Technical Report
813

Learning Algorithms for the Multilayer Perceptron

AD-A202 682

M.D. Eggers
T.S. Khuon

28 October 1988

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Prepared for the Department of the Army
under Electronic Systems Division Contract F19628-85-C-0002.

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This program is sponsored by the U.S. Army Strategic Defense Command under Air Force Contract F19628-85-C-0002.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

LEARNING ALGORITHMS FOR THE MULTILAYER PERCEPTRON

M.D. EGGERS
T.S. KHUON
Group 93

TECHNICAL REPORT 813

28 OCTOBER 1988

LEXINGTON MASSACHUSETTS

ABSTRACT

Central to the development of adaptive pattern processing algorithms (adaptive filters) for random problems — problems where statistics are unknown a priori and/or explicit rules governing behavior cannot be extracted in a reductionist manner — is the pursuit of adaptive architectures for associating arbitrary inputs to outputs. Such "associative memories" are important for providing the mathematical mapping (transfer function) relating inputs to outputs arising from implicit relationships found in a given training ensemble. The adaption of these filters or architectures during training is guided by a learning algorithm, mathematically derived from an objective function to ensure good association properties.

The subject of this paper is an investigation of a class of learning algorithms for the highly parallel multilayer perceptron architecture used in an associative memory context. By controlling the scheduling of patterns presented during training, a generalized class of learning algorithms are shown to result. Specific realizations of the generalized algorithm include steepest descent (parameters adapted following presentation of *all* training patterns), Rumelhart back propagation (parameters adapted following presentation of *each* pattern), and a new algorithm which captures in part the benefits of both, less parameter adaption and faster convergence, by gradually varying the number of patterns presented per parameter adaption. A systematic derivation of the fundamental steepest descent algorithm for the multilayer perceptron is included for clarification, although related learning algorithms have been formulated by others.

Learning results are presented utilizing the algorithms on a simple benchmark association problem. Although performance is similar amongst the algorithms, the relative computational burden differs substantially. Of the algorithms investigated, steepest descent requires the least computation, while back propagation is the most demanding.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

1.1 OBJECTIVE

The objective of a learning algorithm is to adjust the free parameters of an adaptive system architecture to achieve satisfactory system performance in a given problem domain. Often the algorithm can be derived from an objective or risk function, which mathematically incorporates the desired objectives of the problem solution. And by minimizing the objective function through execution of the resulting learning algorithm, a set of free parameters result yielding the defined optimal system. The specific embodiment presented entails deriving a learning algorithm for an adaptive system used for associating arbitrary input and output vector pairs. This arbitrary association problem — associative memory — is quite useful in the *classification of patterns* required for many pattern processing applications including speech recognition, machine vision, and decision making.

Assume an arbitrary set of input/output vector pairs comprising the training set T is given by

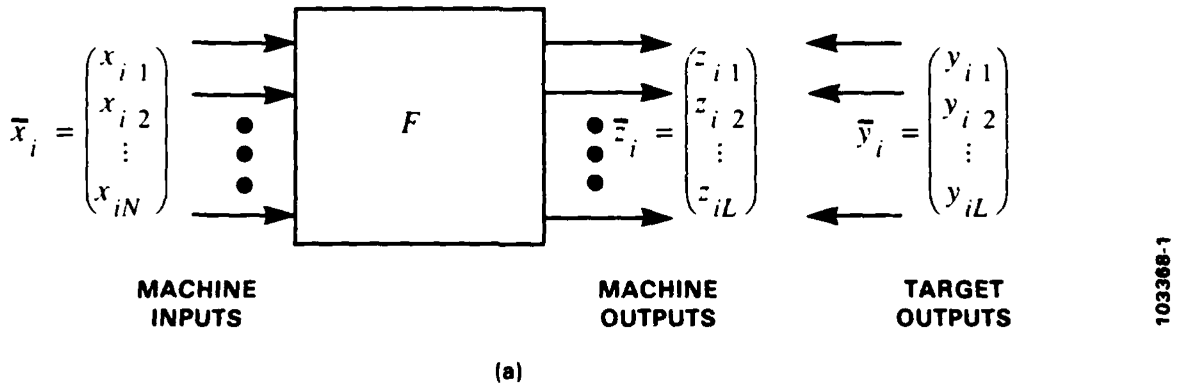
$$T = \{(\bar{x}_1, \bar{y}_1), (\bar{x}_2, \bar{y}_2), \ldots, (\bar{x}_M, \bar{y}_M)\}, \qquad (1)$$

the specific objective being to derive a learning algorithm which best associates the input/output pairs found in the training set. To rigorously derive the best algorithm, the system architecture and objective function must first be defined. Desired objectives include perfect recall and generalization. Mathematically, these objectives can be satisfied by requiring the learning algorithm to adapt the free parameters of the system or machine towards a realization that results in a mapping function F (see Fig. 1.) such that

$$F(\bar{x}_i) = \bar{y}_i \qquad \forall\ i \qquad \text{(perfect recall)} \qquad (2)$$

$$F(\bar{x}_i + \bar{n}) = \bar{y}_i \qquad \forall\ i \qquad \text{(generalization).} \qquad (3)$$

Note, caution must be exercised when interpreting generalization capability. First, generalization is used here in a strict mathematical sense. That is, if the system is excited by a novel input, being close in norm to the masked prototype ($\bar{x} = \bar{x}_i + \bar{n}$ where $\bar{x} \notin T, | \bar{n} | < \delta$) and a correct output is obtained, then the system is said to possess generalization capability. Good generalization is achieved by maintaining large distances in the input space between the inputs having different outputs, and operating well within the memory capacity [1].



$\bar{x}_i = (x_{i1}, x_{i2}, \vdots, x_{iN})$
F
$\bar{z}_i = (z_{i1}, z_{i2}, \vdots, z_{iL})$
$\bar{y}_i = (y_{i1}, y_{i2}, \vdots, y_{iL})$
MACHINE INPUTS
MACHINE OUTPUTS
TARGET OUTPUTS
(a)
103368-1

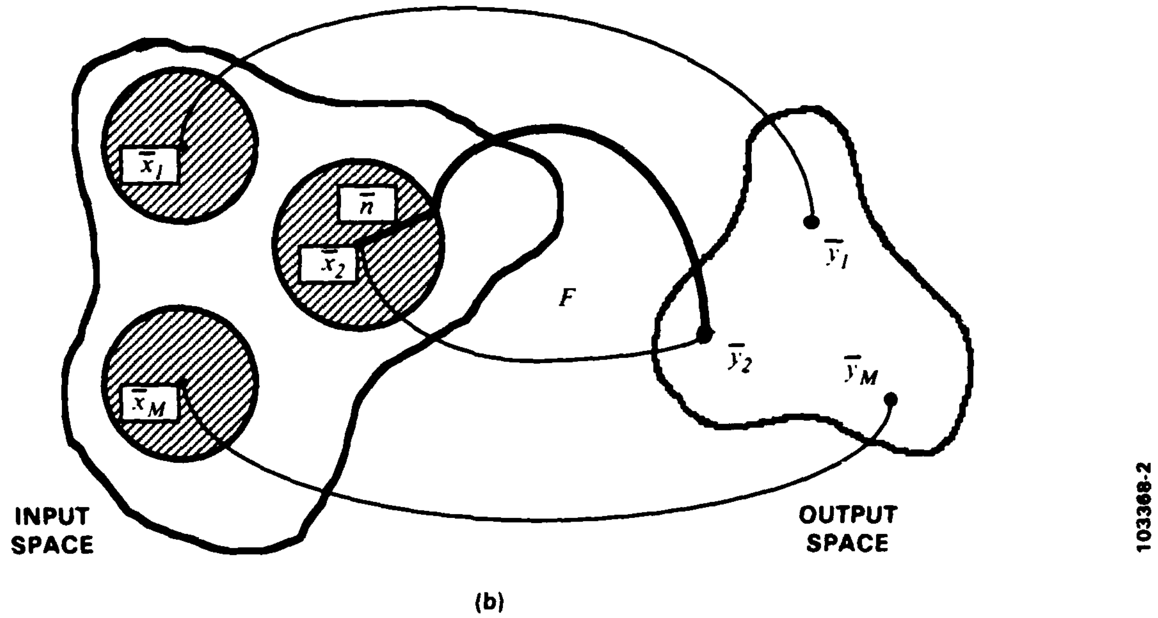


Fig. 1. (a) Machine notation with inputs, outputs and target (desired) outputs.
(b) Mathematical associative memory mapping.

Confusion often arises when a semantic interpretation of generalization is applied to the mathematical system. Basically, semantic generalization is the ability to generalize semantic concepts. As an example, consider teaching a person to identify trees. The training might consist of presenting the student several species of trees with the respective correct labels (i.e., birch, pine, oak). Now if the student is then presented a novel tree, say an apple tree, and correctly recognizes it as a tree, then the student has demonstrated semantic generalization. And indeed this type of generalization, giving rise to robust recognition, is intensely sought by machine intelligence researchers.

The distinction between mathematical and semantic generalization lies in the representation of the data. For if a representation is chosen for the tree learning problem whereby the apple tree input vector is within the region of the input space mapped to the output region designating "tree," mathematical generalization yields the desired semantic generalization. Therefore, the representation of the data determines whether semantic generalization is achieved in a system possessing mathematical generalization capabilities.

1.2 APPROACH

With the desired objectives of the system delineated, the specific system architecture and objective function must be defined, thereby restricting the class of solutions for the association problem.

The system architecture chosen is the multilayer perceptron [2,3]. Illustrated in Fig. 2., the architecture is comprised of processing units and interconnections. Each interconnection has an associated connection strength or weight. The complete collection of weights comprise the set of free parameters for the adaptive system. Each processing unit first performs a weighted accumulation of the respective inputs and bias value, then passes the result through a threshold function.

Reasons supporting the architecture choice include fast system operation, achieved by the highly parallel circuitry. Furthermore, the multilayer architecture is quite expressive. That is, most mapping functions F can be realized with the architecture shown. In fact, Kolmogorov proved that continuous mapping functions of several inputs can be expressed by a three layer network [4]. However the theorem and extensions [5,6] are existence proofs, and hence no practical learning algorithms are offered to direct the selection of the connection weights. However with the processing units shown in Fig. 2b., together with connection strengths given by the learning algorithms to be discussed, the multilayer perceptron architecture has been empirically found (via computer simulation) capable of mapping complex association problems [7-10].

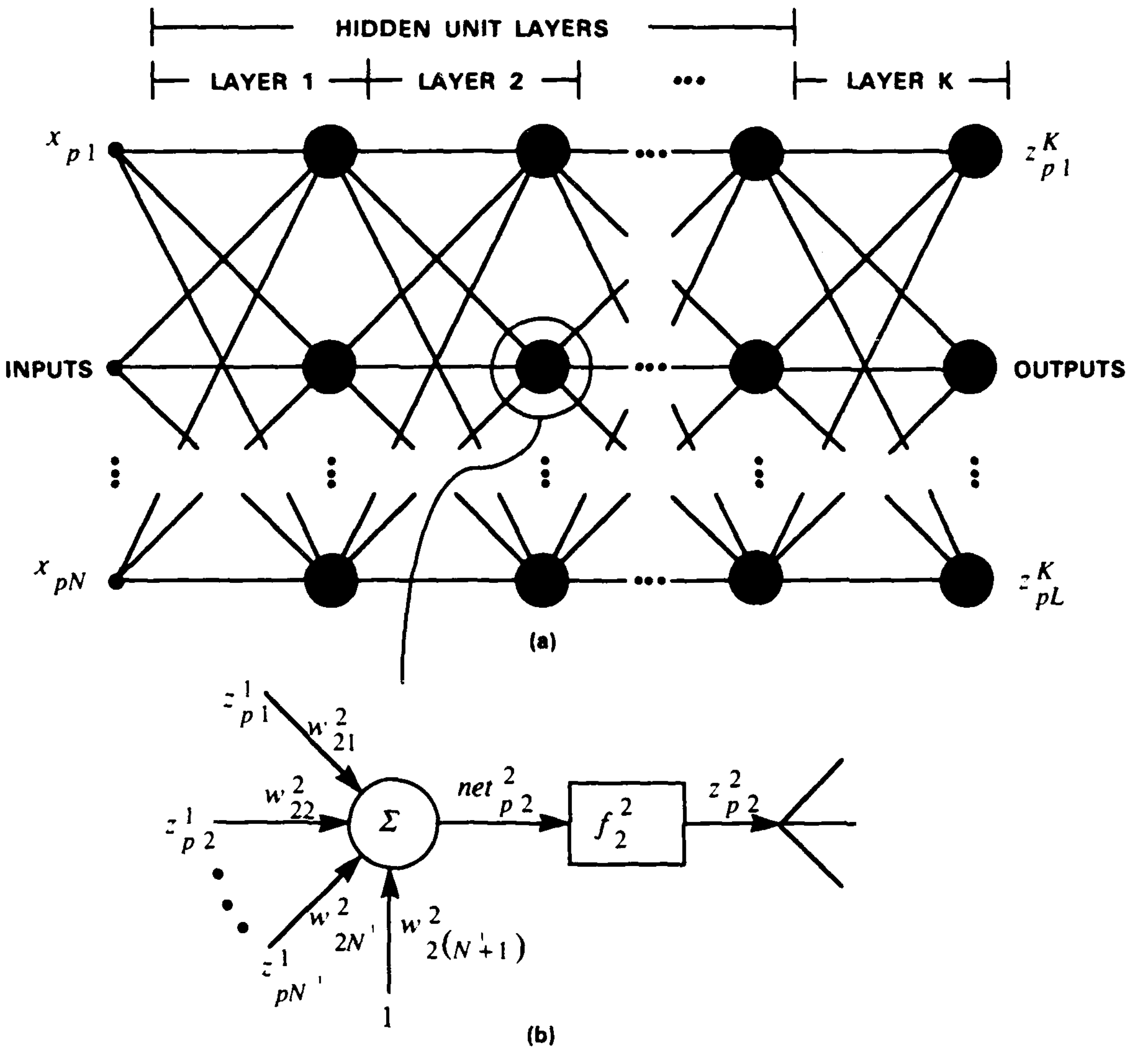


Fig. 2. (a) Multilayer perceptron architecture and (b) processing unit 2 on layer 2 (N^k = number of processing units on layer k).

Next an objective (equivalently risk, energy, performance) function must be specified which incorporates the desired objectives in a mathematical expression. Again for the association problem, the desire is to construct a system whose output best approximates the behavior found in the training data, resulting in perfect recall and generalization. The objective function utilized is the sum squared error over all training patterns

$$E = \sum_{patterns} E_p \tag{4}$$

where

$$E_p = \frac{1}{2} \sum_l \left(y_{pl} - z^K_{pl} \right)^2 \tag{5}$$

represents the error per pattern as the squared difference of the machine and desired outputs when an input pattern x_p is presented. Consequently the approach involves deriving a learning algorithm that selects the free parameters (weights w^k_{ji}) to minimize the overall sum squared error (4), thereby forcing the outputs of the system to mimic the outputs displayed in the training set.

Exploiting the general architecture and objective function, a number of learning algorithms have been developed. First, Widrow and Hoff [11] developed an adaptive network wherein the threshold function $f(x)$ is linear. The resulting LMS (least mean square) algorithm and related extensions have proven successful in numerous signal processing applications requiring adaptive filters [12]. Parker [13] has developed a least squares solution which simultaneously minimizes the changes to internal architecture parameters during training. Thus the algorithm, by the method of Lagrange multipliers, actually represents a solution to a constrained minimization problem.

The current approach involves first presenting a systematic derivation of the steepest descent learning algorithm for the described architecture with an arbitrary nonlinear yet differentiable threshold function $f(x)$, and accompanying squared error objective function. Next modifications of the resulting algorithm are shown to yield the Rumelhart back propagation algorithm [7] and a new algorithm which captures in part the benefits of both steepest descent (less weight adaptions) and back propagation (faster convergence). Results are then presented comparing the performance of the algorithms on a benchmark association problem. Following, the relative computation burden is assessed and the sensitivity of the proposed algorithm to additional a priori information is demonstrated.

2. LEARNING ALGORITHMS

A systematic derivation of the steepest descent learning algorithm for the multilayer perceptron architecture with accompanying sum squared error objective function is presented. Following, a generalized learning algorithm is introduced that reduces to a host of special cases, including steepest descent and Rumelhart back propagation, by simply controlling the manner in which the training patterns are presented.

2.1 DERIVATION

The minimization of the objective function with respect to the connection strengths $\left(w_{ji}^{k}\right)$ is conducted numerically by a steepest descent algorithm (see Appendix for basic formulation), with the weight adaption given by

$$w_{ji}^{k}(n+1) = w_{ji}^{k}(n) + \Delta w_{ji}^{k}(n) \tag{6}$$

where

$$\Delta w_{ji}^{k}(n) = -\eta \left.\frac{\partial E}{\partial w_{ji}^{k}}\right|_{\overline{w}(n)} \tag{7}$$

The iterative process in initiated with $\overline{w}(o)$, a pseudorandom vector. The algorithm (6,7) simply forms a new weight value by stepping from the present position in the direction of steepest descent, wherein the size of the step is governed by the learning rate η.

Clearly the learning algorithm (6,7) is completely specified once the gradient of the error measure is derived for the multilayer system architecture. Proceeding, computing the gradient

$$\frac{\partial E}{\partial w_{ji}^{k}} = \sum_{p} \frac{\partial Ep}{\partial w_{ji}^{k}} \tag{8}$$

requires computing the partial derivative of the error per pattern with respect to the weights, thus by (5)

$$\frac{\partial Ep}{\partial w_{ji}^{k}} = \sum_{l}\left(y_{pl} - z_{pl}^{K}\right)\frac{\partial}{\partial w_{ji}^{k}}\left(y_{pl} - z_{pl}^{K}\right)$$

$$= -\sum_{l}\left(y_{pl} - z_{pl}^{K}\right)\frac{\partial z_{pl}^{K}}{\partial w_{ji}^{k}} \qquad (9)$$

which in turn necessitates calculating the partial derivatives of the output units (layer K) with respect to the weights. Recall from the network architecture, the expression for the lth output unit is

$$z_{pl}^{K} = f_{l}^{K}\left(net_{pl}^{K}\right) \qquad (10)$$

where

$$net_{pl}^{K} \equiv \sum_{m} w_{lm}^{K} z_{pm}^{K-1} \qquad (11)$$

$$f_{l}^{K}(q) = \frac{1}{1+e^{-q}}. \qquad (12)$$

Notice from the architecture (Fig. 2.), net_{pl}^{K} represents the net input to unit l of layer K when excited with input pattern $\bar{x}_{p}$, while f_{l}^{K} is a differentiable threshold function for unit l of layer K. Consequently, using the chain rule on (10)

$$\frac{\partial z_{pl}^{K}}{\partial w_{ji}^{k}} = \left\{f_{l}^{\prime K}\left(net_{pl}^{K}\right)\right\}\frac{\partial net_{pl}^{K}}{\partial w_{ji}^{k}}. \qquad (13)$$

Notice the partial derivative of the net input to the output unit must now be computed with respect to the weight w_{ji}^{k}. And depending upon where the specific weight w_{ji}^{k} occurs in the network architecture, two algorithms result.

2.1.1. Case 1. Hidden $\leftrightarrow$ Output

Consider the weights connecting units from the $K - 1$ layer (last hidden unit layer) to the Kth layer (output layer). Here the layer index on such weights becomes $k = K$ yielding

$$\frac{\partial}{\partial w_{ji}^{K}} net_{pl}^{K} = \sum_{m} z_{pm}^{K-1} \frac{\partial w_{lm}^{K}}{\partial w_{ji}^{K}}$$

$$= \sum_{m} z_{pm}^{K-1} \delta_{l-j} \delta_{m-i}$$

$$= z_{pi}^{K-1} \delta_{l-j} \qquad (14)$$

where

$$\delta_q = \begin{cases} 1 & q = 0 \\ 0 & q \neq 0 \end{cases} \qquad (15)$$

is the Kroneker delta function. Hence, the steepest descent learning rule for the last layer of weights is given by (6) and (7) with

$$\frac{\partial E}{\partial w_{ji}^{K}} = -\sum_{p} \sum_{l} \left(y_{pl} - z_{pl}^{K} \right) f_{l}^{\prime K}\left(net_{pl}^{K} \right) z_{pi}^{K-1} \delta_{l-j}$$

$$= -\sum_{p} \left(y_{pj} - z_{pj}^{K} \right) f_{j}^{\prime K}\left(net_{pj}^{K} \right) z_{pi}^{K-1}$$

$$= -\sum_{p} \Delta_{pj}^{K} z_{pi}^{K-1} \qquad (16)$$

where

$$\Delta_{pj}^{K} = \left(y_{pj} - z_{pj}^{K} \right) f_{j}^{\prime K}\left(net_{pj}^{K} \right) \qquad (17)$$

represents the error signal derived from the jth output unit upon the presentation of pattern $\bar{x}_p$. Consequently, the weights w_{ji}^K in the last layer are adapted following the presentation of *all* training patterns according to

$$\Delta w_{ji}^K(n) = \eta \sum_p \Delta_{pj}^K z_{pi}^{K-1} \quad . \tag{18}$$

2.1.2. Case 2. Hidden ↔Hidden and Input ↔Hidden

Next consider the weights connecting units within the hidden unit layers. These weights are changed sequentially in layers, beginning with the weights between hidden units in layers $K-2$ and $K-1$, proceeding backwards until reaching the first hidden units in the layer coupled to the inputs. Beginning with the weights between hidden unit layers $K-2$ and $K-1$, the partial derivative of the error per pattern with respect to the weight w_{ji}^{K-1}, following (9) through (13), is

$$\frac{\partial Ep}{\partial w_{ji}^{K-1}} = -\sum_l \left(y_{pl} - z_{pl}^K\right) \bullet f_l'^K\left(net_{pl}^K\right) \frac{\partial net_{pl}^K}{\partial w_{ji}^{K-1}} \tag{19}$$

where now

$$\frac{\partial net_{pl}^K}{\partial w_{ji}^{K-1}} = \sum_m w_{lm}^K \frac{\partial z_{pm}^{K-1}}{\partial w_{ji}^{K-1}} \tag{20}$$

Substituting the expression for the mth unit of layer $K-1$, the resulting term necessary for evaluating (20) becomes

$$\frac{\partial}{\partial w_{ji}^{K-1}} z_{pm}^{K-1} = \frac{\partial}{\partial w_{ji}^{K-1}} f_m^{K-1}\left(net_{pm}^{K-1}\right)$$

$$= f_m'^{K-1}\left(net_{pm}^{K-1}\right) \frac{\partial}{\partial w_{ji}^{K-1}} \left(\sum_q w_{mq}^{K-1} z_{pq}^{K-2}\right)$$

$$= f_m'^{K-1}\left(net_{pm}^{K-1}\right) \sum_q z_{pq}^{K-2} \frac{\partial}{\partial w_{ji}'^{K-1}} w_{mq}'^{K-1}$$

$$= f_m'^{K-1}\left(net_{pm}^{K-1}\right) z_{pi}^{K-2} \delta_{m-j} \ . \qquad (21)$$

Finally, substituting (21), (20), and (19) into (8) yields the gradient

$$\frac{\partial E}{\partial w_{ji}'^{K-1}} = -\sum_p \sum_l \left(y_{pl} - z_{pl}^K\right) f_l'^K\left(net_{pl}^K\right)$$

$$\bullet \sum_m w_{lm}'^K f_m'^{K-1}\left(net_{pm}^{K-1}\right) z_{pi}^{K-2} \delta_{m-j}$$

$$= -\sum_p \sum_l \Delta_{pl}^K w_{lj}'^K f_j'^{K-1}\left(net_{pj}^{K-1}\right) z_{pi}^{K-2}$$

$$= -\sum_p \left(\sum_l w_{lj}'^K \Delta_{pl}^K\right) f_j'^{K-1}\left(net_{pj}^{K-1}\right) z_{pi}^{K-2}$$

$$= -\sum_p \Delta_{pj}^{K-1} z_{pi}^{K-2} \qquad (22)$$

where

$$\Delta_{pj}^{K-1} = \left(\sum_l w_{lj}'^K \Delta_{pl}^K\right) f_j'^{K-1}\left(net_{pj}^{K-1}\right) \qquad (23)$$

and thus the error signal Δ_{pj}^{K-1} driving the adaption of the weights in the $K-1$ layer is expressed recursively as a weighted sum of the error signals Δ_{pl}^K computed in the previous layer. This recursion (23) allows the adaption of the weights hidden between layers of hidden units which do not have explicit target values, as do the output units.

Consequently, the weights w_{ji}^{K-1} in the $K-1$ layer are adapted following the presentation of *all* training patterns according to

$$\Delta w_{ji}^{K-1}(n) = \eta \sum_{p} \Delta_{pj}^{K-1} z_{pi}^{K-2} . \tag{24}$$

This procedure is repeated layer by layer until the first layer is reached, upon which the weights are adapted by

$$\Delta w_{ji}^{1}(n) = \eta \sum_{p} \Delta_{pj}^{1} z_{pi}^{0} \tag{25}$$

where

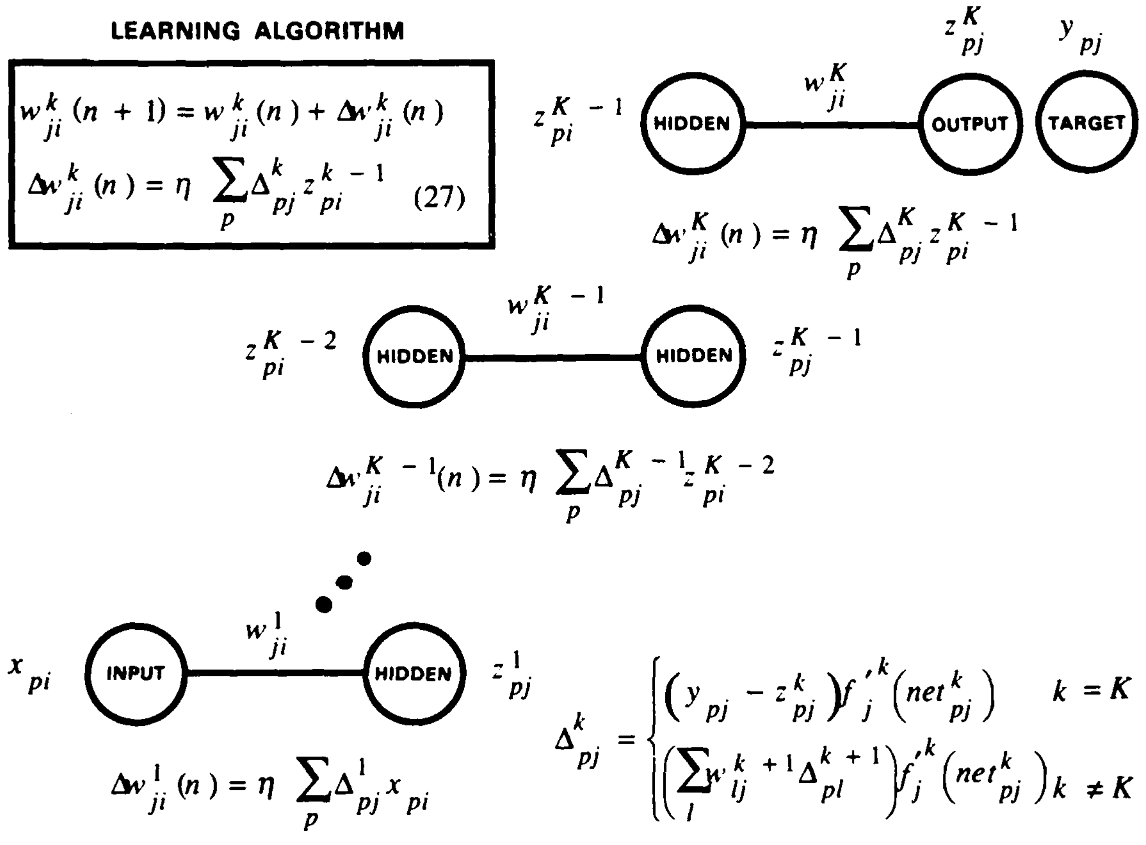
$$z_{pi}^{0} \equiv x_{pi} \qquad i = 1, 2, \ldots, N . \tag{26}$$

In summary, the learning procedure consists of cycling through two phases repeatedly until appropriate performance criteria are satisfied. The initial phase — forward propagation — entails passing an input training pattern $\bar{x}_p$ through the forward circuitry to obtain the network output $\bar{z}_p^K$. Next, the back propagation phase is conducted by forming error signals Δ_{pj}^{K} between the output $\bar{z}_p^K$ and the desired target value $\bar{y}_p$. Now in a recursive fashion demonstrated in Fig. 3. the error signals derived from the output layer are subsequently used to form new error signals Δ_{pj}^{K-1}, which guide weight adaptions in layer $(K-1)$ per (24). The back propagation recursion is completed when the first hidden unit layer is reached. Following the completion of one training cycle (forward propagation $\rightarrow$ back propagation), a new training pattern is presented and the cycle is repeated. Cycling through all training patterns once allows the adaption of the weights according to (27), and is referred to as a training sweep.

The complete circuitry required for the forward and backward propagation phases is shown in Fig.4. for a network with two hidden units. As observed, the compromise for the fast parallel forward circuitry (shown in bold) is the complicated back propagation feedback circuit. However once trained, the weights are fixed and thus only the forward circuitry is necessary for operation.

LEARNING ALGORITHM

$$\boxed{\begin{aligned} w^k_{ji}(n+1) &= w^k_{ji}(n) + \Delta w^k_{ji}(n) \\ \Delta w^k_{ji}(n) &= \eta \sum_p \Delta^k_{pj} z^{k-1}_{pi} \qquad (27) \end{aligned}}$$



$$\Delta w^K_{ji}(n) = \eta \sum_p \Delta^K_{pj} z^{K-1}_{pi}$$

$$\Delta w^{K-1}_{ji}(n) = \eta \sum_p \Delta^{K-1}_{pj} z^{K-2}_{pi}$$

$$\Delta w^1_{ji}(n) = \eta \sum_p \Delta^1_{pj} x_{pi}$$

$$\Delta^k_{pj} = \begin{cases} \left(y_{pj} - z^k_{pj}\right) f'^k_j\left(net^k_{pj}\right) & k = K \\ \left(\sum_l w^{k+1}_{lj} \Delta^{k+1}_{pl}\right) f'^k_j\left(net^k_{pj}\right) & k \neq K \end{cases} \qquad (28)$$

103368-4

Fig. 3. Schematic of learning procedure.

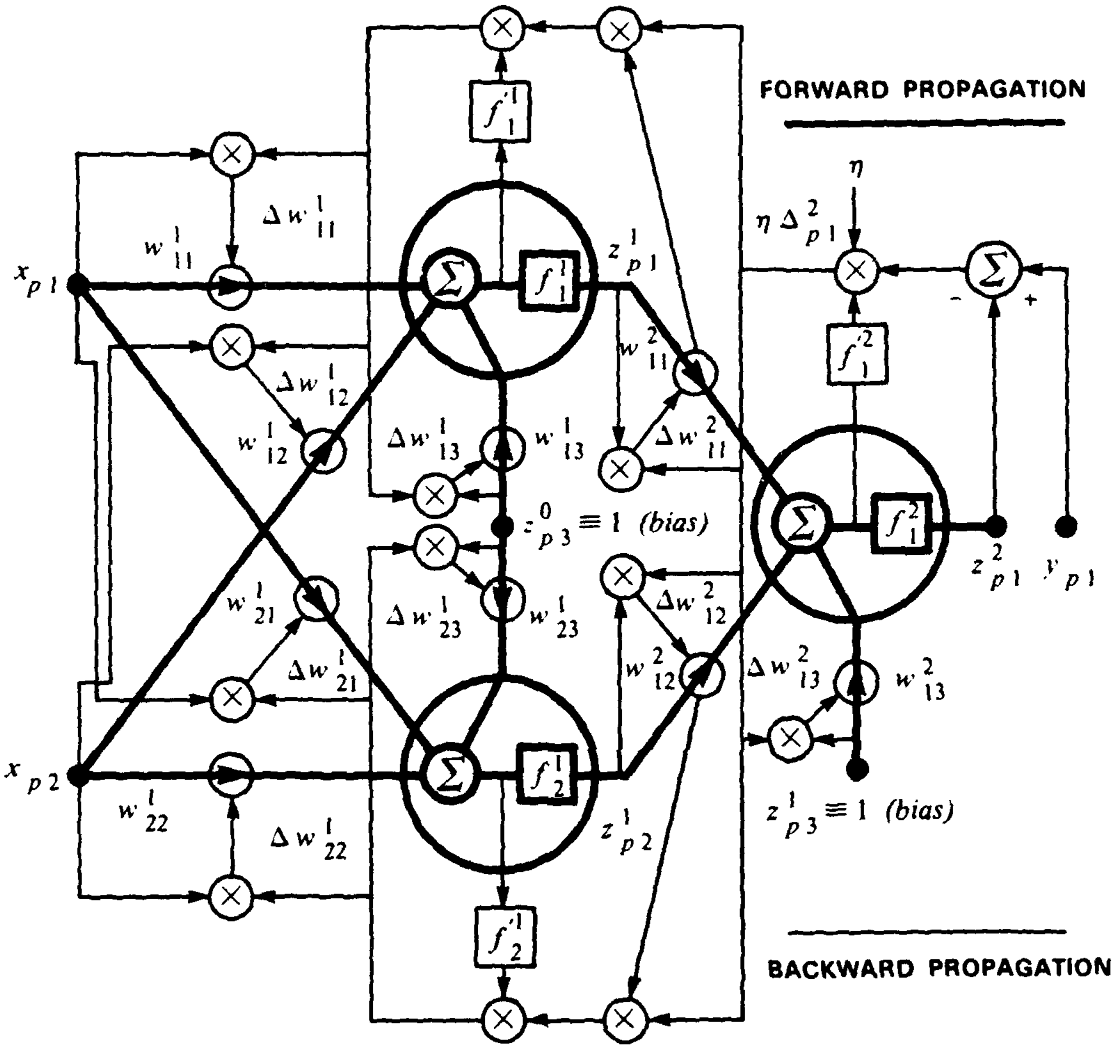


Fig. 4. Learning circuitry for a two layer, two hidden unit network.

2.2 GENERALIZATION AND SPECIAL CASES

To motivate the generalization of the learning algorithm, consider the weight adaption process as a numerical search through a multidimensional weight space. The trajectory is governed by the general weight adaption rule

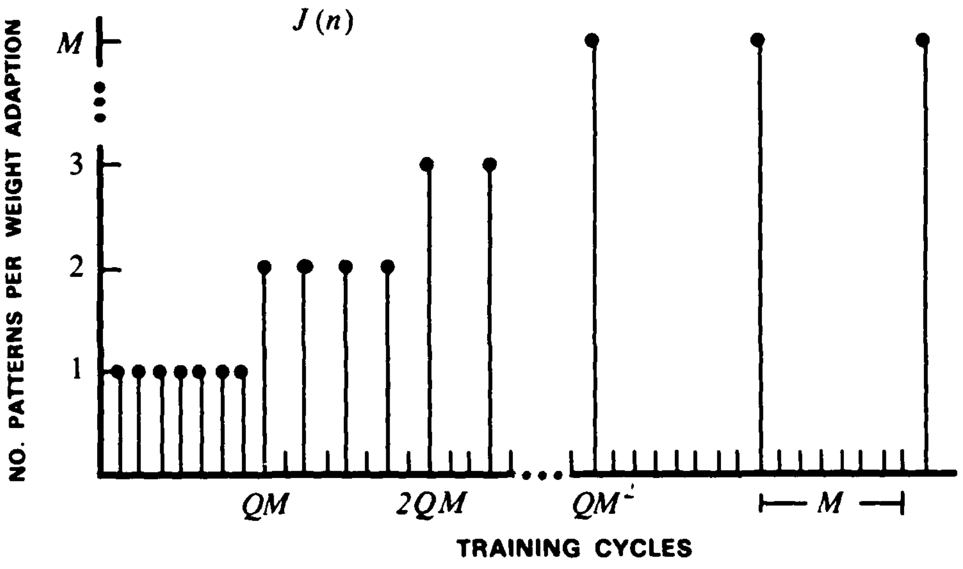
$$\Delta w_{ji}^{k}(n) = \eta \sum_{p=1}^{J(n)} \Delta_{pj}^{k} z_{pi}^{k-1} , \tag{29}$$

and the objective is to find the location where the objective function [sum squared error (4,5)] is minimum.

Now according to the steepest descent learning algorithm ($J(n) = M$), the weights are adapted following the presentation of *all* training patterns. Thus much caution is taken in choosing a trajectory, for the response of the system to the entire training set is considered before each step is taken.

Conversely, the Rumelhart back propagation algorithm [7] results in a more erratic search. Here the weights are adapted following the presentation of a *single* pattern in the training set ($J(n) = 1$). Intuitively, such an erratic trajectory is desired for initially searching a vast multidimensional space. However, the noisy trajectory can be prohibitive in the latter phases of the search, causing the new step to wander away from the near optimal region. In fact, such trajectory jitter is often countered with smoothing by placing a momentum term in the learning equation [7] (actually a single pole low pass filter $\Delta w_{ji}(n+1) = \alpha \Delta w_{ji}(n) + \eta(\Delta_{pj} z_{pi})$).

A compromise is postulated to gain the benefit of both extremes. At the beginning of the search, the weights are adapted after *each* training pattern is presented, while toward the end of the search, *all* training patterns are presented before the weights are adapted. The variation is gradual as depicted in Fig. 5.



103368-6

Fig. 5. Graded training schedule.

Such schedule requires M^2Q training cycles (M durations at QM cycles per duration), while the total number of weight adaptions follow a finite harmonic series

$$N^H_{\Delta w} = QM\left(\sum_{j=1}^{M} \frac{1}{j}\right). \tag{30}$$

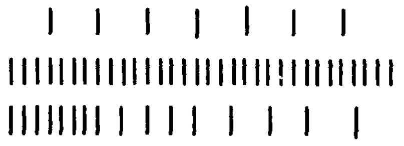
Q is chosen large enough to ensure satisfactory training. Typically Q is estimated by

$$Q = \bar{s} \;/\; M \tag{31}$$

where $\bar{s}$ is the average number of training sweeps required to learn a given mapping using the Rumelhart algorithm [7].

In consequence, three distinct learning algorithms emerge as special cases of the generalized weight adaption rule (29). These cases are distinguished by the manner in which the training patterns are presented per weight adaption according to

$$J(n) = \begin{cases} M & \dots\dots\dots \text{ steepest descent} \\ 1 & \dots\dots\dots\dots\dots \text{ Rumelhart} \\ \sum_{q=1}^{M} U\left(n - QM \sum_{r=1}^{q-1} r^{-1}\right) & \text{harmonic} \end{cases} \tag{32}$$



where

$$U(n) = \begin{cases} 1 & 0 \le n \\ 0 & n < 0 \end{cases}$$

is the unit step function.

3. RESULTS

The performance comparison via simulation consists of exercising the algorithms on a benchmark association problem. Given the training set, the learning process is initiated with a pseudorandom weight vector and is terminated when the outputs of the system are all within 10% of the target values

$$\left(\left|z^K_{pj} - y_{pj}\right| \le 0.1 \ \forall\, p, j\right) \tag{33}$$

denoting zero decision errors. In addition to the tabulation of the decision errors versus training sweep, the root-mean-squared error (RMS = $\sqrt{2E/ML}$) is also included. Furthermore, the relative computational burden of the algorithms is discussed, along with the sensitivity of the proposed algorithm to the additional a priori information required.

3.1 BENCHMARK ASSOCIATION PROBLEM

The classical benchmark problem for analyzing perceptron based architectures is the exclusive - OR (XOR) problem [3]. The XOR mapping

$$T = \left\{\left(\begin{pmatrix}0\\0\end{pmatrix}, 0\right), \left(\begin{pmatrix}0\\1\end{pmatrix}, 1\right), \left(\begin{pmatrix}1\\0\end{pmatrix}, 1\right), \left(\begin{pmatrix}1\\1\end{pmatrix}, 0\right)\right\} \tag{34}$$

involves second-order correlations and hence requires hidden units. The XOR performance comparison is conducted by executing each of the three algorithms supplied with identical pseudorandom weight vectors. Both RMS and decision errors are tabulated as a function of the training sweep. (A training sweep is defined as M training cycles.) Following 200 independent training sessions for each algorithm, statistics are gathered and the architecture is then modified by changing the number of hidden units. Results of the procedure shown in Fig. 6. include the average time required by the algorithms to learn the XOR mapping (expressed by the number of training sweeps) as a function of the number of hidden units. Also shown are the least-squares linear fit expressions [14] for each of the algorithms, yielding a linear variation in the training time versus the logarithm of the number of hidden unit processors. Such relationship reinforces the trade-off between training time and architecture size for each of the algorithms.

Notice the algorithms exhibit similar performance. The harmonic algorithm experiences slightly faster convergence for the larger architectures, while the Rumelhart algorithm is slightly superior at the smaller architectures, and the steepest descent algorithm possesses the slowest convergence.

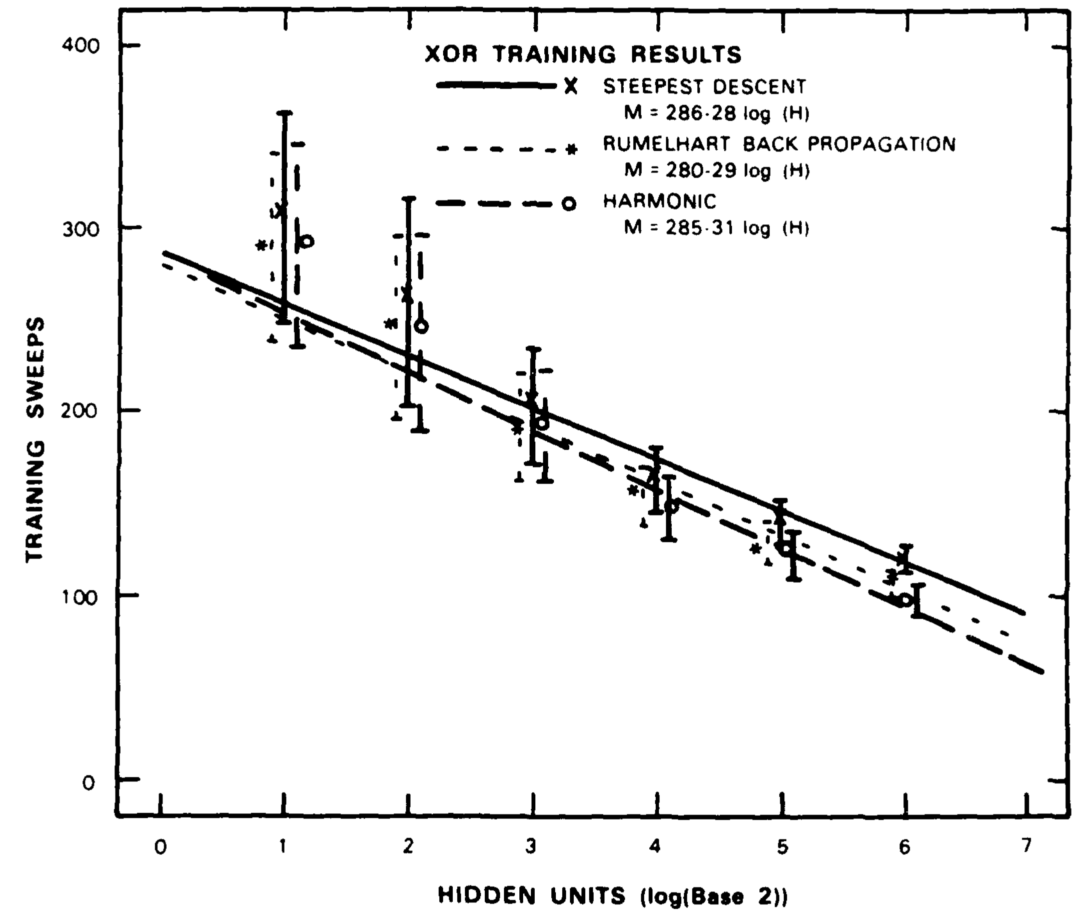


Fig. 6. Averaged results for XOR problem. Network parameters include $\eta = 0.25$, $\alpha = 0.90$. Error bars denote $\pm\ \sigma$.

3.2 COMPUTATION REQUIREMENT COMPARISON

Given an arbitrary association problem, analytically assessing the absolute computational requirements for successful learning using the learning algorithms discussed would be difficult. However, if the number of training cycles is maintained equivalent for each of the algorithms, a relative computation comparison is straightforward. And provided the algorithms yield similar performance on the given association task (as demonstrated in the above XOR problem), such comparison serves to accurately depict the differences in computational requirements.

Now since each algorithm employs the same circuitry to conduct a training cycle (see Fig.4.), the difference in computation lies primarily in the amount of weight adaptions required. Thus the quantity of interest is the relative amount of weight adaptions required by each of the learning algorithms, assuming an equal number of training cycles conducted.

Assuming QM^2 training cycles are required to learn a given mapping, then the number of weight adaptions required for the Rumelhart algorithm is $N^R_{\Delta w} = QM^2$ (since the weights are adapted following each training cycle), while the steepest descent algorithm requires $N^S_{\Delta w} = QM$ (since the weights are adapted following M training cycles). Finally, the number of net weight adaptions for the harmonic algorithm is

$$N^H_{\Delta w} = QM\left(\sum_{i=1}^{M} \frac{1}{i}\right)$$

$$= N^R_{\Delta w}\left(\frac{1}{M} \sum_{i=1}^{M} \frac{1}{i}\right) \tag{35}$$

and is seen to be a fraction of those required for the Rumelhart algorithm.The growth of the weight adaptions as a function of the number of training cycles is illustrated in Fig. 7.

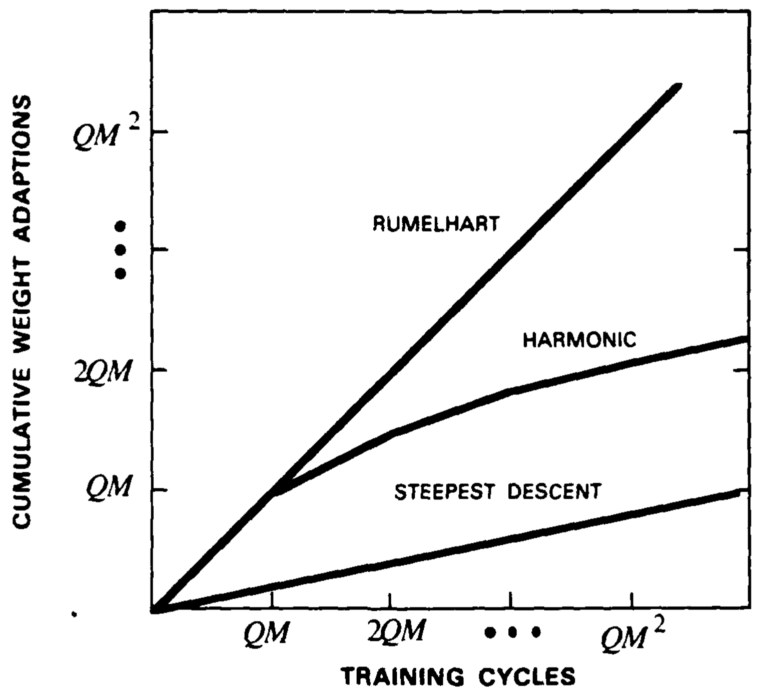


Fig. 7. Cumulative weight adaptions for the learning algorithms.

Notice the harmonic algorithm asymptotically reaches the rate of adaptions required by steepest descent, while the Rumelhart algorithm grows at a rate M times faster.

Therefore assuming equal number of training cycles for each algorithm, the harmonic algorithm requires significantly less weight adaptions than conventional back propagation (35), although not as few as the steepest descent algorithm. In fact for the XOR problem $\left(N^H_{\Delta w} = 52\%\ N^R_{\Delta w} = 208\%\ N^S_{\Delta w}\right)$ the harmonic algorithm requires about half the weight updates required for Rumelhart back propagation, although twice as many for steepest descent.

3.3 SENSITIVITY TO ADDITIONAL A PRIORI INFORMATION

The relative sensitivity of the harmonic algorithm with respect to additional a priori information is addressed. The additional information consists of specifying the training duration parameter Q.

As noted previously Q is typically estimated by running the Rumelhart algorithm repeatedly on the training set to obtain the average number of training sweeps. However, this procedure can be time consuming. Alternatively Q may be arbitrarily selected provided the performance remains satisfactory. The following sensitivity study illustrates the degradation incurred for arbitrary selections of Q for the benchmark association problem.

The study involves varying the value of Q about the nominal value ($Q = Q_o = \bar{s} / M$). Note the limiting cases ($Q = 0$, $Q = \infty$) represent the previously compared algorithms as illustrated in Fig. 8.

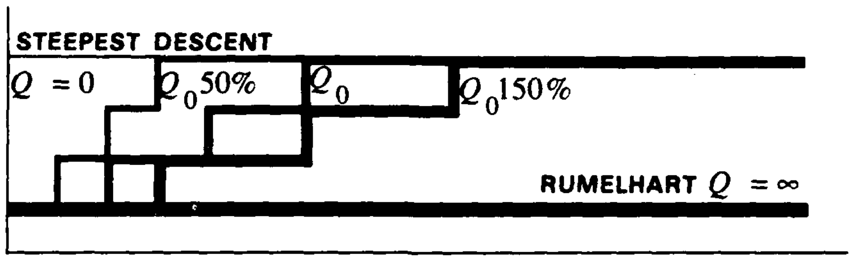
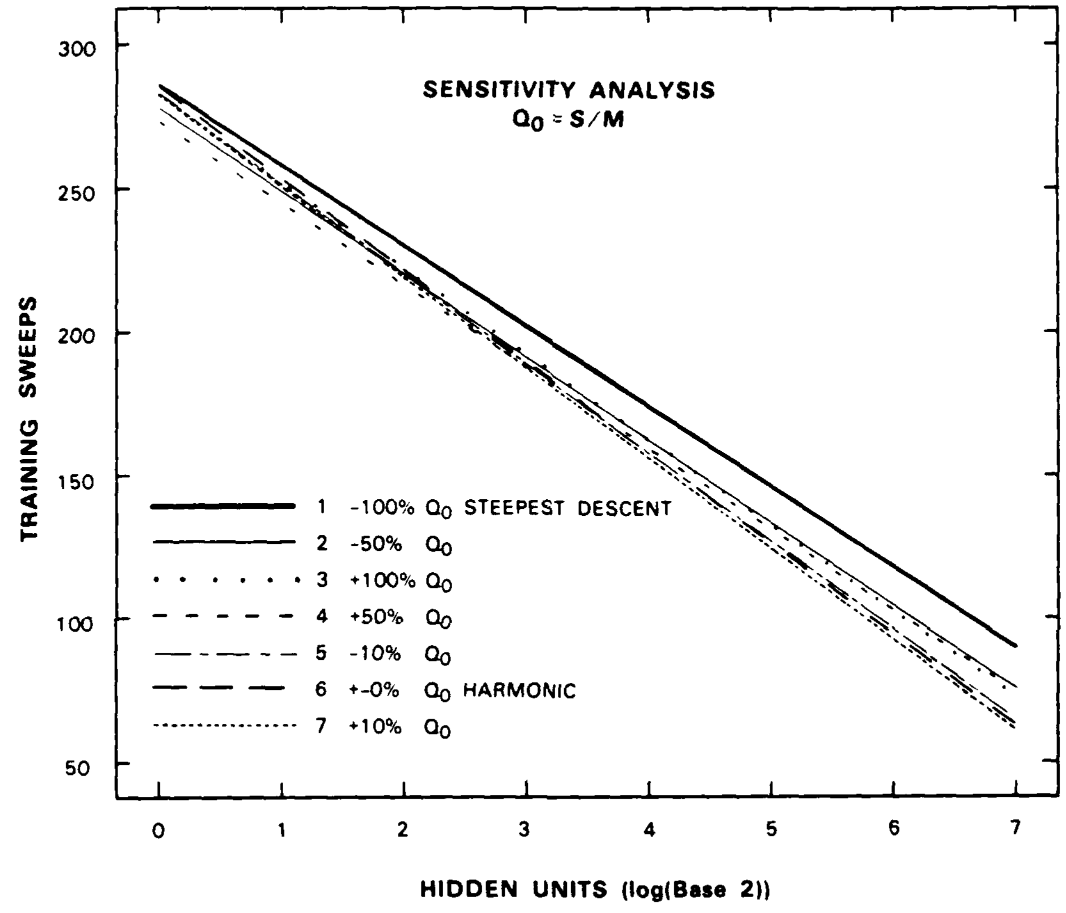


Fig. 8. Training schedules produced by varying duration parameter Q.

The least-squares linear approximations to the training performance data compiled with the various Q values for the same 200 independent sessions used in 3.1 are shown in Fig. 9. The sensitivity appears most pronounced for the excessively large architectures (37% at $\Delta Q = Q_o - 0$ at 2^7 hidden units) and almost negligible at the smaller architectures matched to the XOR problem complexity (3% at $\Delta Q = Q_o - 0$ at 2^2 hidden units). Notice the performance of the various Q cases are upper bounded by steepest descent, $Q = 0$. Thus for the problem tested, improper a priori estimation of Q results at worst in the training performance characteristic of the steepest descent algorithm.

103368-8



99350-2

Fig. 9. Training results for variations in training schedules.

4. CONCLUSION

In conclusion, a generalized learning algorithm was introduced for the multilayer perceptron which reduces to a host of special cases — steepest descent, Rumelhart back propagation, and the proposed harmonic algorithm — simply by altering the scheduling of the patterns presented during training. This general technique of modifying the scheduling of the training patterns is applicable to a variety of iterative minimization techniques, possibly resulting in favorable compromises as demonstrated here in the associative memory context.

Within such context, the training schedule for the proposed harmonic algorithm represents a specific compromise where the search for the optimal weight vector begins by adapting the weights following the presentation of *each* training pattern, while concluding by adapting the weights following the presentation of *all* training patterns.

The results indicate similar training performance amongst the three algorithms compared, although significant differences arise in the amount of weight adaptions required. The steepest descent algorithm requires the least adaptions, followed by the harmonic algorithm (weight adaption rate asymptotically approaches the favorable steepest descent rate), and finally the Rumelhart algorithm (weight adaption rate M times greater than steepest descent).

In all, the choice of the learning algorithm employed for a given association problem is dependent upon the training performance comparison for such problem and the priority given to the relative computational burden.

REFERENCES

1. M. Eggers, *Associative Memory: Biological and Mathematical Aspects*, Technical Report TR798, MIT Lincoln Laboratory, 1987.

2. F. Rosenblatt, *Principles of Neurodynamics*, Spartan Books, New York, 1959.

3. M. Minsky, S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, 1969.

4. A. N. Kolmogorov, On the representation of continuous functions of many variables by superpositions of functions of one variable and addition, *Dokl. Akad. Nauk*, Vol. 114, pp. 679-681, 1957.

5. G. G. Lorentz, The 13th problem of Hilbert, in *Proc. Symp. in Pure Math*, Vol. 28 part II, pp. 419-429, 1976.

6. D. A. Sprecher, On the structure of continuous functions of several variables, *Trans. Am. Math. Soc.*, Vol. 115, pp.340-355, 1964.

7. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning internal representations by error propagation, in *Parallel Distributed Processing*, Vol. 1, D. E. Rumelhart and J. L. McClelland, eds., MIT Press, 1986.

8. T. Sejnowski, C. R. Rosenburg, *NETtalk: A Parallel Network that Learns to Read Aloud*, Technical Report JHU/EECS-86/01, Johns Hopkins University, 1986.

9. D. J. Burr, A neural network digit recognizer, in *Proc. Int. Conf. on Systems, Man and Cybernetics*, IEEE, 1986.

10. R. P. Lippmann, An introduction to computing with neural nets, *IEEE ASSP Magazine* , Vol. 4., pp. 4-22, Apr. 1987.

11. B. Widrow, M. E. Hoff, Jr., Adaptive switching circuits, IRE WESCON Conv. Rec., pt. 4, pp. 96-104, 1960.

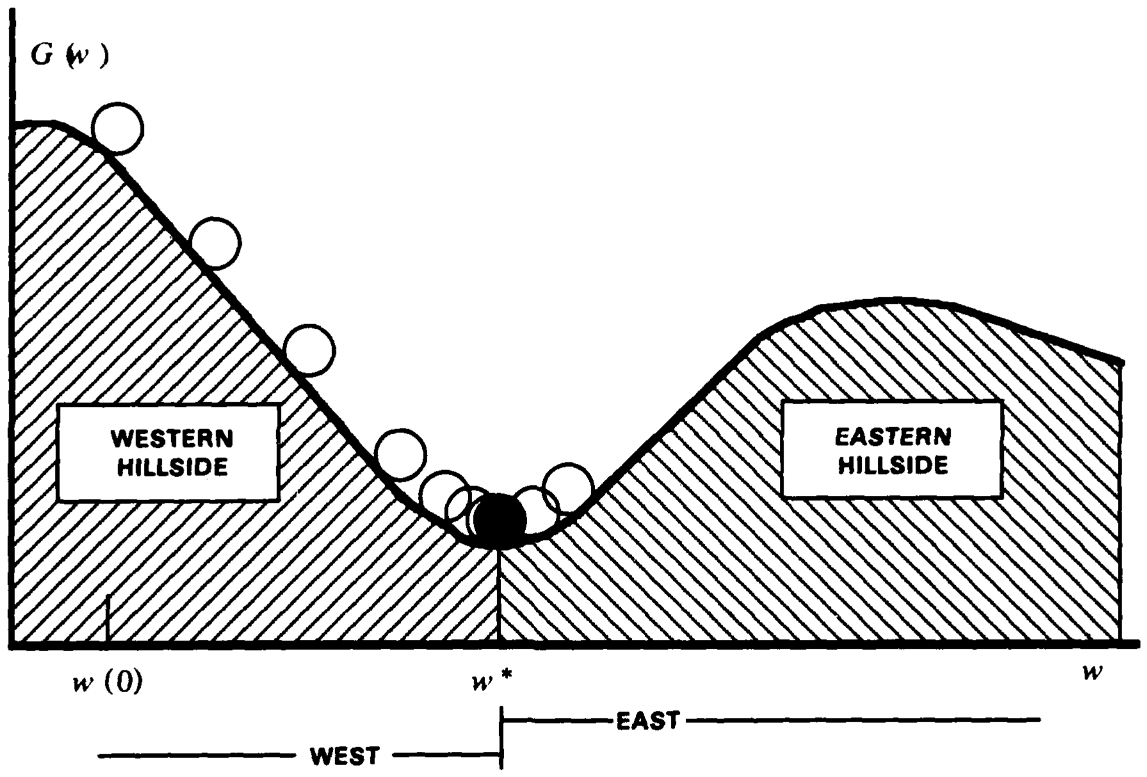
12. B. Widrow, P. E. Mantey, L. J. Griffiths, and B. B. Goode, Adaptive antenna systems, *Proc. of the IEEE*, Vol. 55, No. 12, pp. 2143-2159, Dec. 1967.

13. D. B. Parker, *Learning Logic*, Technical Report TR-47, MIT Center for Computational Research in Economics and Management Science, 1985.

14. P. R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences*, McGraw Hill Book Co., New York, 1969.

APPENDIX

The method of steepest descent can be intuitively formulated with the following example. Consider the single dimension case where $G(w)$ is to be minimized with respect to the independent variable w. The function G can be viewed as a landscape with hills and valleys, as shown in Fig. A-1.



103368-9

Fig. A-1. Landscape interpretation of minimization.

With such interpretation, the goal of the minimization process is to travel from an initial position upon a hillside to the final destination in the valley below. From the diagram, the traveler should travel east (increasing w) when on the western hillside in order to reach the valley. Notice the slope of the western hillside is negative while the eastern hillside is positive. Hence the traveler's plan can be stated mathematically as always moving in the direction opposite to the polarity of the derivative at the present location. Thus the traveler's rule guiding the next step can be formulated in terms of the present step according to

$$w(n+1) = w(n) + \Delta w(n) \tag{36}$$

where

$$\Delta w(n) = -\eta \ \operatorname{sign}\left(\frac{\partial G}{\partial w}\bigg|_{w(n)}\right) \tag{37}$$

and η represents the traveling rate (dependent upon the traveler's physical condition).

A refinement can be made by realizing the traveler should take large steps when high on the hillside far from the valley (large $|\partial G / \partial w|$), and conversely take small steps when nearing the valley (small $|\partial G / \partial w|$). Mathematically, this is equivalent to making the step size proportional to the magnitude of the derivative. Consequently, the refined rule is (36) with

$$\Delta w(n) = -\eta \frac{\partial G}{\partial w}\bigg|_{w(n)} \tag{38}$$

and this *minimization process* is diagrammed in Fig. A-1.

The final extension involves expanding the landscape in many directions. Here G is now a function of several independent variables, denoted by the vector $\overline{w}$. And the traveler simply moves in the direction of steepest descent yielding the following algorithm applied to each component,

$$w_l(n+1) = w_l(n) + \Delta w_l(n) \tag{39}$$

where

$$\Delta w_l(n) = -\eta \frac{\partial G(\overline{w})}{\partial w_l}\bigg|_{\overline{w}(n)}. \tag{40}$$

Disadvantages of the algorithm include the inability to escape from local minima as well as excessive computation times for large dimensions.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public releasee, distribution unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Report 813	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-TR-88-174

6a. NAME OF PERFORMING ORGANIZATION Lincoln Laboratory, MIT	6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION Electronic Systems Division
6c. ADDRESS *(City, State, and Zip Code)* P.O. Box 73 Lexington, MA 02173-0073		7b. ADDRESS *(City, State, and Zip Code)* Hanscom AFB, MA 01731
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U.S. Army Strategic Defense Command Huntsville Sensors Directorate	8b. OFFICE SYMBOL *(If applicable)* DASD-H-SBS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-85-C-0002

8c. ADDRESS *(City, State, and Zip Code)*
P.O Box 1500
Huntsville, AL 35807-3801

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
31310F	80		

11. TITLE *(Include Security Classification)*
Learning Algorithms for the Multilayer Perceptron

12. PERSONAL AUTHOR(S)
M.D. Eggers and T.S. Khuon

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Technical Report	FROM _____ TO _____	28 October 1988	38

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
multilayer perceptron neural network
neural network learning algorithms
machine learning
associative memory
neural networks

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Central to the development of adaptive pattern processing algorithms (adaptive filters) for random problems — problems where statistics are unknown a priori and/or explicit rules governing behavior cannot be extracted in a reductionist manner — is the pursuit of adaptive architectures for associating arbitrary inputs to outputs. Such "associative memories" are important for providing the mathematical mapping (transfer function) relating inputs to outputs arising from implicit relationships found in a given training ensemble. The adaption of these filters or architectures during training is guided by a learning algorithm, mathematically derived from an objective function to ensure good association properties.

The subject of this paper is an investigation of a class of learning algorithms for the highly parallel multilayer perceptron architecture used in an *associative memory* context. By controlling the scheduling of patterns presented during training, a generalized class of learning algorithms are shown to result. Specific realizations of the generalized algorithm include steepest descent (parameters adapted following presentation of *all* training patterns), Rumelhart back propagation (parameters adapted following presentation of *each* pattern), and a new algorithm which captures in part the benefits of both, less parameter adaption and faster convergence, by gradually varying the number of patterns presented per parameter adaption. A systematic derivation of the fundamental steepest descent algorithm for the multilayer perceptron is included for clarification, although related learning algorithms have been formulated by others.

Learning results are presented utilizing the algorithms on a simple benchmark association problem. Although performance is similar amongst the algorithms, the relative computational burden differs substantially. Of the algorithms investigated, steepest descent requires the least computation, while back propagation is the most demanding.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT.] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE *(Include Area Code)*	22c. OFFICE SYMBOL
Lt. Col. Hugh L. Southall, USAF	(617) 981-2330	ESD/TML