PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 37, NO. 3, PP. 237–250 (1993)

TRAINING ALGORITHMS FOR THE SINGLE LAYER PERCEPTRON

Noury ELHADI and Klára CSÉFALVAY

Department of Electromagnetic Theory
Technical University of Budapest
H–1521 Budapest, Hungary

Received: Sept. 20, 1993

Abstract

The perceptron is essentially an adaptive linear combiner with the output quantized to one of two possible states, it is the basic building block of multilayer, feedforward neural networks. This paper describes the learning algorithms for the perceptron. Each algorithm is viewed as a steepest descent method, where the algorithm iteratively minimizes an instantaneous performance function. A new performance function is introduced, and a new algorithm is developed that increases the learning speed. Advantages of the new algorithm are demonstrated in computer experiments.

Keywords: neural networks, adaptive linear element, error correction rules, steepest descent rules.

Introduction

The perceptron can be represented schematically in the form of an array of multipliers and summing junction (WIDROW and LEHR, 1991; AMARI, 1990). In the case of a single node perceptron, the weighted sum of the input vector $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ is computed, or in other words we scalar product the input vector $\mathbf{X}$ and the weight vector $\mathbf{W}$. The output $\mathbf{Y_q}$ is determined by passing the linear output $\mathbf{Y}$ through a hard limiting non-linearity.

The perceptron essentially operates as a pattern classification device. It has the ability to learn to discriminate between input vectors. As shown in *Fig. 1*, it can decide between two classes A and B depending upon the value of the output. The perceptron can perform linear classification, however, the single layer perceptron is limited in its ability to perform certain

$$\mathbf{Y_q} = f\left(\sum_{i=1}^{N} w_i x_i - \Theta\right) \quad \text{where } f(x) = \begin{cases} 1, & \text{if } x > 0 \\ -1, & \text{if } x < 0 \end{cases} \tag{1}$$

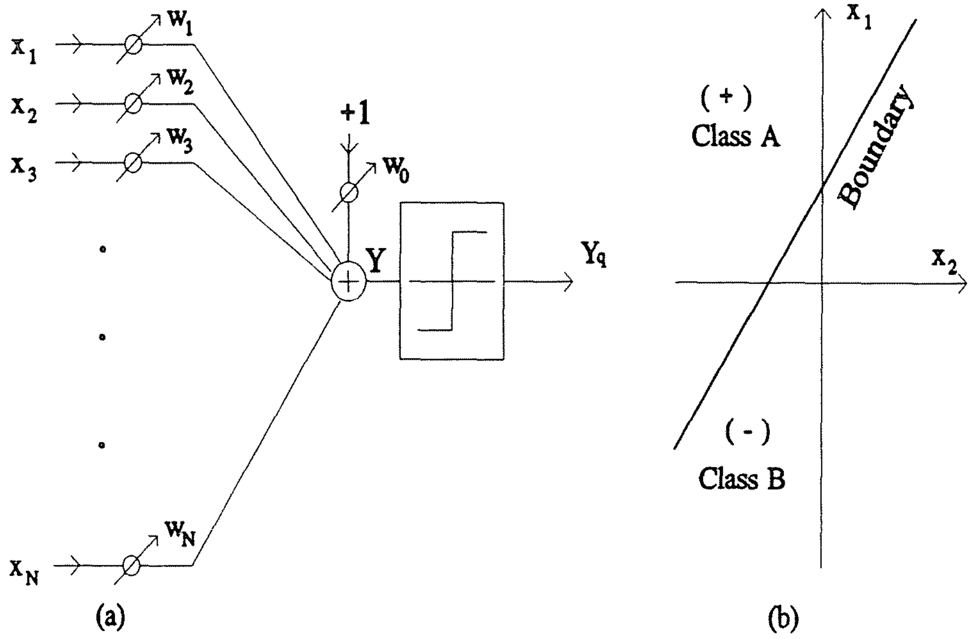

Fig. 1. (a) The perceptron, (b) Decision boundary in the case of a two dimensional input vector

classifications. The weight modification is done by different rules, which will be described later. This main building block is also referred to as the adaptive linear element, usually abbreviated as Adaline.

Training Algorithms for the Single Element

The rules for updating the weight vector **W** until all the input pattern vectors **X** are classified correctly is done by iterative methods. For the single element we have three techniques which are the perceptron convergence procedure, the Widrow–Hoff algorithm, and Mays's rule. They all try adapt to minimize the error for the current training pattern, with minimal disturbance to responses already learned.

The Perceptron Learning Rule

The procedure is to start by initializing the connection weights by small values, or sometimes they are chosen to be zeros (LIPPMANN, 1987). After we present the input vector $\mathbf{X}$ to the network, we update the weight vector $\mathbf{W}$ by:

$$\mathbf{W_{k+1}} = \mathbf{W_k} + \eta(d_k - \mathbf{Y_{k_q}}) \cdot \mathbf{X_k}, \tag{2}$$

where

k	: Time index
$\mathbf{Y_{k_q}}$	: Quantized output
$\mathbf{W_{k+1}}$	: New weight vector
$\mathbf{W_k}$	: Present weight vector
$\mathbf{X_k}$	: Input vector
d_k	: Desired response
η	: Learning rate.

If we denote the quantized error (difference between the desired response and the quantized linear output) by (ε) then we can write *Eq.* (2) in the form

$$\begin{aligned} \mathbf{W_{k+1}} &= \mathbf{W_k} + \eta\varepsilon_k\mathbf{X_k}, \\ \Delta\mathbf{W_k} &= \eta\varepsilon_k\mathbf{X_k}. \end{aligned} \tag{3}$$

The iteration continues until the weight $\mathbf{W}$ converges to some final value. As we see from *Eq.* (3) the modification procedure is simply to add a fraction of the input vector $\mathbf{X}$ to the weight vector $\mathbf{W}$ until the input patterns are classified correctly. This fraction depends upon the quantized error (ε) which is zero when the input vectors are classified correctly. The learning rate (η) controls the adaptation rate, it usually ranges from zero to one. Smaller values of (η) means smaller values of the step size of the weight vector $\mathbf{W}$. The perceptron learning rule stops adapting weights when the input vectors are classified correctly. This can be reached only if the input vectors are linearly separable. If the input vectors are not linearly separable, then the algorithm will not stop adapting the weights, and it will go on forever.

Widrow–Hoff Algorithm

It is the weight modification applied to the Adaline (WIDROW and WINTER, 1988), it is also called the α-LMS algorithm. It is given by *Eq.* (4)

$$\mathbf{W_{k+1}} = \mathbf{W_k} + \alpha\frac{\epsilon_k\mathbf{X_k}}{|\mathbf{X_k}|^2}. \quad \text{where } \epsilon_k = d_k - \mathbf{W_k^T}\mathbf{X_k}. \tag{4}$$

$\mathbf{W_{k+1}}$, $\mathbf{W_k}$, and $\mathbf{X_k}$ are as given by *Eq.* (2), and $\mathbf{W_k^T}$ is the transpose of the weight vector. (ϵ_k) is the linear error which is the difference between the desired response d_k and the output $\mathbf{Y}_\mathbf{k}$. *Eq.* (4) can be written in the form

$$\begin{aligned}\mathbf{W}_{\mathbf{k}+1} &= \mathbf{W}_\mathbf{k} + \alpha \frac{\mathbf{X}_\mathbf{k}(d_k - \mathbf{W}_\mathbf{k}^\mathrm{T}\mathbf{X}_\mathbf{k})}{|\mathbf{X}_\mathbf{k}|^2} = \\ &= \mathbf{W}_\mathbf{k} + \alpha \left(\frac{d_k}{|\mathbf{X}_\mathbf{k}|} - \mathbf{W}_\mathbf{k}^\mathrm{T} \frac{\mathbf{X}_\mathbf{k}}{|\mathbf{X}_\mathbf{k}|} \right) \frac{\mathbf{X}_\mathbf{k}}{|\mathbf{X}_\mathbf{k}|}.\end{aligned} \tag{5}$$

We can define a normalized data set according to

$$\hat{\mathbf{X}}_\mathbf{k} = \frac{\mathbf{X}_\mathbf{k}}{|\mathbf{X}_\mathbf{k}|}, \quad \hat{d}_k = \frac{d_k}{|\mathbf{X}_\mathbf{k}|}. \tag{6}$$

We can write the update equation as

$$\mathbf{W}_{\mathbf{k}+1} = \mathbf{W}_\mathbf{k} + \alpha \hat{\mathbf{X}}_\mathbf{k}(\hat{d}_k - \hat{\mathbf{W}}_\mathbf{k}^\mathrm{T}\hat{\mathbf{X}}_\mathbf{k}). \tag{7}$$

Mays's Algorithm

This algorithm makes use of the increment adaptation rule (WIDROW and LEHR, 1990). Increment adaptation involves the use of a dead zone for the linear output y, this dead zone is from $-\gamma$ to $+\gamma$. If the linear output falls outside the dead zone, adaptation follows a normalized variant of the perceptron rule. If the linear output falls within the dead zone, regardless of the output correct or not, the weights are updated by the normalized variant of the fixed increment perceptron rule. The weight update equation is given by:

$$\mathbf{W}_{\mathbf{k}+1} = \begin{cases} \mathbf{W}_\mathbf{k} + \alpha \epsilon_k \dfrac{\mathbf{X}_\mathbf{k}}{2|\mathbf{X}_\mathbf{k}|^2}, & \text{if } |y_k| \geq \gamma \\ \mathbf{W}_\mathbf{k} + \alpha d_k \dfrac{\mathbf{X}_\mathbf{k}}{|\mathbf{X}_\mathbf{k}|^2}, & \text{otherwise,} \end{cases} \tag{8}$$

where each quantity is as defined in *Eq.* (2). If the training patterns are linearly separable, then increment adaptation will always converge in a finite number of steps, also the use of a dead zone reduces sensitivity to weight changes. If the input patterns are not linearly separable, then the algorithm performs better than the perceptron rule, in this case the weight vector remains in a region associated with low average error.

The other version of the algorithm makes use of the quantized error (ε). If the quantized error is equal to zero and the linear output $\mathbf{Y}$ falls outside the dead zone then there will be no adaptation. In the other case when we have the wrong output or the linear output falls inside the dead zone, then adaptation follows the same ways as in the α-LMS algorithm.

$$\mathbf{W_{k+1}} = \begin{cases} \mathbf{W_k}, & \text{if } \varepsilon_k = 0 \text{ and } |\mathbf{Y_k}| \geq \gamma \\ \mathbf{W_k} + \alpha\epsilon_k \dfrac{\mathbf{X_k}}{|\mathbf{X_k}|^2}, & \text{otherwise.} \end{cases} \tag{9}$$

Performance Functions

In this Section we shall describe an instantaneous performance function for each algorithm given in previous section. These functions will be denoted by (ξ_k), and their instantaneous gradient will be denoted by (∇_k). We shall show that in each algorithm the steepest descent approach is used to move along the error surface. The method of the steepest descent can be described by the equation

$$\mathbf{W_{k+1}} = \mathbf{W_k} + \eta(-\nabla_k), \tag{10}$$

where (η) is the learning rate, and (∇_k) is the value of the gradient of the error surface at a point corresponding to $\mathbf{W} = \mathbf{W_k}$, it is given by:

$$\nabla_k = \frac{\partial \xi_k}{\partial \mathbf{W_k}} = \left[\frac{\partial \xi_k}{\partial w_{1_k}} \frac{\partial \xi_k}{\partial w_{2_k}} \cdots \frac{\partial \xi_k}{\partial w_{i_k}} \cdots \frac{\partial \xi_k}{\partial w_{n_k}} \right]^T . \tag{11}$$

A — For the perceptron learning rule the performance function will be given by

$$\xi_{\text{per}_k} = |\mathbf{Y_k}| - d_k \mathbf{Y_k} \tag{12}$$

And in this case the gradient is given by:

$$\nabla_{\text{per}_k} = \text{sgn}(\mathbf{Y_k})\mathbf{X_k} - d_k \mathbf{X_{k'}} = -\left(d_k - \text{sgn}(\mathbf{Y_k})\right)\mathbf{X_{k'}} = -(\varepsilon_k)\mathbf{X_k}. \tag{13}$$

Substituting *Eq.* (13) in *Eq.* (10) we get the algorithm in (2).

B — For the Widrow–Hoff algorithm the performance function is given by (SHYNK, 1990; SHYNK and SUMIT, 1990)

$$\xi_{\text{wid}_k} = \frac{1}{2|\mathbf{X_k}|^2}(d_k - \mathbf{Y_k})^2 \tag{14}$$

and the gradient is given by *Eq.* (15).

$$\nabla_{\text{wid}_k} = -(d_k - \mathbf{Y_k})\mathbf{X_k}\frac{1}{|\mathbf{X_k}|^2} = -\frac{\epsilon_k \mathbf{X_k}}{|\mathbf{X_k}|^2}. \tag{15}$$

Substituting *Eq.* (15) into *Eq.* (10) we get the algorithm in (4).

C — For Mays's algorithm the performance function is given by:

$$\xi_{\text{mays}\,1_k} = \begin{cases} \dfrac{1}{2|\mathbf{X_k}|^2}(|\mathbf{Y_{q_k}}| - d_k\mathbf{Y_k}), & \text{if } |\mathbf{Y_k}| \geq \gamma \\ -\dfrac{d_k}{|\mathbf{X_k}|^2}\mathbf{Y_k}, & \text{otherwise.} \end{cases} \tag{16}$$

And the gradient is given by

$$\nabla_{\text{mays}\,1_k} = \begin{cases} -\dfrac{\mathbf{X_k}}{2|\mathbf{X_k}|^2}(d_k - \mathbf{Y_{q_k}}), & \text{if } |\mathbf{Y_k}| \geq \gamma \\ -\dfrac{d_k}{|\mathbf{X_k}|^2}\mathbf{X_k}, & \text{otherwise} \end{cases}$$

$$= \begin{cases} -\varepsilon_k\dfrac{\mathbf{X_k}}{2|\mathbf{X_k}|^2}, & \text{if } |\mathbf{Y_k}| \geq \gamma \\ -d_k\dfrac{\mathbf{X_k}}{|\mathbf{X_k}|^2}, & \text{otherwise.} \end{cases} \tag{17}$$

Substituting *Eq.* (17) into (10) we get the algorithm in (8).

— For the other form of the Mays's algorithm we get the following performance function

$$\xi_{\text{mays}\,2_k} = \begin{cases} c, & \text{if } \varepsilon = 0 \text{ and } |y_k| \geq \gamma \\ \dfrac{1}{2|\mathbf{X_k}|^2}(d_k - \mathbf{Y_k})^2, & \text{otherwise.} \end{cases} \tag{18}$$

The gradient is given by:

$$\nabla_{\text{mays}\,2_k} = \begin{cases} 0, & \text{if } \varepsilon = 0 \text{ and } |\mathbf{Y_k}| \geq \gamma \\ -\dfrac{\mathbf{X_k}}{|\mathbf{X_k}|^2}(d_k - \mathbf{Y_k}), & \text{otherwise.} \end{cases} \tag{19}$$

Substituting *Eq.* (19) into *Eq.* (10) we get the algorithm in *Eq.* (9).

New Algorithm for the Single Element

PROPOSITION. *The convergence will be faster if we use the following performance function*

$$\xi_k = \frac{1}{\beta}(1 - \mathbf{Y_k} d_k)^{\beta} \quad \textit{if}\, \mathbf{Y_k} d_k < 0$$
$$\textit{where} d_k = \pm 1, \quad -1 \leq \mathbf{Y_k} \leq 1. \tag{20}$$

The value of the desired response (d_k) is restricted to either $+1$ or -1 because the output is quantized to two possible states. Now based upon the performance function given by *Eq.* (20), we shall derive the update equations for a new learning rule for the single element. The instantaneous gradient of the performance function given in *Eq.* (20) is given by:

$$\nabla_k = \begin{cases} -\mathbf{X_k} d_k (1 - d_k \mathbf{Y_k})^{\beta-1}, & \text{if } d_k \mathbf{Y_k} < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

The above equation can be approximated to take the form

$$\nabla_k \approx \begin{cases} -\mathbf{X_k} d_k (1 - (\beta - 1) d_k \mathbf{Y_k}), & \text{if } d_k \mathbf{Y_k} < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

Using *Eq.* (10), we can write this new learning algorithm for the single perceptron

$$\mathbf{W_{k+1}} = \begin{cases} \mathbf{W_k} + \mu \mathbf{X_k}(d_k - (\beta - 1)\mathbf{Y_k}), & \text{if } \mathbf{Y_k} d_k < 0 \\ \mathbf{W_k}, & \text{otherwise.} \end{cases} \tag{23}$$

If we use a normalized training set as defined by *Eq.* (6) then the update equation will be

$$\mathbf{W_{k+1}} = \begin{cases} \mathbf{W_k} + \mu \frac{\mathbf{X_k}}{|\mathbf{X_k}|}\left(\frac{d_k}{|\mathbf{X_k}|} - (\beta - 1)\mathbf{W_k^T}\frac{\mathbf{X_k}}{|\mathbf{X_k}|}\right), & \text{if } \mathbf{Y_k} d_k < 0 \\ \mathbf{W_k}, & \text{otherwise.} \end{cases}$$

$$(24)$$

$$= \begin{cases} \mathbf{W_k} + \mu \tilde{\mathbf{X}}_\mathbf{k}(\tilde{d}_k) - (\beta - 1)\mathbf{W_k^T}\tilde{\mathbf{X}}_\mathbf{k}), & \text{if } \mathbf{Y_k} d_k < 0 \\ \mathbf{W_k}, & \text{otherwise} \end{cases}$$

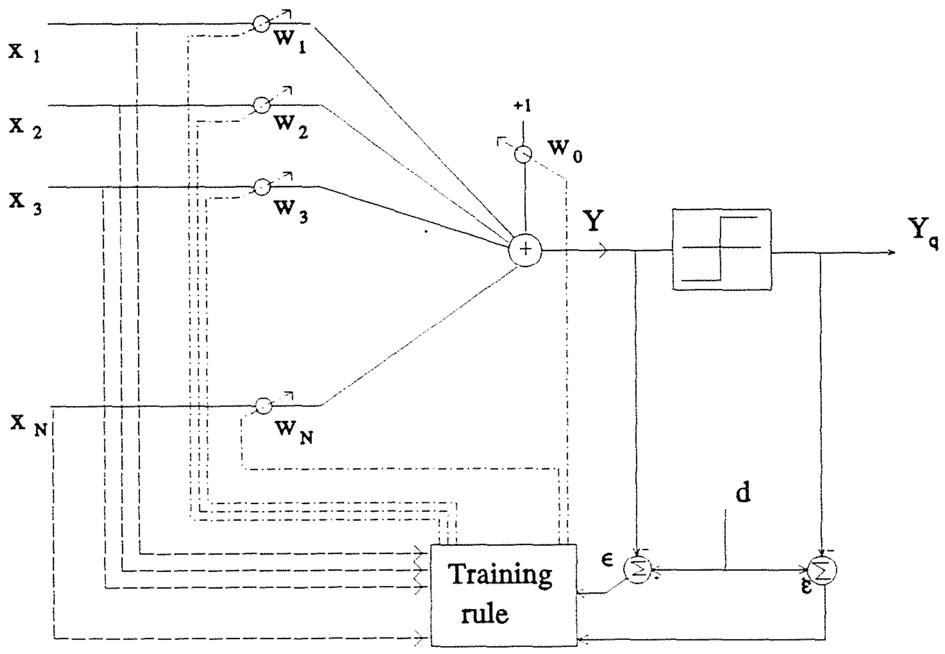

Fig. 2. Weight adaptation for the single element

Optimal Weight Vector for the New Rule

The performance function given by *Eq.* (20) can be expressed into Taylor's series as

$$\xi = \frac{1}{\beta}\left(1 - \beta \mathbf{Y}_{\mathbf{k}} d_k + \frac{\beta(\beta-1)}{2!}(\mathbf{Y}_{\mathbf{k}} d_k)^2 - \frac{\beta(\beta-1)(\beta-2)}{3!}(\mathbf{Y}_{\mathbf{k}} d_k)^3 + \dots\right). \tag{25}$$

We can approximate *Eq.* (25) to become

$$\xi_k \approx \xi_{a_k} = \frac{1}{\beta}\left(1 - \beta \mathbf{Y}_{\mathbf{k}} d_k + \frac{\beta(\beta-1)}{2}(\mathbf{Y}_{\mathbf{k}} d_k)^2\right), \tag{26}$$

where

$d_k = \mp 1$
$\mathbf{Y}_{\mathbf{k}} = \mathbf{W}_{\mathbf{k}}^{\mathbf{T}}.\mathbf{X}_{\mathbf{k}}$
$\mathbf{X}_{\mathbf{k}}$: the input vector
$\mathbf{W}_{\mathbf{k}}^{\mathbf{T}}$: the transpose of the weight vector

We can write the performance function in the form

$$\begin{aligned}\xi_{a_k} &= \frac{1}{\beta}\left(1-\beta(d_k\mathbf{W}_\mathbf{k}^\mathrm{T}\mathbf{X}_\mathbf{k})+\frac{\beta(\beta-1)}{2}(\mathbf{W}_\mathbf{k}^\mathrm{T}\mathbf{X}_\mathbf{k}\mathbf{W}_\mathbf{k}^\mathrm{T}\mathbf{X}_\mathbf{k})\right)\\ &= \frac{1}{\beta}\left(1-\beta(d_k\mathbf{X}_\mathbf{k}^\mathrm{T}\mathbf{W}_\mathbf{k})+\frac{\beta(\beta-1)}{2}(\mathbf{W}_\mathbf{k}^\mathrm{T}.\mathbf{X}_\mathbf{k}\mathbf{X}_\mathbf{k}^\mathrm{T}.\mathbf{W}_\mathbf{k})\right).\end{aligned} \tag{27}$$

We take the expected value of (27) assuming

— The input vectors are statistically stationary independent vectors.
— The weight vector is independent of the input vector $\mathbf{X}_\mathbf{k}$.

$$\begin{aligned}\mathbf{E}(\xi_{a_k}) &= \mathbf{E}\left(\frac{1}{\beta}\left(1-\beta(d_k\mathbf{X}_\mathbf{k}^\mathrm{T}\mathbf{W}_\mathbf{k})+\frac{\beta(\beta-1)}{2}(\mathbf{W}_\mathbf{k}^\mathrm{T}.\mathbf{X}_\mathbf{k}\mathbf{X}_\mathbf{k}^\mathrm{T}.\mathbf{W}_\mathbf{k})\right)\right)\\ &= \frac{1}{\beta}\left(1+\frac{\beta(\beta-1)}{2}\mathbf{W}^\mathrm{T}\mathbf{E}(\mathbf{X}_\mathbf{k}.\mathbf{X}_\mathbf{k}^\mathrm{T})\mathbf{W}-\beta\mathbf{E}(d_k\mathbf{X}_\mathbf{k}^\mathrm{T})\mathbf{W}\right)\end{aligned} \tag{28}$$

$\mathbf{E}(\mathbf{X}_\mathbf{k}.\mathbf{X}_\mathbf{k}^\mathrm{T})$:	$\mathbf{R}$ Input correlation matrix
$\mathbf{E}(d_k\mathbf{X}_\mathbf{k}^\mathrm{T})$:	$\mathbf{P}^\mathrm{T}$is the crosscorrelation between the desired response d_k and the input vector.

Now we can write

$$\begin{aligned}\bar{\xi}_a &= \frac{1}{\beta}+\frac{(\beta-1)}{2}\mathbf{W}^\mathrm{T}\mathbf{R}\mathbf{W}-\mathbf{P}^\mathrm{T}\mathbf{W}\\ &= \frac{1}{\beta}+\frac{(\beta-1)}{2}\mathbf{W}^\mathrm{T}\mathbf{R}\mathbf{W}-\mathbf{W}^\mathrm{T}\mathbf{P}.\end{aligned} \tag{29}$$

The gradient is given by:

$$\nabla = (\beta-1)\mathbf{R}\mathbf{W}-\mathbf{P}. \tag{30}$$

We can obtain the optimal weight vector $\mathbf{W}^*$ by letting the gradient equal to zero and then solve for $\mathbf{W}$,

$$(\beta-1)\mathbf{R}\mathbf{W}^*-\mathbf{P}=0$$

or

$$\mathbf{W}^* = \frac{1}{\beta-1}\mathbf{R}^{-1}\mathbf{P}. \tag{31}$$

We can find the minimum value of the performance function by:

$$\begin{aligned}\xi_{a_{\min}} &= \frac{1}{\beta} + \frac{(\beta-1)}{2}\mathbf{W}^{*\mathrm{T}}\mathbf{R}\mathbf{W}^{*} - \mathbf{P}^{\mathrm{T}}\mathbf{W}^{*} \\ &= \frac{1}{\beta} + \frac{(\beta-1)}{2}\left(\frac{1}{\beta-1}\mathbf{R}^{-1}\mathbf{P}\right)^{\mathrm{T}} \\ \mathbf{R}&\left(\frac{1}{\beta-1}\mathbf{R}^{-1}\mathbf{P}\right) - \mathbf{P}^{\mathrm{T}}\left(\frac{1}{\beta-1}\mathbf{R}^{-1}\mathbf{P}\right) \\ &= \frac{1}{\beta} + \frac{1}{2(\beta-1)}\mathbf{P}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{R}\mathbf{R}^{-1}\mathbf{P} - \frac{1}{\beta-1}\mathbf{P}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{P} \\ &= \frac{1}{\beta} - \frac{1}{2(\beta-1)}\mathbf{P}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{P}.\end{aligned} \tag{32}$$

The above equation can be written in form

$$\xi_{a_{\min}} = \frac{1}{\beta} - \frac{1}{2}\mathbf{P}^{\mathrm{T}}\mathbf{W}^{*}. \tag{33}$$

The Convergence Condition of the Weight Vector

The update equation is given by:

$$\begin{aligned}\mathbf{W}_{\mathbf{k}+1} &= \mathbf{W}_{\mathbf{k}} + \mu\mathbf{X}_{\mathbf{k}}(d_k - (\beta-1)\mathbf{Y}_{\mathbf{k}}) \\ &= \mathbf{W}_{\mathbf{k}} + \mu(d_k\mathbf{X}_{\mathbf{k}} - (\beta-1)\mathbf{X}_{\mathbf{k}}\mathbf{Y}_{\mathbf{k}}) \\ &= \mathbf{W}_{\mathbf{k}} + \mu(d_k\mathbf{X}_{\mathbf{k}} - (\beta-1)\mathbf{X}_{\mathbf{k}}\mathbf{X}_{\mathbf{k}}^{\mathrm{T}}\mathbf{W}_{\mathbf{k}}).\end{aligned} \tag{34}$$

Taking the expected value of *Eq.* (34) we get

$$\begin{aligned}\mathbf{E}[\mathbf{W}_{\mathbf{k}+1}] &= \mathbf{E}[\mathbf{W}_{\mathbf{k}}] + \mu(\mathbf{E}[d_k\mathbf{X}_{\mathbf{k}}] - (\beta-1)\mathbf{E}[\mathbf{X}_{\mathbf{k}}\mathbf{X}_{\mathbf{k}}^{\mathrm{T}}\mathbf{W}_{\mathbf{k}}]) \\ &= \mathbf{E}[\mathbf{W}_{\mathbf{k}}] + \mu(\mathbf{P} - (\beta-1)\mathbf{R}\mathbf{E}[\mathbf{W}_{\mathbf{k}}]).\end{aligned} \tag{35}$$

Eq. (35) can be written in terms of the optimal weight vector $\mathbf{W}^{*}$

$$\begin{aligned}\mathbf{E}[\mathbf{W}_{\mathbf{k}+1}] &= \mathbf{E}[\mathbf{W}_{\mathbf{k}}] + \mu\left((\beta-1)\mathbf{R}\mathbf{W}^{*} - (\beta-1)\mathbf{R}\mathbf{E}[\mathbf{W}_{\mathbf{k}}]\right) \\ &= (\mathbf{I} - \mu(\beta-1)\mathbf{R})\mathbf{E}[\mathbf{W}_{\mathbf{k}}] + \mu(\beta-1)\mathbf{R}\mathbf{W}^{*}.\end{aligned} \tag{36}$$

The correlation matrix **R** could be diagonalized as (WIDROW and STEARNS, 1985)

$$\begin{gathered}\Lambda = \mathbf{Q}^{\mathrm{T}}\mathbf{R}\mathbf{Q}, \\ \mathbf{Q} = [q_1, q_2, \ldots, q_n], \\ q_i : \text{is the } i^{th} \text{ eigenvector of } \mathbf{R}.\end{gathered} \tag{37}$$

We translate from $\mathbf{W}$ to a new coordinate $\mathbf{V}$ using

$$\mathbf{V} = \mathbf{W} - \mathbf{W}^*. \tag{38}$$

Then *Eq.* (36) becomes

$$\begin{aligned} \mathbf{E}[\mathbf{W}_{\mathbf{k+1}}] - \mathbf{W}^* &= \mathbf{E}[\mathbf{W}_{\mathbf{k}}] - \mathbf{W}^* + \mu(\beta - 1)\mathbf{R}(\mathbf{W}^* - \mathbf{E}[\mathbf{W}_{\mathbf{k}}]), \\ \mathbf{E}[\mathbf{V}_{\mathbf{k+1}}] &= \mathbf{E}[\mathbf{V}_{\mathbf{k}}] - \mu(\beta - 1)\mathbf{R}\mathbf{E}[\mathbf{V}_{\mathbf{k}}] \\ &= (\mathbf{I} - \mu(\beta - 1)\mathbf{R})\mathbf{E}[\mathbf{V}_{\mathbf{k}}]. \end{aligned} \tag{39}$$

We rotate to the principle axes using

$$\mathbf{V} = \mathbf{Q}\tilde{\mathbf{V}}. \tag{40}$$

Then *Eq.* (39) becomes

$$\mathbf{Q}\mathbf{E}[\tilde{\mathbf{V}}_{\mathbf{k+1}}] = (\mathbf{I} - \mu(\beta - 1)\mathbf{R})\mathbf{Q}\mathbf{E}[\tilde{\mathbf{V}}_{\mathbf{k}}]. \tag{41}$$

Multiplying both sides of *Eq.* (41) by $\mathbf{Q}^{-1}$ we obtain

$$\mathbf{Q}^{-1}\mathbf{Q}\mathbf{E}[\tilde{\mathbf{V}}_{\mathbf{k+1}}] = \mathbf{Q}^{-1}(\mathbf{I} - \mu(\beta - 1)\mathbf{R})\mathbf{Q}\mathbf{E}[\tilde{\mathbf{V}}_{\mathbf{k}}],$$

or

$$\mathbf{E}[\tilde{\mathbf{V}}_{\mathbf{k+1}}] = (\mathbf{Q}^{-1}\mathbf{I}\mathbf{Q} - \mu(\beta - 1)\mathbf{Q}^{-1}\mathbf{R}\mathbf{Q})\mathbf{E}[\tilde{\mathbf{V}}_{\mathbf{k}}]. \tag{42}$$

Since we have $\mathbf{Q}^{-1} = \mathbf{Q}^{\mathrm{T}}$, *Eq.* (42) could be written in the form

$$\mathbf{E}[\tilde{\mathbf{V}}_{\mathbf{k+1}}] = (\mathbf{I} - \mu(\beta - 1)\Lambda)\mathbf{E}[\tilde{\mathbf{V}}_{\mathbf{k}}]. \tag{43}$$

The solution of *Eq.* (43) is

$$\mathbf{E}[\tilde{\mathbf{V}}_{\mathbf{k}}] = (\mathbf{I} - \mu(\beta - 1)\Lambda)^k\tilde{\mathbf{V}}_0, \tag{44}$$

where $\tilde{\mathbf{V}}_0$ is the initial Weight vector in the principle axis.

Convergence is guaranteed if

$$\lim_{k\to\infty}(\mathbf{I} - \mu(\beta - 1)\Lambda)^k = 0. \tag{45}$$

Since Λ is a diagonal matrix, we can write *Eq.* (45) as a set of n equations

$$\lim_{k\to\infty}(1 - \mu(\beta - 1)\lambda_i)^k = 0, \quad 1 \le i \le n. \tag{46}$$

The condition to converge can be written as

$$\begin{gathered} |1-\mu(\beta-1)\lambda_i| < 1, \\ 0 < \mu(\beta-1)\lambda_i < 2, \\ 0 < \frac{(\beta-1)\mu}{2} < \frac{1}{\lambda_i}, \qquad 1 \leq i \leq n. \end{gathered} \tag{47}$$

Since we have

$$\lambda_i \leq \lambda_{\max} \leq \mathrm{tr}[\mathbf{R}],$$

where ($\lambda_{\max}$) is the largest eigenvalue of $\mathbf{R}$, and $\mathrm{tr}[\mathbf{R}]$ is the trace of $\mathbf{R}$, we can write the convergence condition as

$$\begin{gathered} 0 < \frac{(\beta-1)\mu}{2} < \frac{1}{\lambda_{\max}}, \\ 0 < \frac{\mu(\beta-1)}{2} < \frac{1}{\mathrm{tr}[\mathbf{R}]} \end{gathered}. \tag{48}$$

Computer Simulations

Table 1
Number of learning cycles needed to converge with different size of input patterns. Learning rate = 0.9.

The size of the input vectors	Iterations required to converge			
	The perceptron rule	The Widrow — Hoff rule	Mays's rule	The new rule
2	8	5	5	4
5	49	22	22	7
10	60	48	49	15
20	248	264	264	26
40	226	362	362	19
60	642	417	417	83
80	812	672	798	128
100	1390	1172	1152	131
120	1536	1295	1295	167

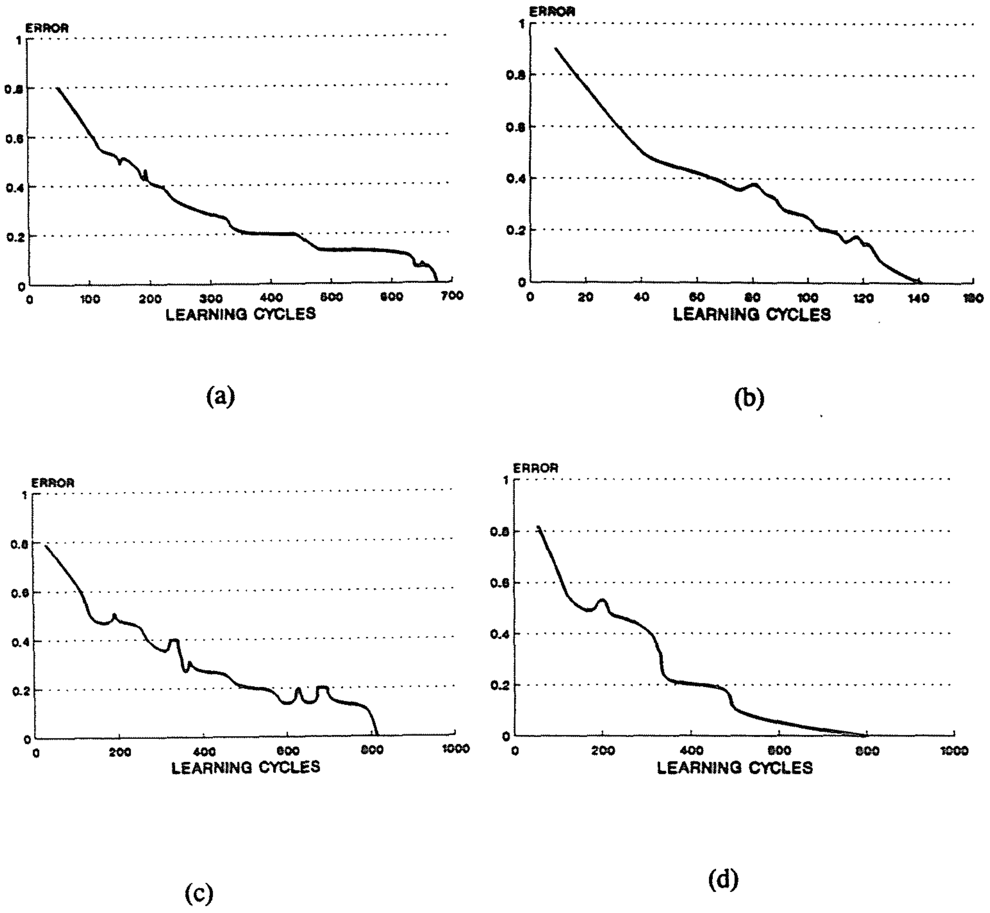

Fig. 3. Learning curves for the four algorithms, (a) The Widrow–Hoff algorithm, (b) The new algorithm, (c) The perceptron training rule, (d) Mays's algorithm

In order to have an empirical validation, extensive performance comparisons between the perceptron learning rule, the α-LMS, Mays's rule and the new learning rule by the author have been conducted. In every case the new learning rule converges faster than the remaining rules. *Table 1* shows some examples of simulation results for different size of the input vectors. Normalized training set was used to train the networks. There were ten trials, and each entry in the *Table 1* is the average of these trials. The learning rate was the same for all rules, it was equal to (0.9). The same initial weights were used for all algorithms. The results in *Table 1* show that the rule converges faster than any of the other rules. *Fig. 3* shows the learning curves for the four rules. No attempt was made to optimize the new rule, only $\beta = 1.7$ value was conducted.

References

WIDROW, B. — LEHR, M. (1990): 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation, *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1415–1441.

AMARI, S. C. (1990): Mathematical Foundation of Neurocomputing, *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1443–1463.

WIDROW, B. — WINTER, R. G. (1988): Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition, *IEEE Comp. Mar.* pp. 25–39.

LIPPMANN, R. P.(1978): An Introduction to the Computing with Neural Nets, *IEEE Transactions on Acoustics, Speech, and Signal Processing Magazine.*

SHYNK, J. (1990): Performance Surfaces of a Single-layer Perceptron, *IEEE Transactions on Neural Networks*, Vol. 1, No. 3. pp. 268–274.

SHYNK, J. — SUMIT, R. (1990): Convergence Properties and Stationary Points of a Perceptron Learning Algorithm. *Proceedings of the IEEE*, Vol. 78, No. 10, October, pp. 1599–1604.

WIDROW, B. — STEARNS, S. D. (1985): Adaptive Signal Processing. Englewood Cliffs, NJ: Prentice-hall, 1985.