

After completing the n-th learning cycle the weights become This result has been proven using the

$$w^{nP} = \left(\frac{P-1}{P}\right)^{P-1} w^0 + \\ + \sum_{l=0}^{n-1}\left(\frac{P-1}{P}\right)^{lP}\left(\frac{\eta}{P}\sum_{k=0}^{P-1}\left(\frac{P-1}{P}\right)^k x_{P-k}\, d_{P-k}\right) \qquad (10)$$

mathematical induction, and the proof is omitted here for brevity. If we consider the limit of the training after infinite number of cycles we obtain

$$lim_{n\to\infty} w^{nP} = 0w^0 + \\ + \sum_{l=0}^{\infty}\left(\frac{P-1}{P}\right)^{lP}\left(\frac{\eta}{P}\sum_{k=0}^{P-1}\left(\frac{P-1}{P}\right)^k x_{P-k}\, d_{P-k}\right) \qquad (11)$$

which simplifies to

$$lim_{n\to\infty} w^{nP} = \eta\sum_{k=0}^{P-1}\left(\frac{P-1}{P}\right)^k x_{P-k}\, d_{P-k} \qquad (12)$$

Let us note that in contrast to the previously discussed learning rule in which learning constant $\eta$ was decreasing, here it must be fixed, otherwise the limit as in (12) will be equal zero. Furthermore, we can see another effect: if the training is terminated in the middle of the cycle, resulting weights are not a solution because for large enough cycle number n we have

$$w = \eta\sum_{k=0}^{P-1}\left(\frac{P-1}{P}\right)^k x_{P-k}\, d_{P-k} + \\ + \eta\sum_{k=0}^{b}\left(\frac{P-1}{P}\right)^k x_{P-k}\, d_{P-k} \qquad (13)$$

where b is a number of step within the learning cycle after which we stopped learning. It can be seen from (12), that the weights oscillate periodically around the solution and are close to it only after the end of each cycle. This is apparently the only disadvantage of the method which has to be paid for the very rapid training convergence of this algorithm.

## IV. Experimental Results and Conclusions

Simulation of learning efficiency using the new modified supervised Hebbian rule has been tested on a number of cases, including pattern classifier described below. Non-augmented continuous inputs have been fed to a single-layer network with bipolar neurons having continuous characteristics. The new algorithm has shown much better performance than both a regular supervised Hebbian learning and competitive unsupervised learning which have been used in the comparative study. The same learning constants $\eta$=0.1 were used for all algorithms, $\alpha$=0.8 was used for decreasing $\eta$ in the first algorithm.

Results of these three experiments are shown on figures below. On Fig.1 and Fig.2 changes of weights of one of the neurons are presented (the slowest one) during all steps of

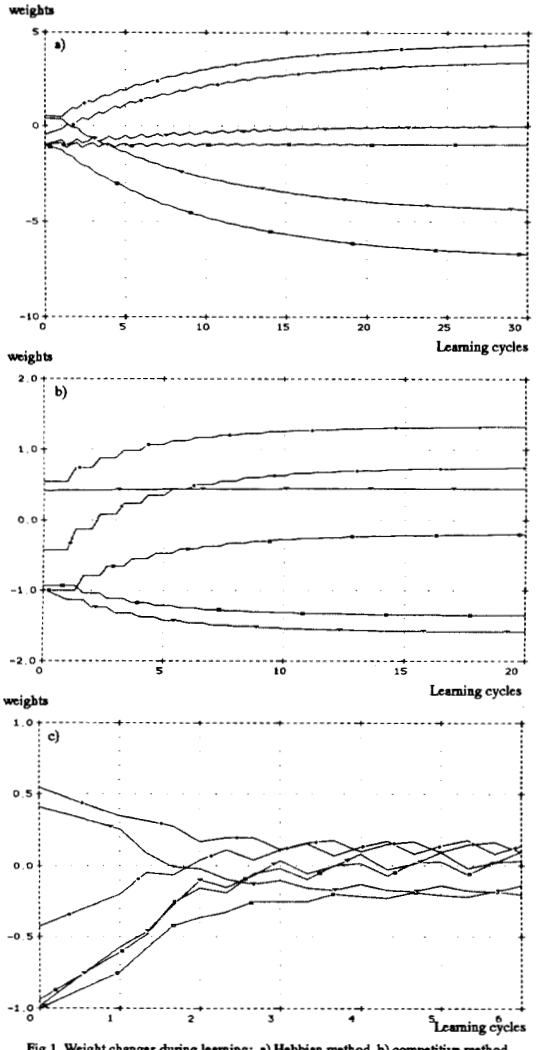


Fig.1. Weight changes during learning: a) Hebbian method b) competitive method c) modified Hebbian method

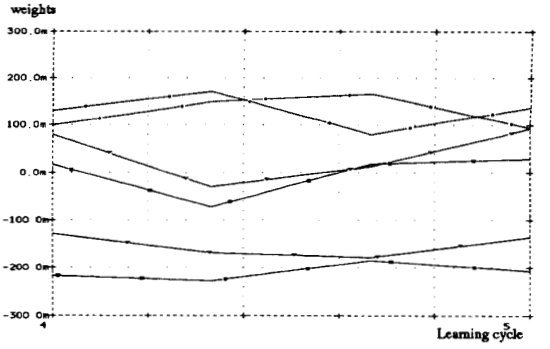


Fig.2. Weight changes during one cycle of modified Hebbian learning.

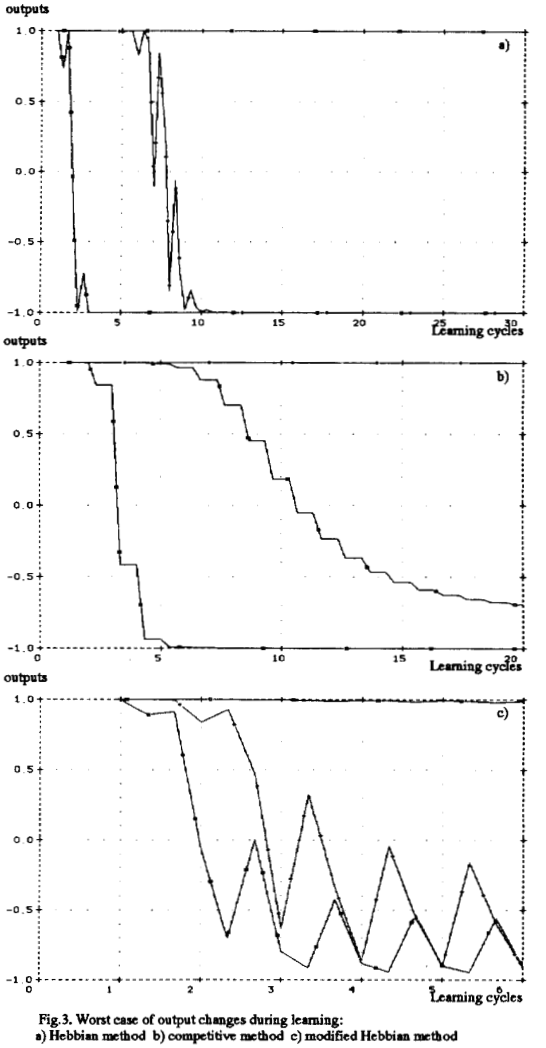


Fig.3. Worst case of output changes during learning:
a) Hebbian method b) competitive method c) modified Hebbian method

learning cycles. Fig.1a illustrates the case of supervised Hebbian learning, Fig.1b unsupervised competitive learning, Fig.1c and Fig.2 are for modified Hebbian learning rule. It can be seen from Fig.2 that weights are changing periodically every training cycle.

The well-trained networks from Fig.1a and Fig.1c will respond every time to a pattern from one set by activating only one neuron, responsible for that particular data. The network from Fig.1b, trained according to the winner-take-all algorithm shall respond with one neuron with the highest activation than others, and therefore, negative output of other neuron is not required.

Clearly, the movement of weights is not as important as the overall output performance of network. Fig.3 and Fig.4 illustrate the output of one of the neurons for each method tested after each learning step during the cycles using one of the learned patterns. The graphs highlight the case of the slowest learning. It can be seen that both Hebbian and competitive learning are worse than the proposed algorithm while tested at the end of each training cycle. The new modified Hebbian learning rule produces the well trained network after the fourth learning cycle. On Fig.4 it can be seen that for best results learning preferably should be stopped at the end of a learning cycle.

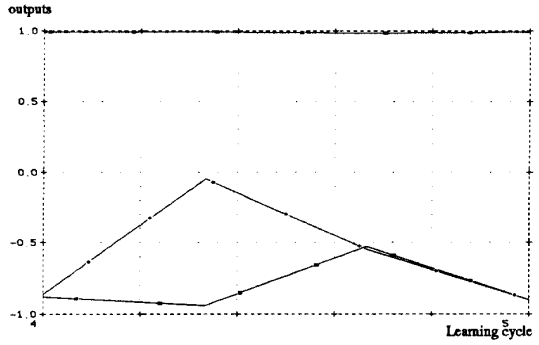


Fig. 4. Worst case of output changes during one cycle of modified Hebbian method.

The end-of-training condition can be specified as unchanged network responses to the training patterns at the end of a cycle in comparison to response at the previous end of cycle compared with certain accuracy. Outputs stable in this sense indicate that learning is complete.

The new algorithm has also been tested for the case of unsupervised Hebbian learning phase which follows the supervised training phase. It can be observed on Fig.5a that a well trained neuron changes his response during unsupervised learning performed with initial weights obtained from supervised experiment. Fig.5b presents another neuron which has improved its performance during unsupervised learning.

The new supervised learning algorithm is numerically less complex than most classical supervised learning algorithms because no need exists here for evaluating network response in every learning step. It is about of the same complexity as supervised Hebbian learning but it shows, in general, faster convergence. It represents therefore, a valuable modification of Hebbian learning, especially for large layers trained on large number of input patterns.

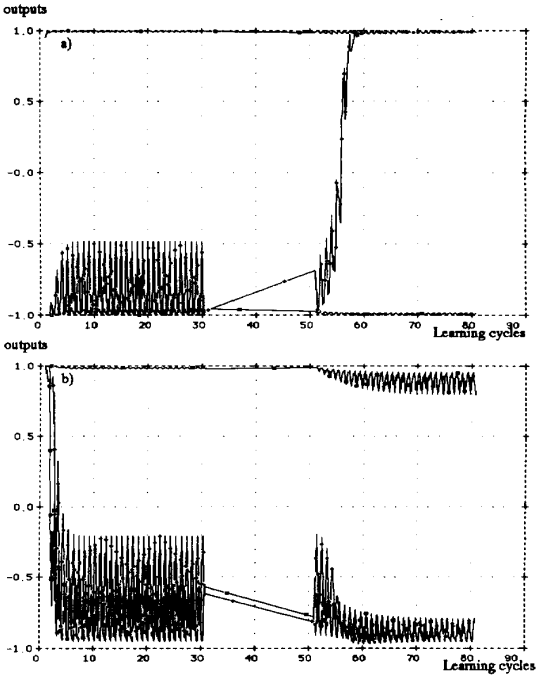


Fig.5. Neurons' output to one of the patterns - unsupervised learning following supervised learning.

## REFERENCES


[1] S. I. Amari, Mathematical Foundations of Neurocomputing, Proc. IEEE, vol. 78, No. 9, Sept. 1990, pp.1443-1463

[2] J. Hertz, A. Krough, R. G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley Publishing Company, Redwood City, Calif., 1991.

[3] J. M. Zurada, Introduction to Artificial Neural Systems, West Publishing Company, St. Paul, Minn., 1992.