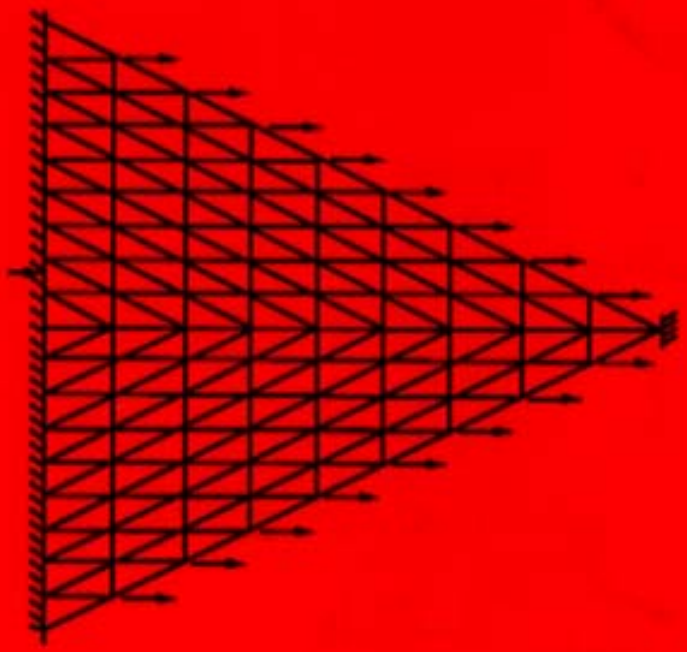
THEORETICAL ASPECTS OF INDUSTRIAL DESIGN



Edited by David A. Field
Vadim Komkov

siam

CHAPTER 10

On the Adaptation Algorithms for Feedforward Neural Networks

Stanislaw H Żak*
Hebertt J. Sira-Ramirez**

Abstract. In this paper new training algorithms are proposed for a class of feedforward neural networks with differentiable and nondifferentiable nonlinearities. The class of neural networks considered in this paper can be viewed as generalized single and multi-layered perceptrons. The learning parameters in the proposed algorithms are adjusted to force the error between the actual and desired outputs to satisfy a stable difference equation.

1. INTRODUCTION. An artificial neural network is a large-scale nonlinear circuit of interconnected simple circuits called nodes or neurons. These networks resemble patterns of the biological neural networks hence the term artificial neural networks. One of the reasons for studying such circuits came from attempts "to understand how known biophysical properties and (the) architectural organization of neural systems can provide the immense computational power characteristic of the brains of higher animals" (Tank and Hopfield, 1986, p. 533). Another reason of interest in neural networks is the low execution speed of conventional computers which perform a program of instructions serially or sequentially. In contrast, neural networks operate in parallel. The ability to be interconnected in a regular fashion results in higher computation rates. Furthermore, regular interconnections of the same basic cells lead to easier design and testing of a chip. Potential applications of neural networks are in such areas as speech and image recognition, linear and nonlinear optimization, automatic control, and in highly parallel computers to mention but a few.

In this paper our interest is in the class of feedforward neural networks which can be viewed as generalized perceptrons.

The development of the perceptron can be traced back to the early days of pattern recognition (See Grossberg (1988), Lippmann (1987), Pao (1989), Widrow and Winter (1988), Widrow and Lehr (1990), and Haykin (1984) for more details). Its application as an adaptive system to the control of many degrees of freedom robotic manipulators was proposed by Albus in 1975. More recently, Widrow and Winter (1988), and Widrow and Lehr (1990) discussed numerous applications of perceptrons for adaptive filtering (See also

*School of Electrical Engineering, Purdue University, West Lafayette, IN 47907, USA.

**Departamento Sistemas de Control, Escuela de Ingenieria de Sistemas, Universidad de Los Andes, Merida-VENEZUELA.

Åström and Wittenmark (1989), and Haykin (1984)), adaptive pattern recognition, and adaptive signal processing.

The central role in advancing the practicality of perceptrons, and neural networks in general, are played by adaptation algorithms. In the case of the single perceptron one of the most well known algorithms that minimizes the mean square error between the desired output and the actual output is due to Widrow and Hoff. For the layered perceptron the central role is played by the back propagation algorithm (see Widrow and Winter (1988) for historical remarks, Rumelhart, McClelland et al. (1986) and Pao (1989), for the derivation of this algorithm and Rumelhart, McClelland et al (1986) and Rangwala and Dornfeld (1989) for its applications). For an interesting discussion of the back propagation in the context of the control of dynamic systems see Narendra and Parthasarathy (1990). One of the drawbacks of the back propagation algorithm is the requirement that the nonlinear activation functions be differentiable.

In this paper we propose a new class of adaptation, or training, algorithms for generalized single and multi-layer perceptrons. Our proposed algorithms, unlike the back-propagation training algorithm, do not require differentiability along the network's signal paths. On the contrary, we include activation functions which are not only nondifferentiable but also discontinuous like saturation functions and/or hard limiters. Another feature of the training algorithms proposed in this paper is that the learning parameters are adjusted in such a way so that the error between the actual and desired outputs satisfies a stable difference equation. This is also characteristic of the celebrated Widrow-Hoff algorithm for single-layer perceptrons.

The paper is organized as follows. In the next Section we briefly review the Widrow-Hoff adaptation rule. This rule constitutes a nice starting point in our development. In Section 3 we propose a new adaptation algorithm for the single perceptron. The proposed algorithm is a generalization of the Widrow-Hoff adaptation rule. In Section 4 we present a new training algorithm for generalized two-layered perceptrons. In Section 5 we are concerned with a new adaptation algorithm for generalized three-layered perceptrons. Algorithms presented in Sections 4 and 5 are generalizations of algorithms of Sira-Ramirez and Żak (1991). The conclusions of the paper are found in Section 6.

2. BRIEF REVIEW OF THE WIDROW-HOFF ADAPTATION RULE.

The single perceptron as an adaptive threshold element is shown in Fig. 1.

One can use the Widrow-Hoff delta rule (see Widrow and Winter (1988) or Widrow and Lehr (1990) for its discussion) to adjust the weights ω_i $(i = 1,2,...,n)$. The algorithm

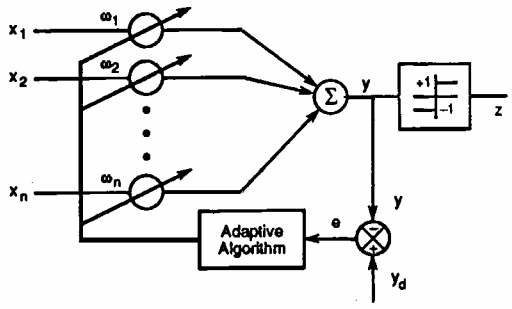


Fig. 1. Single perceptron

can be written as follows:

$$W(k+1) = W(k) + \frac{\alpha e(k) X}{X^T X}, \quad X \neq 0 \tag{2.1}$$

where

k = the time index or the adaption cycle number,
$W(k) = [\omega_1(k),...,\omega_n(k)]^T$ is the value, at time k, of the weight vector,
$X = [x_1,...,x_n]^T$ is the present input pattern,
$e(k) = y_d - y(k)$ is the present error,
α = the reduction factor whose practical range is (0.1, 1.0).

After some manipulations, one can conclude that the error is reduced by a factor of α at each new learning iteration as the weights are changed while holding the input pattern X fixed. More specifically, the error obeys the following difference equation

$$e(k+1) = (1-\alpha)\, e(k)\,. \tag{2.2}$$

As one can see from the above equation, the choice of α controls the speed of convergence towards zero of the learning error signal $e(\cdot)$.

In this paper as in Widrow and Winter (1988, p. 34) and Widrow and Lehr (1990, p. 1424) we analyze only the case when the same input pattern X is presented in successive iterations and no other input pattern is being presented, that is, $X(k) = X(k+1) = ... = X$. We thus reduce the error while holding the input pattern fixed. The new input pattern would start the next adaptation cycle. We do not analyze how much the responses to previous input patterns are disturbed when adapting to respond to a new training pattern.

3. NEW ADAPTATION RULES FOR A SINGLE PERCEPTRON. We present the new adaptation algorithm in the following theorem.

Theorem 1. Let $\Theta : \mathbb{R}^n \to \mathbb{R}$ be an operator, then if the weights ω_i of the single perceptron, shown in Fig. 1, are adapted according to the rule

$$W(k+1) = W(k) + \frac{\alpha\, e(k)\, \Theta[X]}{X^T \Theta[X]} \quad (X^T \Theta[X] \neq 0)\,, \tag{3.1}$$

with $0 < \alpha < 2$, then the error $e(k)$ tends asymptotically to zero with the rate of convergence $(1-\alpha)$.

Proof. Note that

$$\begin{aligned} e(k+1) - e(k) &= y_d - y(k+1) - [y_d - y(k)] \\ &= -\sum_{i=1}^{n} [\omega_i(k+1) - \omega_i(k)]\, x_i \\ &= -X^T [W(k+1) - W(k)]\,. \end{aligned}$$

We can now use the proposed update rule to obtain:

$$e(k+1) - e(k) = -X^T \frac{\alpha\, e(k)\, \Theta[X]}{X^T\, \Theta[X]} = -\alpha\, e(k) \quad \text{if } X^T \Theta[X] \neq 0\,.$$

Hence

$$e(k+1) = (1-\alpha)\, e(k)\,.$$

Thus, if $0 < \alpha < 2$ then

$$\lim_{k \to \infty} e(k) = 0$$

□

Note that in the new adaptation algorithm, as well as in the Widrow-Hoff algorithm, the error is reduced by a factor of α.

Observe that if Θ is the identity operator then the new algorithm (3.1) is the same as the Widrow-Hoff adaptation rule. If on the other hand,

$$\Theta[X] = \begin{bmatrix} \text{sgn } x_1 \\ \text{sgn } x_2 \\ \vdots \\ \text{sgn } x_n \end{bmatrix} = \text{SGN X},$$

then (3.2) is the same as the algorithm proposed by Sira-Ramirez and Żak (1991).

Example 1. (System Identification) This example deals with the problem of system modeling as shown in Fig. 2 (See Widrow and Winter (1980), p. 26, for more details). Our problem is to identify the behavior of an unknown dynamical system. In the simulations, the "unknown" dynamical system is described by the following equations

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -2x_1 - 2x_2 + u \\ y &= x_1 . \end{aligned} \tag{3.2}$$

We take $\Theta[X] = \text{SGN X}$.

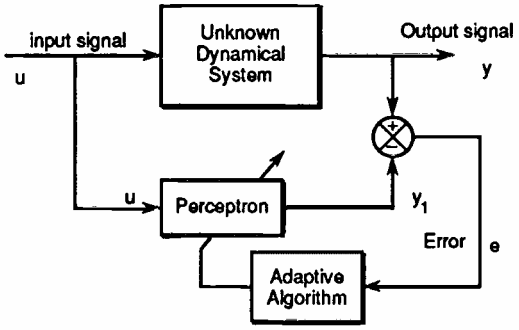


Fig. 2. Scheme for identification of the behavior of an unknown dynamic system.

The input vector to the perceptron is provided by means of the delay elements arranged in a transversal filter scheme as depicted in Fig. 3.

The input signal to the system u(t), the output y(t) of the unknown system, the output $y_1(t)$ of the perceptron and the tracking error e(t) are shown in Fig. 4.

The adaptation time history of the perceptron weights is shown in Figure 5.

We shall next present new adaptation algorithms for the multi-layer perceptrons. The proposed algorithms are an extension of the training algorithms presented in Sira-Ramirez and Żak (1991).

4. ADAPTATION ALGORITHMS FOR TWO-LAYER GENERALIZED PERCEPTRONS. In the next two Sections we will be concerned with generalized multi-layer perceptrons which are feedforward networks with one or more layers of nodes between the input and output nodes. It is generally acknowledged that the applicability of multi-layer neural networks is highly dependent on the efficiency of the training algorithms. One of the best known training algorithms for multi-layer perceptrons is the back propagation algorithm. A disadvantage of the back propagation algorithm is the inherent requirement of continuous differentiability of the nonlinearities. Our proposed algorithms, unlike the back propagation training algorithm, do not require

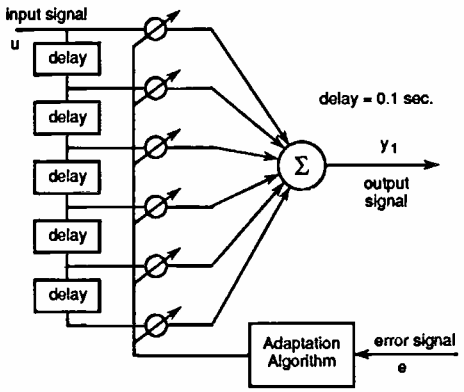


Fig. 3. Structure of the perceptron used to identify the behavior of the unknown system in Example 1.

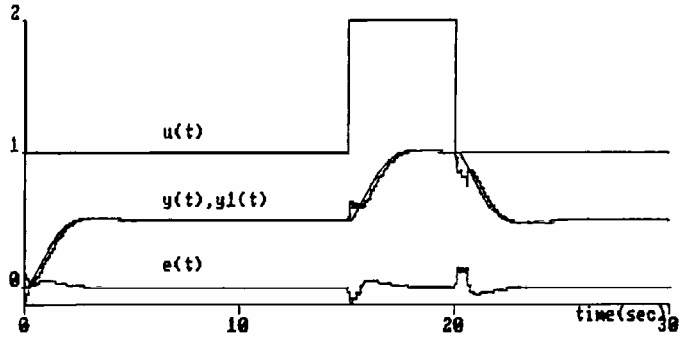


Fig. 4. Time history of u(t), y(t), $y_1(t)$, and e(t).

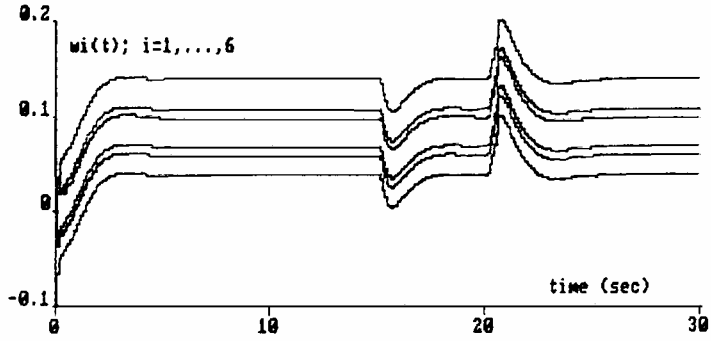


Fig. 5. Adaptation of the perceptron weights.

differentiability along the network's signal paths. On the contrary we include activation functions which are of the hard limiter type or saturation type. In the back propagation algorithm one presents an input to the network and calculates the output corresponding to the current set of learning parameters. One then compares the actual network output with the desired output and calculates the Euclidean distance between the actual and desired outputs, called the error function. The learning procedure aims at minimizing the error function by suitable adjustments of the learning parameters. One calculates the gradient of the error function with respect to the learning parameters starting at the

output nodes and working back towards the input nodes through the hidden layers. Once the gradient is calculated, the learning parameters are adjusted using the gradient descent method. A substantial departure from the back propagation procedure is proposed in our new training algorithms. Here, the learning parameters are adjusted to force the error between the actual and desired outputs to satisfy a stable difference error equation, rather than to minimize an error function. This approach allows one to better control the stability and speed of convergence by appropriate choice of parameters of the error difference equation.

We shall start our analysis by considering a two-layer adaptive neural network as depicted in Fig. 6.

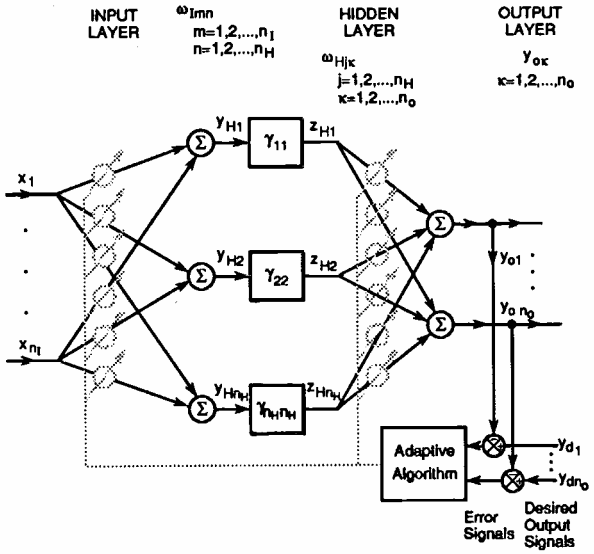


Fig. 6. A two-layer adaptive neural network with a diagonal nonlinear operator Γ in the hidden layer.

Notation and definitions. Before presenting new training algorithms we shall introduce some notation and definitions.

Let $\Gamma : \mathbb{R}^\ell \to \mathbb{R}^\ell$ denotes a nonlinear operator with the following property

$$\boxed{\Gamma[-X] = -\Gamma[X]} \tag{4.1}$$

The operator Γ can have, in particular, the following forms

$$\Gamma[X] = \begin{bmatrix} \operatorname{sgn} x_1 \\ \vdots \\ \operatorname{sgn} x_\ell \end{bmatrix},$$

where

$$\operatorname{sgn} x_i = \begin{cases} 1 & \text{if } x_i > 0 \\ -1 & \text{if } x_i < 0 . \end{cases}$$

Another possible form of the operator Γ is

$$\Gamma[X] = \begin{bmatrix} \text{sat } x_1 \\ \vdots \\ \text{sat } x_\ell \end{bmatrix},$$

where

$$\text{sat } x_i = \begin{cases} +1 & \text{if } x_i > 1 \\ x_i & \text{if } x_i \in [-1,1] \\ -1 & \text{if } x_i < 1 \end{cases},$$

or

$$\Gamma[X] = \begin{bmatrix} \text{sig } x_1 \\ \vdots \\ \text{sig } x_\ell \end{bmatrix}$$

where

$$\text{sig } x_i = \frac{1 - e^{-x_i}}{1 + e^{-x_i}}.$$

Observe that Γ does not have to be a "diagonal" operator. However, we require that Γ is an odd operator, that is, it satisfies (4.1). Notice that Γ can also be an identity operator.

The κ-th output of the network in Fig. 6 is obtained as the weighted sum of the hidden neuron outputs, that is

$$y_{o\kappa} = \sum_{j=1}^{n_H} \omega_{Hj\kappa}(k) z_{Hj}(k) \qquad \kappa = 1,2,...,n_o$$

where n_H is the number of hidden neurons, n_o is the number of outputs, and z_{Hj} are the hidden neurons outputs. The quantities $\omega_{Hj\kappa}(k)$ represent the interconnection weights existing between the hidden neuron layer and the outputs. Thus, in vector notation

$$y_{o\kappa}(k) = [W_H^\kappa(k)]^T Z_H(k) \qquad \kappa = 1,2,...,n_o$$

where

$$[W_H^\kappa(k)] = \begin{bmatrix} \omega_{H1\kappa}(k) \\ \omega_{H2\kappa}(k) \\ \vdots \\ \omega_{Hn_H\kappa}(k) \end{bmatrix},$$

and

$$Z_H(k) = \begin{bmatrix} z_{H1}(k) \\ z_{H2}(k) \\ \vdots \\ z_{Hn_H}(k) \end{bmatrix} = \begin{bmatrix} \gamma_{11}[y_{H1}(k)] \\ \gamma_{22}[y_{H2}(k)] \\ \vdots \\ \gamma_{n_Hn_H}[y_{Hn_H}(k)] \end{bmatrix} = \Gamma[Y_H(k)].$$

If we denote by W_H the matrix of column vectors W_H^κ, then the vector $Y_o(k)$ of the output components $y_{o\kappa}(k)$ can be represented as

$$\boxed{Y_o(k) = [W_H(k)]^T Z_H(k)}$$

Furthermore

$$y_{Hn}(k) = \sum_{m=1}^{n_I} \omega_{Imn}(k)x_m$$
$$= [W_I^n(k)]^T X \ , \quad n = 1,2,...,n_H$$

where

$$[W_I^n(k)] = \begin{bmatrix} \omega_{I1n}(k) \\ \omega_{I2n}(k) \\ \vdots \\ \omega_{In_In}(k) \end{bmatrix},$$

and

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_I} \end{bmatrix}.$$

Denoting by W_I the matrix of column vectors W_I^n we obtain the vector $Y_H(k)$ of the components $y_{Hn}(k)$:

$$\boxed{Y_H(k) = [W_I(k)]^T X}$$

A schematic representation of the two-layer adaptive generalized perceptron is depicted in Fig. 7. Observe that the nonlinear operator Γ does not have to be a diagonal one. However, it must satisfy (4.1).

Let Y_d denotes the desired output vector with components. $y_{d\kappa}$, $\kappa = 1,2,...,n_o$. The error vector E(k) at time k is

$$E(k) = \begin{bmatrix} e_1(k) \\ \vdots \\ e_{n_o}(k) \end{bmatrix} = Y_d - Y_o(k)$$

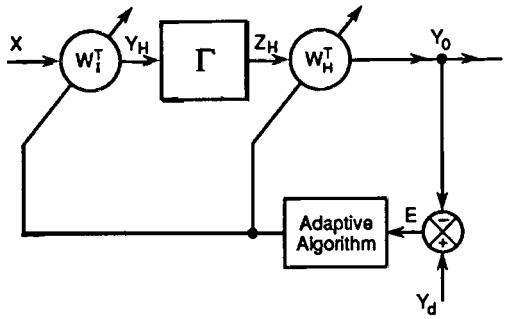


Fig. 7. Schematic representation of the two-layer generalized perceptron.

The weights updates are represented by the following equations:

$$\omega_{Hj\kappa}(k+1) = \omega_{Hj\kappa}(k) + u_{Hj\kappa}(k)\,, \quad \kappa = 1,2,...,n_o\,, \quad j = 1,2,...,n_H\,,$$
$$\omega_{Imn}(k+1) = \omega_{Imn}(k) + u_{Imn}(k)\,, \quad m = 1,2,...,n_I\,, \quad n = 1,2,...,n_H$$

or, in matrix notation

$$W_H(k+1) = W_H(k) + U_H(k)\,,$$
$$W_I(k+1) = W_I(k) + U_I(k)\,,$$

where $U_H(k) \in R^{n_H \times n_o}$ and $U_I(k) \in R^{n_I \times n_H}$ are the correction matrices, with column vectors $U_H^\kappa(k)$ and $U_I^n(k)$ respectively, updating the weight matrices at time k. Such column vectors are given by:

$$U_H^\kappa(k) = \begin{bmatrix} u_{H1\kappa}(k) \\ u_{H2\kappa}(k) \\ \vdots \\ u_{Hn_H\kappa}(k) \end{bmatrix}\,; \quad \kappa = 1,2,...,n_o$$

and

$$U_I^n(k) = \begin{bmatrix} u_{I1n}(k) \\ u_{I2n}(k) \\ \vdots \\ u_{In_In}(k) \end{bmatrix}\,; \quad n = 1,2,...,n_H$$

Description of the Algorithm. The following lemma will be needed in subsequent considerations.

Lemma 1. The error vector E(k) satisfies the following difference equation as a function of the input layer and hidden layer matrix update weights $U_H(k)$ and $U_I(k)$:

$$E(k+1) - E(k) = [W_H(k)]^T\{\Gamma[Y_H(k)] - \Gamma[Y_H(k) + [U_I(k)]^T X]\} - [U_H(k)]^T\Gamma[Y_H(k) + [U_I(k)]^T X]\,. \tag{4.2}$$

Proof. We have

$$\begin{aligned} E(k+1) - E(k) &= Y_o(k) - Y_o(k+1) \\ &= [W_H(k)]^T Z_H(k) - [W_H(k) + U_H(k))]^T Z_H(k+1) \\ &= [W_H(k)]^T [Z_H(k) - Z_H(k+1)] - [U_H(k)]^T Z_H(k+1) \\ &= [W_H(k)]^T\{\Gamma[Y_H(k)] - \Gamma[Y_H(k+1)]\} - [U_H(k)]^T\Gamma[Y_H(k+1)]\,. \end{aligned}$$

Since

$$Y_H(k+1) = [W_I(k+1)]^T X = [W_I(k) + U_I(k)]^T X = Y_H(k) + [U_I(k)]^T X$$

we have that

$$E(k+1) - E(k) = [W_H(k)]^T\{\Gamma[Y_H(k)] - \Gamma[Y_H(k) + [U_I(k)]^T X]\} - [U_H(k)]^T\Gamma[Y_H(k) + [U_I(k)]^T X]\,.$$

□

The new training algorithm for the generalized two-layer perceptron is presented in the following theorem.

Theorem 2. If the weight correction matrices $U_I(k)$ and $U_H(k)$ are respectively chosen as

$$U_I(k) = \frac{-2\Theta_1[X][Y_H(k)]^T}{X^T\Theta_1[X]}, \quad (X^T\Theta_1[X] \neq 0) \tag{4.3}$$

and

$$U_H(k) = -2\,W_H(k) - \frac{\Theta_2[Z_H(k)]\,\{AE(k)\}^T}{[Z_H(k)]^T\Theta_2[Z_H(k)]}, \quad (Z_H^T\Theta_2[Z_H] \neq 0) \tag{4.4}$$

where Θ_1 and Θ_2 are operators, then the learning error vector $E(k)$ satisfies the following asymptotically stable difference equation:

$$E(k+1) = (I - A)\,E(k) \tag{4.5}$$

where A is an $n_o \times n_o$ diagonal matrix given by

$$A = \begin{bmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \dots & \alpha_{n_o} \end{bmatrix} \tag{4.6}$$

such that $|1 - \alpha_\kappa| < 1 \quad \kappa = 1,2,\dots n_o$.

Proof. Note that the transpose of the weight correction matrix $U_I(k)$ is given by

$$[U_I(k)]^T = \frac{-2Y_H(k)\{\Theta_1[X]\}^T}{X^T\Theta_1[X]}.$$

Substituting this last expression into the error difference equation (4.2), of Lemma 1, and taking into account (4.1) one obtains

$$\begin{aligned} E(k+1) - E(k) &= 2[W_H(k)]^T\Gamma[Y_H(k)] + [U_H(k)]^T\Gamma[Y_H(k)] \\ &= 2[W_H(k)]^T Z_H(k) + [U_H(k)]^T Z_H(k) \\ &= [2\,W_H(k) + U_H(k)]^T Z_H(k). \end{aligned}$$

Substituting (4.4) into the above error vector difference equation yields the following asymptotically stable error dynamics

$$E(k+1) = (I - A)\,E(k).$$

□

Notice that A may also be chosen as an arbitrary nondiagonal matrix such that the matrix (I-A) has its eigenvalues in the open unit circle of the complex plane. A particularly simple form of (4.3) and (4.4) is obtained when Θ_1 and Θ_2 are chosen to be the identity operators.

Observe that the proposed expressions for the weight correction matrices are nonunique. Thus it is left to a designer to select formulae which would be most suitable for the problem at hand.

Example 2. In this example we test the proposed adaptation algorithm of Theorem 2 for a two-layer perceptron. The two-layer perceptron used in the following simulation experiment is depicted in Fig. 8. Here we take $\Gamma[X] = \text{SGN } X$, $\Theta_1(\cdot) = \Theta_2(\cdot) = \text{SGN}(\cdot)$. The weights of this perceptron will be adjusted according to the laws (4.3) and (4.4). Specifically:

$$\omega_{Imn}(k+1) = \omega_{Imn}(k) + u_{Imn}(k) \qquad m = 1,2,3,4$$
$$u_{Imn}(k) = -2\,\frac{y_{H_n}(k)\,\mathrm{sgn}(x_m)}{\sum_{j=1}^{4} |x_j|} \quad ; \quad n = 1,2$$

$$\omega_{Hj\kappa}(k+1) = \omega_{Hj\kappa}(k) + u_{Hj\kappa}(k) \qquad j = 1,2,$$
$$u_{Hj\kappa}(k) = -2\,\omega_{Hj\kappa}(k) - \frac{\alpha\, e(k)\, z_{H\kappa}(k)}{2} \quad ; \quad \kappa = 1\,.$$

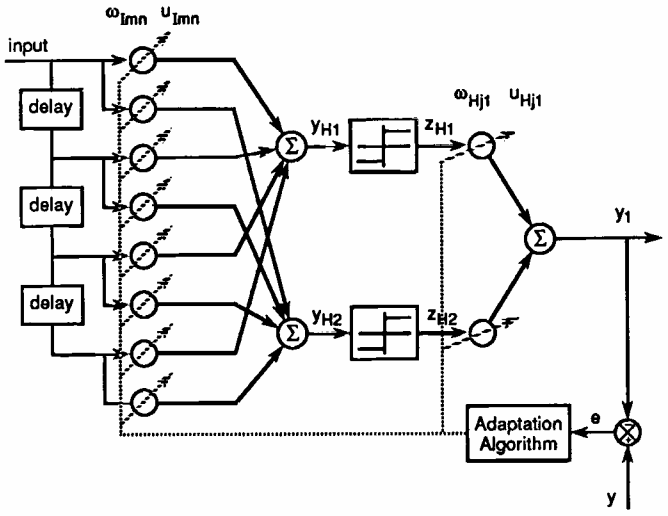


Fig. 8. A two-layer perceptron used in Example 2.

Our problem, as in Example 1, is to identify the behavior of an "unknown" dynamical systems (3.2) using the configuration shown in Fig. 2, where the structure of the perceptron in this simulation experiment is depicted in Fig. 8.

The input signal u(t), the output y(t) of the unknown system, the output $y_1(t)$ of the two-layered perceptron and the tracking error are shown in Fig. 9.

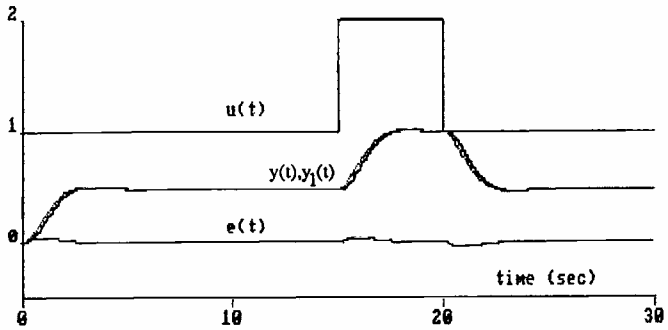


Fig. 9. Time history of u(t), y(t), $y_1(t)$, and e(t).

The weights adaptation time history of the two layer perceptron used in this experiment is shown in Fig. 10.

A substantial improvement over the single perceptron of Example 1 is obtained in reducing the error when the two-layer perceptron is used.

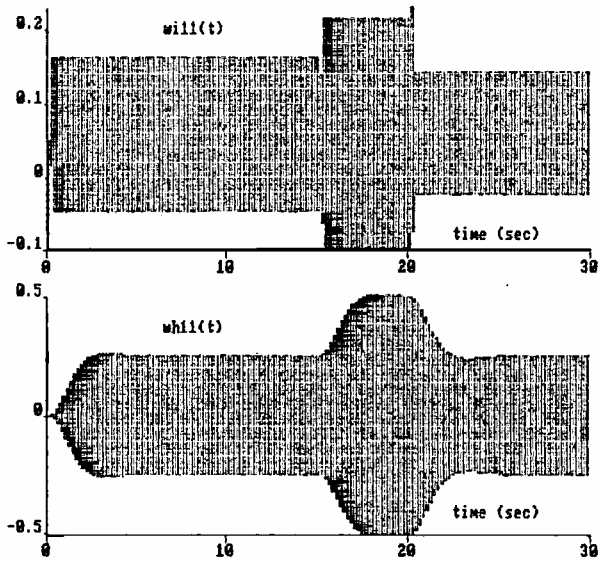


Fig. 10. Adaptation of the two-layer perceptron weights.

5. ADAPTATION ALGORITHMS FOR THREE-LAYER GENERALIZED PERCEPTRONS. In this Section we shall utilize similar type of arguments to those used in the previous Section to devise training algorithms for the three-layer feedforward neural networks. An example of such a network is shown in Fig. 11.

Notation and Definitions. The κ-th output of the network in Fig. 11 is obtained as the weighted sum of the outputs of the first hidden layer

$$y_{o\kappa} = \sum_{j=1}^{n_{1H}} \omega_{1Hj\kappa}(k)\, z_{1Hj}(k),\ \kappa = 1,2,...,n_o$$

where n_{1H} is the number of neurons in the first hidden layer, n_o is the number of the network outputs, and z_{1Hj} are the outputs of the neurons in the first hidden layer. The quantities $\omega_{1Hj\kappa}$ represent the interconnection weights between the first hidden layer and the outputs of the net. In vector notation

$$y_{o\kappa} = [W_{1H}^{\kappa}(k)]^T\, Z_{1H}(k), \quad \kappa = 1,2,...,n_o$$

where

$$W_{1H}^{\kappa}(k) = \begin{bmatrix} \omega_{1H1\kappa}(k) \\ \omega_{1H2\kappa}(k) \\ \vdots \\ \omega_{1Hn_{1H}\kappa}(k) \end{bmatrix},$$

and

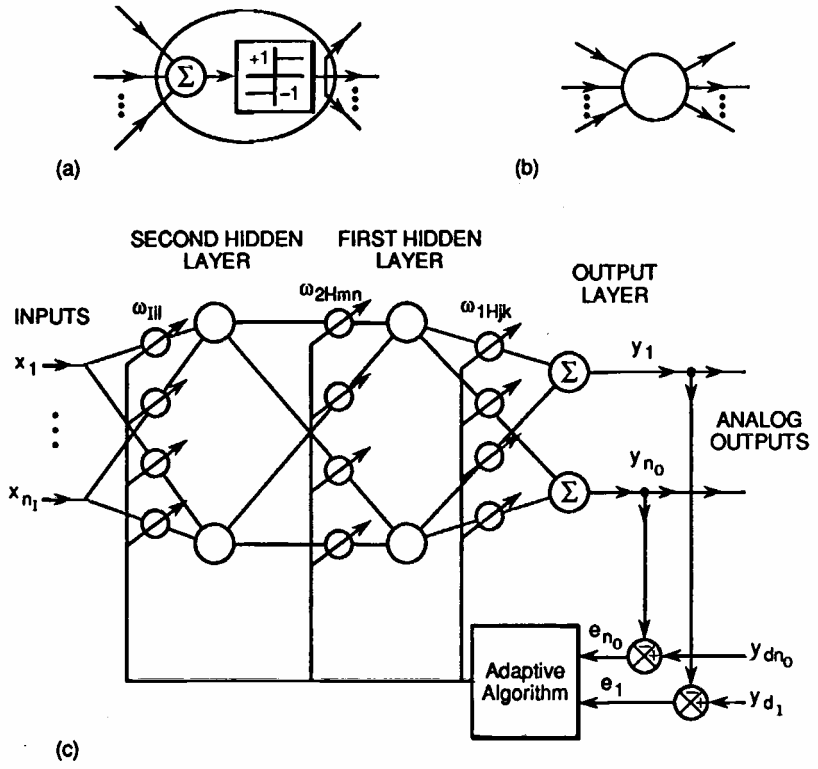


Fig. 11. (a) Model of a single neuron
(b) Symbol for a single neuron
(c) A three-layer adaptive perceptron.

$$Z_{1H}(k) = \begin{bmatrix} z_{1H_1}(k) \\ z_{1H_2}(k) \\ \vdots \\ z_{1Hn_{1H}}(k) \end{bmatrix} = \begin{bmatrix} \gamma_{11}[y_{1H1}(k)] \\ \gamma_{22}[y_{1H2}(k)] \\ \vdots \\ \gamma_{n_{1H}n_{1H}}[y_{1Hn_{1H}}(k)] \end{bmatrix} = \Gamma[Y_{1H}(k)] \, .$$

If we denote by W_{1H} the matrix of column vectors W_{1H}^{κ}, then the vector $Y_o(k)$ of the output components $y_{o\kappa}(k)$ is obtained as (see Fig. 12)

$$\boxed{Y_o(k) = [W_{1H}(k)]^T Z_{1H}(k)}$$

Note that $W_{1H} \in \mathbb{R}^{n_{1H} \times n_o}$.

The signals y_{1Hn} of the neurons in the first hidden layer are given by

$$y_{1Hn}(k) = \sum_{m=1}^{n_{2H}} \omega_{2Hmn}(k) z_{2Hm}(k)$$
$$= [W_{2H}^{n}(k)]^{T} Z_{2H}(k), \quad n = 1,2,\ldots,n_{1H}$$

where

$$W_{2H}^{n}(k) = \begin{bmatrix} \omega_{2H1n}(k) \\ \omega_{2H2n}(k) \\ \vdots \\ \omega_{2Hn_{2H}n}(k) \end{bmatrix},$$

and

$$Z_{2H}(k) = \begin{bmatrix} z_{2H1}(k) \\ z_{2H2}(k) \\ \vdots \\ z_{2Hn_{2H}}(k) \end{bmatrix}.$$

Denote by W_{2H} the matrix of column vectors W_{2H}^{n}. Then the vector of the neuron signals

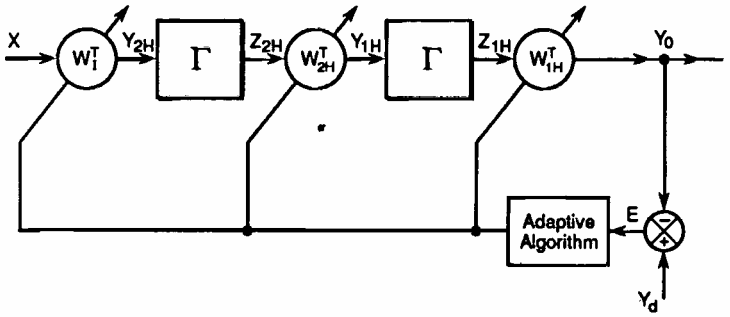


Fig. 12. Schematic representation of the three-layer adaptive feedforward neural network.

Y_{1H} of the first hidden layer is (see Fig. 12)

$$\boxed{Y_{1H}(k) = [W_{2H}(k)]^{T} Z_{2H}(k)}$$

Note that $W_{2H}(k) \in \mathbb{R}^{n_{2H} \times n_{1H}}$. Observe that

$$Z_{2H}(k) = \Gamma [Y_{2H}(k)] .$$

The components of the vector $Y_{2H}(k)$ are given by

$$y_{2Hi}(k) = \sum_{i=1}^{n_I} \omega_{Iii}(k) x_i$$
$$= [W_I^i(k)]^{T} X , \quad i = 1,\ldots,n_{2H}$$

where

$$W_I^i(k) = \begin{bmatrix} \omega_{I1i}(k) \\ \omega_{I2i}(k) \\ \vdots \\ \omega_{In_Ii}(k) \end{bmatrix}$$

and

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_1} \end{bmatrix}.$$

If we denote by W_I the matrix of column vectors $W_I^j(k)$ then

$$\boxed{Y_{2H}(k) = [W_I(k)]^T X}$$

where

$$W_I(k) \in \mathbb{R}^{n_1 \times n_{2H}}$$

Let Y_d denotes the vector of desirable output values with components, $y_{d\kappa}$; $\kappa = 1,2,...,n_o$. The error vector $E(k)$ at time k is then obtained as

$$E(k) = \begin{bmatrix} e_1(k) \\ \vdots \\ e_{n_o}(k) \end{bmatrix} = Y_d - Y_o(k)$$

Let the weights update law be represented by the following equations:

$$\omega_{1Hj\kappa}(k+1) = \omega_{1Hj\kappa}(k) + u_{1Hj\kappa}(k)\,,$$

$$\omega_{2Hmn}(k+1) = \omega_{2Hmn}(k) + u_{2Hmn}(k)\,,$$

$$\omega_{Ii1}(k+1) = \omega_{Ii1}(k) + u_{Ii1}(k)\,,$$

or, in matrix notation

$$W_{1H}(k+1) = W_{1H}(k) + U_{1H}(k)\,,$$

$$W_{2H}(k+1) = W_{2H}(k) + U_{2H}(k)\,,$$

$$W_I(k+1) = W_I(k) + U_I(k)\,,$$

where $U_{1H}(k) \in \mathbb{R}^{n_{1H} \times n_o}$, $U_{2H}(k) \in \mathbb{R}^{n_{2H} \times n_{1H}}$, and $U_I(k) \in \mathbb{R}^{n_1 \times n_{2H}}$ are the correction matrices updating the weight matrices at time k. In Fig. 12 we represent schematically the generalized three-layer adaptive perceptron using the above notation. Note that the nonlinear operator Γ does not have to be a diagonal one.

Description of the Algorithm. The following lemma will be used in the subsequent considerations

Lemma 2. The error vector $E(k)$ satisfies the following difference equation as a function of the input and hidden layers matrix update weights $U_{1H}(k)$, $U_{2H}(k)$, and $U_I(k)$:

$$\begin{aligned} E(k+1) - E(k) &= [W_{1H}(k)]^T \left\{ \Gamma\left[Y_{1H}(k)\right] - \Gamma\left\{[W_{2H}(k) + U_{2H}(k)]^T \Gamma(Y_{2H}(k) + [U_I(k)]^T X)\right\}\right\} \\ &\quad - [U_{1H}(k)]^T \Gamma\left\{[W_{2H}(k) + U_{2H}(k)]^T \Gamma(Y_{2H}(k) + [U_I(k)]^T X)\right\} \end{aligned} \tag{5.1}$$

Proof. We have

$$
\begin{aligned}
E(k+1) - E(k) &= Y_o(k) - Y_o(k+1) \\
&= [W_{1H}(k)]^T Z_{1H}(k) - [W_{1H}(k+1)]^T Z_{1H}(k+1) \\
&= [W_{1H}(k)]^T Z_{1H}(k) - [W_{1H}(k) + U_{1H}(k)]^T Z_{1H}(k+1) \\
&= [W_{1H}(k)]^T \{\Gamma[Y_{1H}(k)] - \Gamma[Y_{1H}(k+1)]\} \\
&\quad - [U_{1H}(k)]^T \Gamma[Y_{1H}(k+1)] . \qquad (5.2)
\end{aligned}
$$

Note that

$$
\begin{aligned}
Y_{1H}(k+1) &= [W_{2H}(k+1)]^T Z_{2H}(k+1) \\
&= [W_{2H}(k) + U_{2H}(k)]^T \Gamma[Y_{2H}(k+1)] \\
&= [W_{2H}(k) + U_{2H}(k)]^T \Gamma([W_I(k) + U_I(k)]^T X) .
\end{aligned}
$$

Substituting the above equation into (5.2) yields

$$
\begin{aligned}
&E(k+1) - E(k) \\
&= [W_{1H}(k)]^T \left\{ \Gamma[Y_{1H}(k)] - \Gamma\left\{ [W_{2H}(k) + U_{2H}(k)]^T \Gamma([W_I(k) + U_I(k)]^T X) \right\} \right\} \\
&\quad - [U_{1H}(k)]^T \Gamma\left\{ [W_{2H}(k) + U_{2H}(k)]^T \Gamma([W_I(k) + U_I(k)]^T X) \right\} .
\end{aligned}
$$

The last equation is equivalent to (5.1). □

We shall now present the new training algorithm for the generalized three-layer perceptron (see Fig. 12) in the following theorem.

Theorem 3. If the weight correction matrices $U_I(k)$, $U_{2H}(k)$, and $U_{1H}(k)$ are respectively chosen as

$$
U_I(k) = -\frac{2\{\Theta_1[X]\}[Y_{2H}(k)]^T}{X^T \Theta_1[X]} , \quad (X^T \Theta_1[X] \neq 0) \qquad (5.3)
$$

$$
U_{2H}(k) = -\frac{2\{\Theta_2[Z_{2H}(k)]\}[Y_{1H}(k)]^T}{Z_{2H}^T(k)\, \Theta_2[Z_{2H}(k)]} , \quad (Z_{2H}^T \Theta_2[Z_{2H}] \neq 0) \qquad (5.4)
$$

and

$$
U_{1H}(k) = \frac{\{\Theta_3[Z_{1H}(k)]\}[A\, E(k)]^T}{Z_{1H}^T(k)\, \Theta_3[Z_{1H}(k)]} , \quad (Z_{1H}^T \Theta_3[Z_{1H}] \neq 0) \qquad (5.5)
$$

where Θ_1, Θ_2, Θ_3 are operators, then the learning error vector $E(k)$ satisfies the following asymptotically stable difference equation

$$
E(k+1) = (I - A)E(k) ,
$$

where A is an $n_o \times n_o$ diagonal matrix chosen as in (4.6).

Proof. Note that the transpose of the weight correction matrix $U_I(k)$ is given by

$$[U_I(k)]^T = -\frac{2\, Y_{2H}(k)\{\Theta_1[X]\}^T}{\{\Theta_1[X]\}^T X}.$$

Substituting this last expression into the error difference equation (5.1) and utilizing (4.1) gives

$$\begin{aligned} E(k+1) - E(k) &= [W_{1H}(k)]^T \left\{ Z_{1H}(k) + \Gamma(Y_{1H}(k) + [U_{2H}(k)]^T Z_{2H}(k)) \right\} \\ &+ [U_{1H}(k)]^T \Gamma \left\{ Y_{1H}(k) + [U_{2H}(k)]^T Z_{2H}(k) \right\}. \end{aligned} \tag{5.6}$$

Upon substituting the transpose of the weight correction matrix $U_{2H}(k)$ into (5.6) yields

$$\begin{aligned} &E(k+1) - E(k) \\ &= -[U_{1H}(k)]^T \Gamma[Y_{1H}(k)] = -[U_{1H}(k)]^T Z_{1H}(k). \end{aligned} \tag{5.7}$$

Finally, substituting the transpose of the weight correction matrix $U_{1H}(k)$ into (5.7) we obtain $E(k+1) = (I - A)\, E(k)$. □

Notice again that A may also be chosen as an arbitrary nondiagonal matrix such that the matrix $[I - A]$ has its eigenvalues in the open unit circle of the complex plane.

A particularly simple form of the weight correction matrices $U_I(k)$, $U_{2H}(k)$, and $U_{1H}(k)$ is obtained when Θ_1, Θ_2 and Θ_3 in (5.3)-(5.5) are chosen to be the identity operators.

Observe that the expressions for the weight correction matrices are not unique.

6. CONCLUSIONS. In this paper training procedures have been presented for a class of feedforward neural networks. The class of neural networks we have considered can be viewed as generalized perceptrons. The learning parameters are adjusted is such a way so that the error between the desired and actual outputs satisfies a stable difference equation. Research is now underway to apply the proposed adaptation algorithms to the control of nonlinear dynamic processes.

REFERENCES

Albus, J. S. (1975). A new approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC), *Transactions of the ASME, J. Dynamic Systems, Measurement, and Control,* **97**, Series G, No. 3, pp. 220-227.

Åström, K. J. and Wittenmark, B. (1989). *Adaptive Control,* Addison-Wesley, Reading, Mass.

Grossberg, S. (1988). Nonlinear neural networks: Principles, mechanisms, and architectures, *Neural Networks,* **11**, No. 1, pp. 17-61.

Haykin, S. (1984). *Introduction to Adaptive Filters,* Macmillan Publishing Co., New York.

Lippmann, R. P. (1987). An introduction to computing with neural nets, *IEEE ASSP Magazine,* **4**, No. 2, pp. 4-22.

Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Networks,* **1**, No. 1, pp. 4-27.

Pao, Y. H. (1989). *Adaptive Pattern Recognition and Neural Networks,* Addison-Wesley, Reading, Mass.

Rangwala, S. S. and Dornfeld, D. A. (1989). Learning and optimization of machining operations using computing abilities of neural networks, *IEEE Trans. Systems,*

Man, and Cybernetics, **19**, No. 2, pp. 299-314.

Rumelhart, D. E. McClelland, J. L. and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* "A Bradford book", The MIT Press, Cambridge, Mass.

Sira-Ramirez, H. J. and Żak, S. H. (1991). The adaptation of perceptrons with applications to inverse dynamics identification of unknown dynamic systems, to appear in *IEEE Trans. Systems, Man, and Cybernetics.*

Tank, D. W. and Hopfield, J. J. (1986). Simple "neural" optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit, *IEEE Trans. Circuits and Systems,* **CAS-33**, No. 5, pp. 533-541.

Widrow, B. and Lehr, M. A. (1990). 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation, *Proceedings of the IEEE,* 78, No. 9, pp. 1415-1442.

Widrow, B. and Winter, R. (1988). Neural nets for adaptive filtering and adaptive pattern recognition, *Computer,* **21**, No. 3, pp. 25-39.