# Deterministic Nonmonotone Strategies for Effective Training of Multilayer Perceptrons

Vassilis P. Plagianakos, George D. Magoulas, *Member, IEEE*, and Michael N. Vrahatis

***Abstract*—In this paper, we present deterministic nonmonotone learning strategies for multilayer perceptrons (MLPs), i.e., deterministic training algorithms in which error function values are allowed to increase at some epochs. To this end, we argue that the current error function value must satisfy a nonmonotone criterion with respect to the maximum error function value of the $M$ previous epochs, and we propose a subprocedure to dynamically compute $M$. The nonmonotone strategy can be incorporated in any batch training algorithm and provides fast, stable, and reliable learning. Experimental results in different classes of problems show that this approach improves the convergence speed and success percentage of first-order training algorithms and alleviates the need for fine-tuning problem-depended heuristic parameters.**

***Index Terms*—Adaptive learning rate algorithms, backpropagation (BP) algorithm, multilayer perceptrons (MLPs), nonmonotone minimization, unconstrained minimization.**

## I. Introduction

THE BATCH training of a multilayer perceptron (MLP) is consistent with the theory of unconstrained optimization and can be viewed as the minimization of the function $E$; that is to finding a minimizer $w^* = (w_1^*, w_2^*, \ldots, w_n^*) \in \mathbb{R}^n$, such that

$$w^* = \min_{w \in \mathbb{R}^n} E(w) \tag{1}$$

where $E$ is the batch error measure defined as the sum-of-squared-differences error function (SSE) over the entire training set.

Next, we minimize $E$ by considering the family of gradient-based training algorithms having the iterative form

$$w^{k+1} = w^k + \eta^k \varphi^k, \qquad k = 0, 1, 2, \ldots \tag{2}$$

where $w^k$ is the current weight vector, $\varphi^k$ is a search direction and $\eta^k$ is the learning rate at the $k$th iteration. Various choices of the direction $\varphi^k$ give rise to distinct algorithms. A broad class of methods uses the search direction $\varphi^k = -\nabla E(w^k)$, where the gradient $\nabla E(w)$ can be obtained by means of backpropagation of the error through the layers of the network [62].

The most popular training algorithm of this class, which is named batch backpropagation (BP) [62], is a first-order method that minimizes the error function using the steepest descent method [19]

$$w^{k+1} = w^k - \eta \nabla E(w^k) \tag{3}$$

where $\eta$ is a heuristically chosen parameter, called *learning rate*. Appropriate learning rate values help avoid convergence to a saddle point or a maximum. In practice, a small constant learning rate is chosen ($0 < \eta < 1$) in order to secure the convergence of the BP training algorithm and avoid oscillations in a steep direction of the error surface. However, it is well known that this approach tends to be inefficient. For example, this happens when the search space contains long ravines that are characterized by sharp curvature across them and a gently slopping floor [28], [62]. Moreover, this approach introduces difficulties in obtaining convergence of BP training algorithms [33], [38]. Nevertheless, there are theoretical results that guarantee the convergence of batch BP algorithms for a constant learning rate. In this case, the learning rate should be proportional to the inverse of the Lipschitz constant which, in practice, is not easily available [2], [42], [69].

A variety of approaches adapted from numerical analysis have been applied, in an attempt to use second derivative related information to accelerate the learning process [6], [44], [46], [53], [68], [72]. However, second-order training algorithms are, in certain cases, computationally intensive for MLPs with several hundred weights [7]. Furthermore, it is not certain that the extra computational cost speeds up the minimization process for nonconvex functions when far from a minimizer [12], [50], as is usually the case with the neural-network training problem [6]. Therefore, the development of improved gradient-based training algorithms is a subject of considerable ongoing research and receives significant attention of neural-network practitioners.

In this paper, we present deterministic gradient-based training algorithms in which error function values are allowed to increase at some epochs. In this way, the learning process exhibits nonmonotone behavior [14], however, the training procedure is fast, stable, and reliable.

This paper is organized as follows. In Section II, the class of first-order BP algorithms with adaptive learning rate is presented. Monotone learning strategies are briefly described in Section III. In Section IV, the notion of nonmonotone learning is introduced and new nonmonotone training algorithms are presented. Experimental results are reported in Section V to evaluate the performance of the nonmonotone algorithms and compare them with several other methods. Section VI presents the conclusions.

Manuscript received March 27, 2000; revised March 28, 2002.

V. P. Plagianakos and M. N. Vrahatis are with the Department of Mathematics and UP Artificial Intelligence Research Center, University of Patras, Patras, GR-261.10, Greece (e-mail: vpp@math.upatras.gr; vrahatis@math.upatras.gr).

G. D. Magoulas is with the Department of Information Systems and Computing, Brunel University, West London UB8 3PH, U.K. (e-mail: George.Magoulas@brunel.ac.uk).

Digital Object Identifier 10.1109/TNN.2002.804225

## II. Adaptive Learning Rate Algorithms

Research on MLP training usually focuses on BP algorithms with adaptive learning rate in order to accelerate the training procedure. The following strategies are usually suggested.

1) Start with a small learning rate and increase it exponentially, if successive epochs reduce the error, or rapidly decrease it, if a significant error increase occurs [5], [71].
2) Start with a small learning rate and increase it, if successive epochs keep gradient direction fairly constant, or rapidly decrease it, if the direction of the gradient varies greatly at each epoch [11].
3) For each weight, an individual learning rate is given, which increases if the successive changes in the weights are in the same direction and decreases otherwise [28], [54], [60], [63].
4) Use a closed formula to calculate a common learning rate for all the weights at each iteration [27], [42], [56] or a different learning rate for each weight [15], [43].

Note that all the above-mentioned strategies employ heuristic parameters in an attempt to enforce the decrease of the learning error at each iteration and secure the converge of the training algorithm.

A different approach is based on Goldstein's and Armijo's work on steepest-descent and gradient methods. The method of Goldstein [20] requires the assumption that $E$ is twice continuously differentiable on $\mathcal{S}(w^0)$, where $\mathcal{S}(w^0) = \{w\colon E(w) \leqslant E(w^0)\}$ is bounded, for some initial vector $w^0$. It also requires that $\eta$ is chosen to satisfy the relation $\sup ||H(w)|| \leqslant \eta^{-1} < \infty$, where $H(w)$ denotes the Hessian of $E$ at $w$, in some bounded region, where the relation $E(w) \leqslant E(w^0)$ holds. However, the manipulation of the full Hessian is too expensive in computation and storage for MLPs with several hundred weights [7]. In [36], a technique based on appropriate perturbations of the weights has been proposed for the online estimation of the extreme eigenvalues and eigenvectors of the Hessian without calculating the full matrix $H$. According to experiments reported in [36], the largest eigenvalue of the Hessian is mainly determined by the MLP architecture, the initial weights and by short-term low-order statistics of the training data. This technique can be used to determine $\eta$ requiring additional presentations of the training set in the early training.

Cauchy's method [2], [10], suggests that the value of the learning rate $\eta$ is related to the value of the Lipschitz constant $L$, which depends on the morphology of the error surface. In this case, the BP algorithm takes the form

$$w^{k+1} = w^k - \frac{1}{2L}\nabla E(w^k) \tag{4}$$

and converges to the point $w^*$ which minimizes $E$ (see [2] for conditions under which convergence occurs and a convergence proof). In [42], a local estimation of the Lipschitz constant has been proposed in a learning rate adaptation strategy, which provides increased rate of convergence, and guarantees the stability of the learning process.

## III. Monotone Learning Strategies

A training algorithm can be made globally convergent by determining the learning rate in such a way that the error is exactly minimized along the current search direction at each epoch, i.e., $E(w^{k+1}) < E(w^k)$. To this end, an iterative search, which is often expensive in terms of error function evaluations, is required. It must be noted that the above simple condition does not guarantee global convergence for general functions, i.e., converges to a local minimizer from any initial condition (see [13] for a general discussion of globally convergent methods).

In adaptive learning rate algorithms, monotone reduction of the error function at each iteration can be achieved by searching a local minimum with small weight steps. These steps are usually constrained by problem-dependent heuristic learning parameters. The use of heuristic strategies enforces the monotone decrease of the learning error and secures the convergence of the training algorithm to a minimizer of $E$. However, the use of inappropriate values for the heuristic learning parameters can considerably slow down the rate of training or even lead to divergence and to premature saturation [37], [61]; there is a tradeoff between convergence speed and stability of the training algorithm. Additionally, the use of heuristics for bounding the learning rate prevents the development of efficient algorithms that will converge to a local minimizer of the error function starting from any initial weight vector, i.e., globally convergent algorithms [13].

A monotone learning strategy, which does not apply heuristics to bound the length of the minimization step, consists in accepting a positive learning rate $\eta^k$ along the search direction $\varphi^k \neq 0$, if it satisfies the *Wolfe conditions*

$$E(w^k + \eta^k\varphi^k) - E(w^k) \leqslant \sigma_1\eta^k\langle\nabla E(w^k),\, \varphi^k\rangle \tag{5}$$

$$\langle\nabla E(w^k + \eta^k\varphi^k),\, \varphi^k\rangle \geqslant \sigma_2\langle\nabla E(w^k),\, \varphi^k\rangle \tag{6}$$

where $0 < \sigma_1 < \sigma_2 < 1$ and $\langle\cdot,\,\cdot\rangle$ stands for the usual inner product in $\mathbb{R}^n$. The first inequality ensures that the error is reduced sufficiently, and the second prevents the learning rate from being too small. It can be shown that if $\varphi^k$ is a descent direction and $E$ is continuously differentiable and bounded below along the ray $\{w^k + \eta\varphi^k \,|\, \eta > 0\}$, then there always exists a learning rate satisfying (5) and (6) [13], [50]. Relation (6) can be replaced by

$$E(w^k + \eta^k\varphi^k) - E(w^k) \geqslant \sigma_2\eta^k\langle\nabla E(w^k),\, \varphi^k\rangle \tag{7}$$

where $\sigma_2 \in (\sigma_1,\, 1)$ (see [13]). The strategy based on Wolfe's conditions provides an efficient and effective way to ensure that the error function is globally reduced sufficiently. In practice, conditions (6) or (7) are generally not needed because the use of a backtracking strategy avoids very small learning rates [43], [70].

An alternative strategy has been proposed in [57]. It is applicable to any descent direction $\varphi^k$ and uses two parameters $\alpha,\, \beta \in (0,\, 1)$. Following this approach, the learning rate is $\eta^k = \beta^{m_k}$, where $m_k \in \mathbb{Z}$ is any integer that satisfies the conditions

$$E(w^k + \beta^{m_k}\varphi^k) - E(w^k) \leqslant \beta^{m_k}\alpha\langle\nabla E(w^k),\, \varphi^k\rangle \tag{8}$$

$$E(w^k + \beta^{m_k-1}\varphi^k) - E(w^k) > \beta^{m_k-1}\alpha\langle\nabla E(w^k),\, \varphi^k\rangle. \tag{9}$$

To ensure global convergence, monotone strategies that employ (5) and (6) or (8) and (9) must be combined with learning

rate tuning subprocedures. For example, a simple subprocedure for tuning the length of the minimization step is to decrease the learning rate by a reduction factor $1/q$, where $q > 1$ [52], so that it satisfies conditions (5) and (6) at each epoch. This *backtracking strategy* has the effect that the learning rate is decreased by the largest number in the sequence $\{q^{-m}\}_{m=1}^{\infty}$, so that (5) is satisfied. When seeking to satisfy (5) it is important to ensure that the learning rate is not reduced unnecessarily so that (6) is not satisfied. Since during training, the gradient vector is known only at the beginning of the iterative search for a new weight vector, condition (6) cannot be checked directly (this task requires additional gradient evaluations at each epoch), but is enforced simply by placing a lower bound on the acceptable values of the learning rate. This bound on the learning rate has the same theoretical effect as condition (6) and ensures global convergence [13].

## IV. Nonmonotone Learning Strategies

Although monotone learning strategies provide an efficient and effective way to ensure that the error function is reduced sufficiently, they have the disadvantage that no information, which might accelerate convergence, is stored and used [16]. To alleviate this situation we propose a nonmonotone learning strategy that exploits the accumulated information with regard to the $M$ most recent values of the error function. The following condition is used to formulate the new approach and to define a criterion of acceptance of any weight iterate:

$$E(w^k + \eta^k \varphi^k) - \max_{0 \leqslant j \leqslant M} \{E(w^{k-j})\} \leqslant \gamma \eta^k \langle \nabla E(w^k), \varphi^k \rangle \tag{10}$$

where $M$ is a nonnegative integer, named *nonmonotone learning horizon*, $\gamma$ takes values in the range $(0, 1)$, $\eta^k$ indicates the learning rate, and $\varphi^k$ is the search direction at the $k$th epoch. The above condition allows for an increase in the function values, which is regulated by the value of $\gamma$, without affecting the global convergence properties, as it has been proved theoretically in [21] and [59]. In practice, a value of $0 < \gamma \ll 1$ is suggested [21]. Thus, in our experiments, we have used a fixed value $\gamma = 10^{-5}$ for all test problems, in order to test the robustness of our methods in different types of problems.

Furthermore, it can be shown that the nonmonotone learning strategy generates a globally convergent sequence for any algorithm that follows search direction $\varphi^k \neq 0$, provided that two positive numbers $c_1$, $c_2$ exist, such that

$$\nabla E(w^k)^\top \varphi^k \leqslant -c_1 \|\nabla E(w^k)\| \tag{11}$$

$$\|\varphi^k\| \leqslant c_2 \|\nabla E(w^k)\|. \tag{12}$$

This follows directly from the convergence theorem in [21].

Next, we summarize the basic steps of the nonmonotone learning strategy at the $k$th iteration.

1) Update the weights $w^{k+1} = w^k + \eta^k \varphi^k$.
2) If $E(w^k + \eta^k \varphi^k) - \max_{0 \leqslant j \leqslant M}\{E(w^{k-j})\} \leqslant \gamma \eta^k \langle \nabla E(w^k), \varphi^k \rangle$, store $w^{k+1}$, set $k = k + 1$ and go to Step 1); otherwise go to Step 3).
3) Use a tuning technique for $\eta^k$ and return to Step 2).

A simple technique to tune $\eta^k$ at Step 3) is to decrease the learning rate by a reduction factor $1/q$, where $q > 1$, as mentioned in the previous section. Here, we remark that the selection of $q$ is not critical for successful learning; however, it has an influence on the number of error function evaluations required to obtain an acceptable weight vector. Thus, some training problems respond well to one or two reductions in the learning rate by modest amounts (such as 1/2), while others require many such reductions, but might respond well to a more aggressive learning rate reduction (for example, by factors of 1/10, or even 1/20). On the other hand, reducing $\eta^k$ too much can be costly since the total number of epochs will be increased. The value $q = 2$ is usually suggested in the literature [2] and, indeed, it was found to work effectively and efficiently in the experiments reported in this paper. The above procedure constitutes an efficient method of determining an appropriate learning rate without additional gradient evaluations. As a consequence, the number of gradient evaluations is, in general, less than the number of error function evaluations.

The nonmonotone learning strategy can be incorporated in any batch training algorithm. It can be used as a subprocedure that secures and accelerates the convergence of a batch training algorithm by providing the ability to handle large learning rates, and, in this way, learning by neural networks becomes feasible on a first-time basis for a given problem. Additionally, it alleviates problems generated by poor selection of the user-defined learning parameters, such as decreased rate of convergence, or even divergence and premature saturation [37].

### A. Nonmonotone Learning Horizon

Experimental results, exhibited in [55] and [56], indicate that the choice of the parameter $M$ is critical for the implementation and depends on the nature of the problem. Below, we propose a procedure to finding estimates of the nonmonotone learning horizon $M$ at each iteration $k$ that utilizes the notion of the Lipschitz constant.

It is well known that the Lipschitz constant is related to the morphology of a function [2], [10]. For example, the Lipschitz constant for a function having steep regions is large. On the other hand, when the function is flat the Lipschitz constant is small. However, in neural-network training practice, neither the morphology of the error surface nor the value of the Lipschitz constant are known *a priori*. In order to alleviate this situation in [42] a local estimation of the Lipschitz constant has been proposed, which provides information related to the local shape of the error function (see also [69] for the usefulness of this estimate).

The following procedure provides an elegant way to adapt the value of the nonmonotone learning horizon dynamically at the $k$th iteration:

$$M^k = \begin{cases} M^{k-1} + 1, & \Lambda^k < \Lambda^{k-1} < \Lambda^{k-2} \\ M^{k-1} - 1, & \Lambda^k > \Lambda^{k-1} > \Lambda^{k-2} \\ M^{k-1}, & \text{otherwise} \end{cases} \tag{13}$$

where $\Lambda^k$ is the local estimation of the Lipschitz constant at the $k$th iteration [42]

$$\Lambda^k = \frac{||\nabla E(w^k) - \nabla E(w^{k-1})||}{||w^k - w^{k-1}||} \tag{14}$$

which can be obtained without additional error function or gradient evaluations. If $\Lambda^k$ constantly increases during two consecutive epochs, the sequence of the weight vectors approaches a steep region and the value of $M^k$ should be decreased to avoid overshooting a possible minimum point. On the other hand, when $\Lambda^k$ constantly decreases at two consecutive epochs, the method possibly enters a valley in the weight space, and the value of $M^k$ should be increased. This allows the method to accept larger learning rates and move faster out of the flat region. Finally, when the value of $\Lambda^k$ has a rather random behavior (increasing or decreasing for only one epoch), the value of $M^k$ remains unchanged.

It is evident that $M^k$ has to be positive. Thus, if relation (13) gives a nonpositive value in $M^k$, the nonmonotone learning horizon is set equal to $M^k = 1$ in order to ensure that the error function is reduced sufficiently at the current iteration.

### *B. Developing Nonmonotone Training Algorithms*

Next, we describe a nonmonotone version of the BP with momentum [28] and [62] (BPM), as well as two nonmonotone adaptive learning rate training algorithms.

*1) Nonmonotone BPM:* In [28], [62] a simple, heuristic, strategy for accelerating the BP algorithm has been proposed, which is based on the use of a momentum term. The momentum term can been incorporated in the steepest descent method as follows:

$$w^{k+1} = w^k - (1-m)\eta\nabla E(w^k) + m(w^k - w^{k-1}) \tag{15}$$

where $m$ is the momentum constant. One drawback with the above scheme is that, if $m$ is set to a comparatively large value, gradient information from previous epochs is more influential than the current gradient information in updating the weights. One solution is to increase the learning rate, however, in practice, this approach frequently proves ineffective and leads to instability or saturation. Thus, if $m$ is increased, it may be necessary to make a compensatory reduction in $\eta$ to maintain network stability. In the experiment reported in the next section, we alleviate this problem by combining the BPM with the nonmonotone learning strategy. This nonmonotone BPM method is named NMBPM.

*2) Nonmonotone BP With Variable Stepsize (BPVS):* The BPVS [42] exploits the local shape of the error surface by estimating the Lipschitz constant at each epoch, and setting the learning rate $\eta^k$ according to the following formula:

$$\eta^k = \frac{1}{2\Lambda^k} = \frac{||w^k - w^{k-1}||}{2||\nabla E(w^k) - \nabla E(w^{k-1})||} \tag{16}$$

where $\Lambda^k$ is the local estimation of the Lipschitz constant $L$ at the $k$th epoch. Thus, when the error surface has steep regions, $\Lambda^k$ is large, and a small value for the learning rate is appropriate in order to guarantee convergence. On the other hand, when the error surface has flat regions, $\Lambda^k$ is small and a large learning rate is appropriate to accelerate the convergence. Thus, BPVS updates the weights using (3) where the learning rate is calculated using (16).

In order to eliminate the possibility of using an unsuitable local estimation of the Lipschitz constant, we combine the BPVS method with the nonmonotone learning strategy. This version of the BPVS, which provides nonmonotone training, is named NMBPVS.

*3) Nonmonotone Barzilai–Borwein Backpropagation (BBP):* In a previous work [55], we have proposed a neural network training algorithm called BBP, which is based on the Barzilai and Borwein method [4].

In BBP, the search direction is always the negative gradient direction and the learning rate is updated using the following formula:

$$\eta^k = \frac{\langle \delta^{k-1},\, \delta^{k-1} \rangle}{\langle \delta^{k-1},\, \psi^{k-1} \rangle} \tag{17}$$

where $\delta^{k-1} = w^k - w^{k-1}$ and $\psi^{k-1} = \nabla E(w^k) - \nabla E(w^{k-1})$. The motivation for this choice is that it provides a two-point approximation to the secant equation underlying quasi-Newton methods [57]. The key features of this method are the low storage requirements and the inexpensive computations. Moreover, it does *not* guarantee descent in the error function $E$. Our experiments (see also [56]) show that this property is valuable in neural-network training because, very often, the method escapes from local minima and flat valleys, whereas other methods are trapped. To secure the convergence of the method, even when the above formula gives unsuitable learning rates, we apply the nonmonotone learning strategy. We call this modified training algorithm NMBBP. The new method retains the ability of BBP to escape from undesirable regions in the weight space, as shown in Fig. 1. In the upper part of Fig. 1, the NMBBP convergence behavior in the eXclusive-OR (XOR) problem is presented, while in the lower part the learning rate behavior is shown. The method successfully escapes from a region with almost constant error function value $E \simeq 0.5$, using large positive as well as negative learning rates. Note that negative learning rates are possible due to the learning rate update formula (17).

### *C. Adaptive Learning Rate Algorithm Model With Nonmonotone Strategy*

A high-level description of a generic first-order algorithm that incorporates the nonmonotone strategy and utilizes a user-defined nonmonotone learning horizon is outlined below.

*Initialization:* Randomly initialize the weight vector $w^0$, and set the maximum number of epochs (ME), the running index $k = 0$, the user-defined value of the error goal $\varepsilon$, $\gamma \in (0, 1)$, the user-defined integer value of $M \in [1, ME]$, and the learning rate to an arbitrary positive value $\eta^0$.

1) For $k = 0$, compute $w^1 = w^0 - \eta^0 \nabla E(w^0)$. Repetition: For $k = 1, 2, \ldots, ME$.
2) Compute $\eta^k$ using the learning rate update formula (16) or (17).
3) Update the weight vector $w^{k+1}$ according to the relation

$$w^{k+1} = w^k - \eta^k \nabla E(w^k).$$

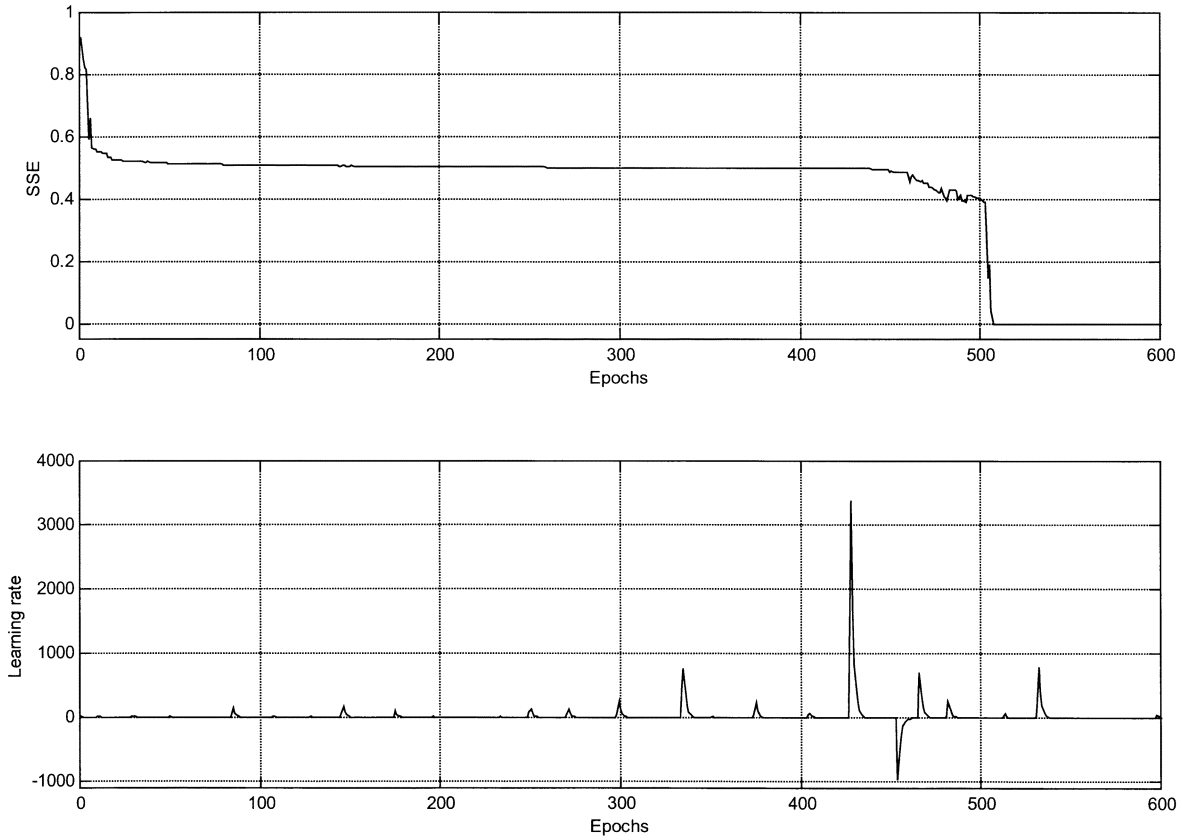


Fig. 1. XOR problem: (a) NMBBP convergence behavior. (b) Adaptive learning rate behavior.

4) If $M > k$ then set $M' = k$, else set $M' = M$.
5) Check the nonmonotone strategy

$$E(w^k - \eta^k \nabla E(w^k)) - \max_{0 \leqslant j \leqslant M'} \{E(w^{k-j})\} \leqslant -\gamma \eta^k \|\nabla E(w^k)\|^2.$$

If this is fulfilled go to Step 7).
6) Set $= eta^k = \eta^k/2$ and go to Step 5.
7) If the convergence criterion $E(w^{k+1}) \leqslant \varepsilon$ is met then terminate.
8) If $k < ME$, set $k = k + 1$ and go to Step 2); otherwise terminate.

*Termination:* Get the final weight vector $w^{k+1}$ and the corresponding error value $E(w^{k+1})$.

Instead of using a user-defined value for the nonmonotone learning horizon $M$, the adaptive procedure of relation (13) can be applied to dynamically evaluate $M^k$ at each iteration.

## V. Experimental Results

In this section, we evaluate the performance of the nonmonotone learning strategy by performing four sets of experiments: 1) we compare the algorithms NMBPVS and NMBBP with some well-known and widely used training algorithms; 2) we study the performance of the NMBPVS and NMBBP algorithms using various values of $M$, as well as an adaptive $M$; 3) we compare the performance of the training algorithms with and without the use of the nonmonotone strategy; and 4) we evaluate the generalization capability of the nonmonotone algorithms in five test problems. In all cases, average performance presented below was validated using the well-known test for statistical hypotheses, named $t$-test at the significance level $\alpha \leqslant 0.05$ (see, for example, [34]), using the SPSS 10 statistical software package.

### A. Comparative Study

Below, we evaluate the performance of the NMBPVS and NMBBP algorithms, and compare them with the batch versions of BP [62], BPM [28], and adaptive BP (ABP) [71]. The BPM applies the weight update rule of (15). A widely used value for the momentum term is $m = 0.9$ and for the learning rate is $\eta = 0.1$, as discussed in [23], [39]. The ABP applies the following learning rate update formula:

$$\eta^k = \begin{cases} \varphi \eta^{k-1}, & E(w^k) < E(w^{k-1}) \\ \beta \eta^{k-1}, & E(w^k) > \rho E(w^{k-1}) \\ \eta^{k-1}, & \text{otherwise} \end{cases}$$

where $\rho$ is the maximum error ratio in two successive iterations, $\varphi > 1$ and $\beta < 1$ are positive constants, named the learning rate increment and decrement factor, respectively. In our experiments we have used the values $\rho = 1.04$, $\varphi = 1.05$ and $\beta = 0.7$ as suggested in [71].

We have also performed experiments using the globally convergent modifications of the conjugate gradient methods: Fletcher–Reeves (FR) [18]; Polak–Ribiere (PR) [18]; Polak–Ribiere (PR) constrained by the FR method (PR–FR) [18].

For the NMBPVS and NMBBP algorithms, we decided to use the same values for $M$ ($M = 10$) and $\gamma$ ($\gamma = 10^{-5}$) for all experiments in order to verify that the nonmonotone algorithms

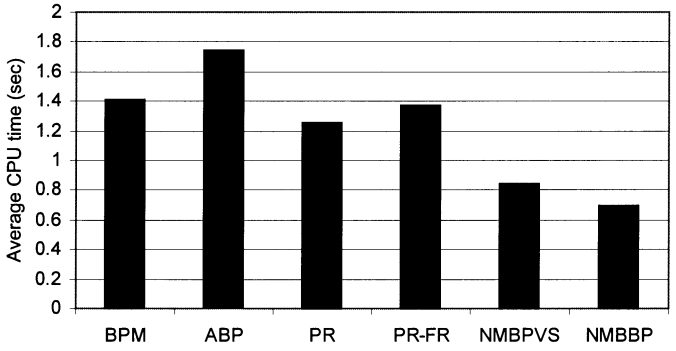


Fig. 2. Three-bit parity problem: Average CPU time for convergence of each training algorithm.

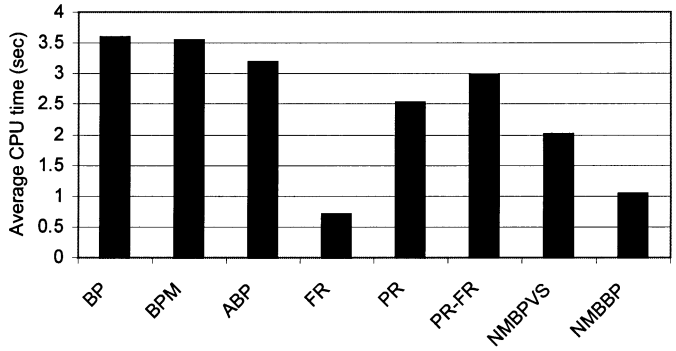


Fig. 3. Function approximation problem: Average CPU time for convergence of each training algorithm.

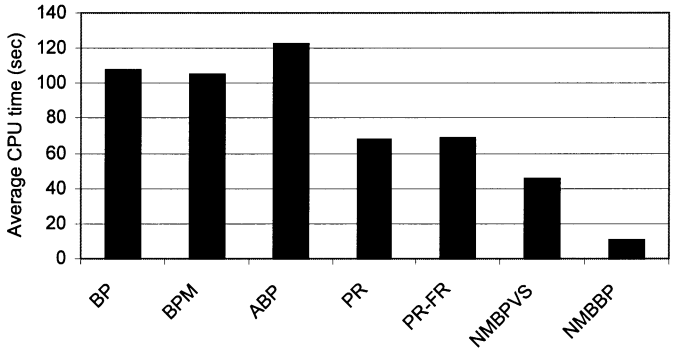


Fig. 4. Alphabetic font learning problem: Average CPU time for convergence of each training algorithm.

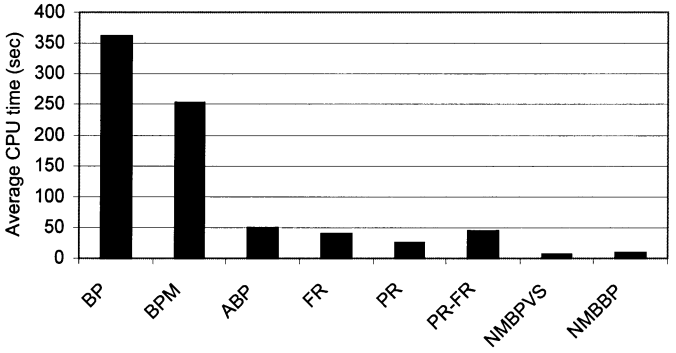


Fig. 5. Numeric font learning problem: Average CPU time for convergence of each training algorithm.

perform reasonably well to different types of problems without parameter fine tuning. The same sequence of input patterns has been presented to the MLPs and the weights are updated only after the entire set of the learning patterns has been presented (batch training).

Typically, initial weights and biases are initialized to small random values. Alternatively, techniques for finding the "best" available (in terms of convergence behavior) initial weight distribution range depending on the data set used, the network architecture and the activation functions can be used (see [39] for a review on this methods). Unless otherwise noted, the Nguyen–Widrow method [39], [49] was used in our experiments. This technique helps to prevent premature saturation at the hidden neurons by calculating the interval from which weights connecting input-hidden neurons are taken in accordance with the number of input neurons ($\mathcal{N}$) and the number of hidden neurons ($\mathcal{M}$). First, the parameter $\rho$ is calculated

$$\rho = 0.7(\mathcal{M}^{1/\mathcal{N}}) \tag{18}$$

and then the methods proceeds to choose a set of weights $w^r = (w^r_{11}, \ldots, w^r_{nm}, \ldots, w^r_{\mathcal{N}\mathcal{M}})$ randomly from the interval $(-1, +1)$. Finally, weights from input to hidden neurons are initialized using the relation

$$w^0_{nm} = \frac{\rho w^r_{nm}}{\|w^r\|}. \tag{19}$$

This process of weight initialization results in distributing the initial weights at the hidden layer in such a way that it is more likely that each input pattern will cause a hidden neuron to learn efficiently, accelerating convergence and preventing premature saturation [49]. In our experiments we applied the Nguyen–Widrow method. One thousand independent trials have been performed by taking a different set of weights $w^r$ for each trial. In order to have a fair comparison of the training algorithms, we have used exactly the same initial weights for all algorithms tested.

For each of the four test problems described below, we present a table summarizing the performance of the algorithms for simulations that reached solution within a predetermined *limit* of error function evaluations (specified below). The reported parameters are: $\mathrm{Min}$ the minimum number of epochs, $\mu$ the mean value of epochs, $\mathrm{Max}$ the maximum number of epochs, $\sigma$ the standard deviation, and *Succ.* the simulations succeeded out of 1000 trials within the error function evaluations limit. If an algorithm fails to converge within the error function evaluation limit, it is considered that it fails to train the MLP but its epochs are not included in the experimental analysis of the algorithm. Figs. 2–5 exhibit the comparative CPU times of all the algorithms tested.

We must also note that for the BP, BPM, and ABP each epoch corresponds to one gradient and one error function evaluation. On the other hand, the number of error function evaluations (FEs) of the conjugate gradient methods (PR, FR, and PR–FR) is, in general, larger than the number of gradient evaluations (GEs). These methods require one GE at each epoch but more than one FEs, due to the use of the inexact line search. NMBPVS and NMBBP also require additional FEs at each epoch as they use the nonmonotone strategy. As a consequence,

TABLE I
COMPARATIVE RESULTS FOR THE THREE-BIT PARITY PROBLEM

| Algorithm | | $Min$ | $\mu$ | $Max$ | $\sigma$ | $Success$ |
|---|---|---|---|---|---|---|
| BP | | * | * | * | * | * |
| BPM | | 246 | 485.4 | 973 | 195.4 | 48.0% |
| ABP | | 465 | 599.2 | 924 | 103.9 | 45.0% |
| FR | | * | * | * | * | * |
| PR | (FE) | 189 | 465.3 | 972 | 188.2 | 53.2% |
| | (GE) | 148 | 399.7 | 836 | 179.2 | |
| PR–FR | (FE) | 201 | 520.9 | 943 | 196.8 | 56.2% |
| | (GE) | 170 | 432.3 | 829 | 187.2 | |
| NMBPVS | (FE) | 103 | 292.1 | 986 | 161.9 | 72.7% |
| | (GE) | 98 | 285.5 | 960 | 156.8 | |
| NMBBP | (FE) | 47 | 298.9 | 978 | 212.4 | 67.9% |
| | (GE) | 37 | 181.9 | 601 | 118.9 | |

* the algorithm failed to converge within the error function evaluations limit

TABLE II
COMPARATIVE RESULTS FOR THE FUNCTION APPROXIMATION PROBLEM

| Algorithm | | $Min$ | $\mu$ | $Max$ | $\sigma$ | $Success$ |
|---|---|---|---|---|---|---|
| BP | | 328 | 706.7 | 998 | 175.6 | 13.8% |
| BPM | | 332 | 699.2 | 993 | 174.8 | 13.7% |
| ABP | | 166 | 628.1 | 994 | 216.8 | 26.9% |
| FR | (FE) | 44 | 173.2 | 474 | 133.3 | 17.3% |
| | (GE) | 38 | 87.8 | 164 | 43.4 | |
| PR | (FE) | 151 | 532.7 | 954 | 232.4 | 41.1% |
| | (GE) | 119 | 465.5 | 940 | 235.1 | |
| PR–FR | (FE) | 80 | 597.5 | 997 | 283.3 | 43.0% |
| | (GE) | 69 | 574.1 | 982 | 278.5 | |
| NMBPVS | (FE) | 64 | 443.9 | 991 | 238.5 | 66.8% |
| | (GE) | 64 | 429.2 | 960 | 228.9 | |
| NMBBP | (FE) | 29 | 241.7 | 988 | 195.1 | 92.2% |
| | (GE) | 26 | 158.2 | 625 | 114.7 | |

even when NMBPVS and NMBBP fail to converge within the predetermined limit of error function evaluations, their number of gradient evaluations is smaller than the corresponding number of the other methods. Keeping in mind that for some problems, [48], a gradient evaluation is more costly than an error function evaluation (see, for example, [46], where Møller suggests counting gradient evaluations more than error function evaluations), one can understand that these methods require fewer floating point operations and are actually much faster. From the above discussion, it is clear why, in the tables below, there are two rows for the conjugate gradient algorithms and the NMBPVS and NMBBP; the first one indicates the statistics for the FE and the second one for the GE.

*1) 3-Bit Parity Problem:* The parity problem can be considered as a generalized XOR problem but it is more difficult. Here, we have used a more difficult variation, having only two hidden nodes based on hyperbolic tangent activation functions and one linear output node. Thus, the task is to train a 3–2–1 MLP (eight weights, three biases) to produce the sum, mod 2, of three binary inputs—also known as computing the "odd parity" function. The termination condition has been $E \leqslant 0.01$ within 1000 error function evaluations. The results are summarized in Table I.

Despite the effort we have made to fine tune its learning rate, BP has failed to converge within the error function evaluations limit in all the simulation runs. BPM and ABP algorithms performed much better, with BPM slightly outperforming ABP in the particular implementation of the parity problem with two hidden nodes. The conjugate gradient methods PR and PR–FR have been more efficient and effective than the first-order methods, i.e., BP, BPM, and ABP, but the FR method has failed to converge within the error function evaluations limit in all the simulation runs. As shown in Table I, the NMBPVS and NMBBP algorithms exhibit the best performance since they have the highest success rates, as well as the least average number of FE and GE. In addition, they exhibit the fastest execution time, as shown from the results of Fig. 2.

*2) Continuous Function Approximation Problem:* An 1–15–1 MLP (30 weights, 16 biases) is trained to approximate the continuous function $f(x) = \sin(x)\cos(2x)$, where the input values are scattered in the interval $[0, 2\pi]$. The network is trained until $E \leqslant 0.1$ within 1000 error function evaluations. The MLP is based on hidden neurons of logistic activations and on a linear output neuron. Comparative results are shown in Table II.

It is worth noticing the performance of the FR method, since it exhibits the least number of FE and GE, but it has a poor success percentage (17%). The nonmonotone algorithms exhibit very good average performance, having the best success rate, and an average execution time which is better than all other methods tested except FR (see Fig. 3).

*3) Alphabetic Font Learning Problem:* For this problem, 26 matrices with the capital letters of the English alphabet are presented to a 35–30–26 MLP (1830 weights, 56 biases). Each letter has been defined in terms of binary values on a grid of size $5 \times 7$. The network is based on hidden neurons of logistic activations and on linear output neurons. The error function evaluation limit has been set to 2000 and the termination criterion has been $E \leqslant 0.1$. The results in Table III show that the nonmonotone algorithms greatly outperform all other methods tested in terms of FEs and GEs. Furthermore, Fig. 4 shows that the nonmonotone algorithms have the fastest execution time.

TABLE III
COMPARATIVE RESULTS FOR THE ALPHABETIC FONT PROBLEM

| Algorithm | | $Min$ | $\mu$ | $Max$ | $\sigma$ | $Success$ |
|---|---|---|---|---|---|---|
| BP | | 1098 | 1561.9 | 1999 | 202.8 | 76.8% |
| BPM | | 1142 | 1519.1 | 1931 | 169.3 | 4.9% |
| ABP | | 1119 | 1773.1 | 1999 | 168.9 | 37.2% |
| FR | | * | * | * | * | * |
| PR | (FE) | 340 | 1018.5 | 1673 | 275.7 | 100.0% |
| | (GE) | 196 | 936.7 | 1669 | 317.1 | |
| PR–FR | (FE) | 388 | 998.5 | 1715 | 285.1 | 100.0% |
| | (GE) | 244 | 988.3 | 1713 | 292.0 | |
| NMBPVS | (FE) | 337 | 669.4 | 1112 | 134.9 | 100.0% |
| | (GE) | 322 | 633.3 | 1041 | 131.1 | |
| NMBBP | (FE) | 113 | 182.0 | 309 | 33.5 | 100.0% |
| | (GE) | 73 | 119.4 | 193 | 20.5 | |

* the algorithm failed to converge within the error function evaluations limit

TABLE IV
COMPARATIVE RESULTS FOR THE NUMERIC FONT PROBLEM

| Algorithm | | $Min$ | $\mu$ | $Max$ | $\sigma$ | $Success$ |
|---|---|---|---|---|---|---|
| BP | | 9421 | 14489.2 | 19947 | 2783 | 66.2% |
| BPM | | 5328 | 10142.1 | 18756 | 1943 | 54.1% |
| ABP | | 228 | 1975.6 | 13822 | 2509 | 91.2% |
| FR | (FE) | 366 | 2501.2 | 26560 | 5632.4 | 42.0% |
| | (GE) | 260 | 620.3 | 3321 | 571 | |
| PR | (FE) | 806 | 1475.5 | 5585 | 763.7 | 96.1% |
| | (GE) | 148 | 649.7 | 1099 | 109.1 | |
| PR–FR | (FE) | 1498 | 2723.5 | 5737 | 820.1 | 100.0% |
| | (GE) | 533 | 750.3 | 1400 | 120.0 | |
| NMBPVS | (FE) | 104 | 380.4 | 1902 | 217.9 | 100.0% |
| | (GE) | 88 | 268.9 | 1034 | 134.1 | |
| NMBBP | (FE) | 100 | 350.3 | 1532 | 182.5 | 100.0% |
| | (GE) | 100 | 346.4 | 1488 | 178.5 | |

*4) Numeric Font Learning Problem:* In this experiment a 64–6–10 MLP (444 weights, 16 biases) is trained to recognize $8 \times 8$ pixel machine printed numerals from 0 to 9 in helvetica [42]. The network is based on neurons of the logistic activation model, and the weights and biases are initialized with random numbers from the interval $(-1, 1)$. The termination condition for all algorithms tested has been $E \leqslant 10^{-3}$. The results are summarized in Table IV. Clearly, the NMBPVS and NMBBP methods exhibit the best performance having 100% success. NMBPVS has the lowest average number of gradient evaluations and needs the shortest CPU time (see Fig. 5).

### B. Effect of the Nonmonotone Learning Horizon

To study the effect of the nonmonotone learning horizon $M$ on the performance of the nonmonotone training algorithms, the NMBPVS and NMBBP algorithms have been tested in two experiments. For each algorithm 100 independent simulation runs have been performed, using the same random initial weights and biases for both algorithms. We have fixed the value of $\gamma = 10^{-5}$ and we have set the maximum nonmonotone learning horizon to ten.

The first test case concerns the alphabetic font learning problem mentioned above. The weights and biases have been initialized following the Nguyen–Widrow method [49] and the termination condition has been $E \leqslant 0.1$ within 2000 function evaluations.

The second test case is a texture classification problem [40]–[42]. A total of 12 Brodatz texture images [9]: 3, 5, 9, 12, 15, 20, 51, 68, 77, 78, 79, 93 (see Fig. 6) of size $512 \times 512$ is acquired by a scanner at 150 dpi. From each texture image, ten subimages of size $128 \times 128$ are randomly selected, and the cooccurrence method, introduced by Haralick [24] is applied. In the cooccurrence method, the relative frequencies of gray-level pairs of pixels at certain relative displacements are computed and stored in a matrix. As it has been suggested by other researchers [51], [65], the combination of the nearest neighbor pairs at orientations $0^\circ$, $45^\circ$, $90^\circ$, and $135^\circ$ is used in the experiment. A set of ten 16-dimensional training patterns are created from each image. The patterns are presented in a finite sequence $C = (c_1, c_2, \ldots, c_p)$ of input–output pairs $c_p = (u_p, t_p)$ where $u_p$ are the real valued input vectors in $\mathbb{R}^{16}$,

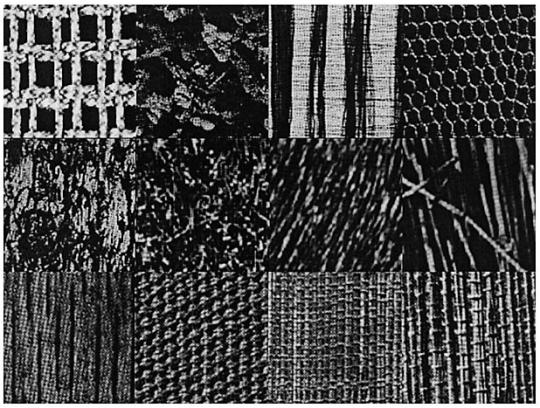

Fig. 6. Twelve texture patterns obtained from the digitizing images found in the "Brodatz Album." Textures: 20, 5, 51, 3, 12, 9, 93, 15, 68, 77, 78, 79.

and $t_p$ are binary output vectors in $\mathbb{R}^{12}$, for $p = 1, \ldots, 120$, determining the corresponding training pattern. A 16–8–12 MLP (224 weights, 20 biases) is trained to classify the patterns into the 12 texture types. The network is based on neurons of logistic activations with biases and the weights and biases were initialized with random numbers from the interval $(-1, 1)$. The termination condition is a classification error $CE \leqslant 3\%$; i.e., the network classifies correctly 117 out of the 120 patterns.

Detailed results regarding the effect of the nonmonotone learning horizon $M$ on the NMBPVS and NMBBP are illustrated in Figs. 7–10. The curves in these figures show the dependency of the average number of error and gradient evaluations from the parameter $M$. The case with an adaptive $M$ is drawn as a straight line from left to right, in order to easily compare it to any of the other test cases. The figures show that the performance of the NMBPVS and NMBBP with an adaptive nonmonotone horizon $M$ is better or equally good to the average performance of any predefined $M$.

Regarding the success percentage of the methods, the NMBPVS exhibits 100% success percentage in both experiments, independently of $M$. On the other hand, the performance of the NMBBP depends on $M$. In the alphabetic font learning problem, the NMBBP has 100% success percentage for every $M$, while in the texture classification problem the method exhibits the highest percentage of success when using an adaptive $M$, as shown in Fig. 11. In this figure, the letter "$A$," along the horizontal axis, indicates the case where an adaptive nonmonotone learning horizon $M$ is used. In general, it seems that the use of the adaptive nonmonotone horizon helps the training procedure.

### C. Effect of the Nonmonotone Strategy

The nonmonotone learning strategy can be incorporated in any training algorithm. Below, we present experimental results, and compare the performance of three training algorithms with their nonmonotone versions. In all cases an adaptive nonmonotone learning horizon has been used and 1000 simulations were run.

*1) BPM Algorithm:* To test the effectiveness of the proposed learning strategy on the BPM algorithm, two experiments have been performed. The first experiment refers to the training of

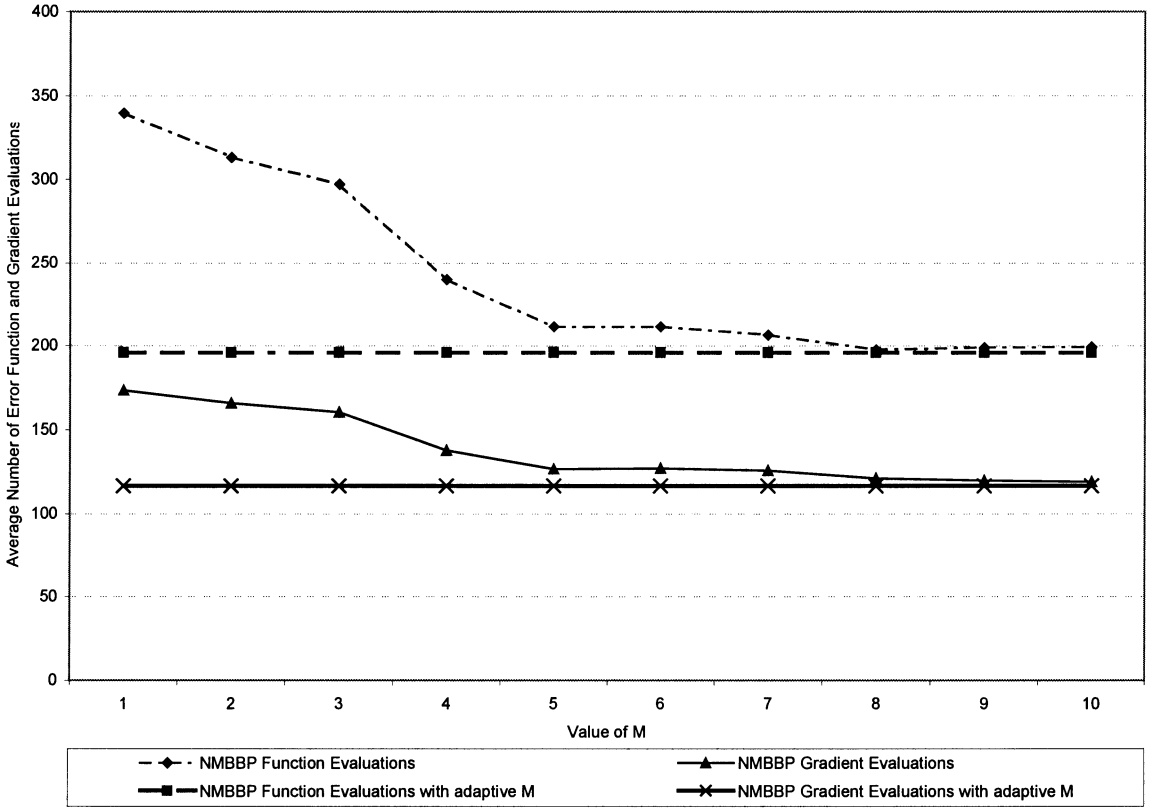


Fig. 7. Alphabetic font learning problem: Average number of error function and gradient evaluations for the NMBBP method with respect to different values of $M$.

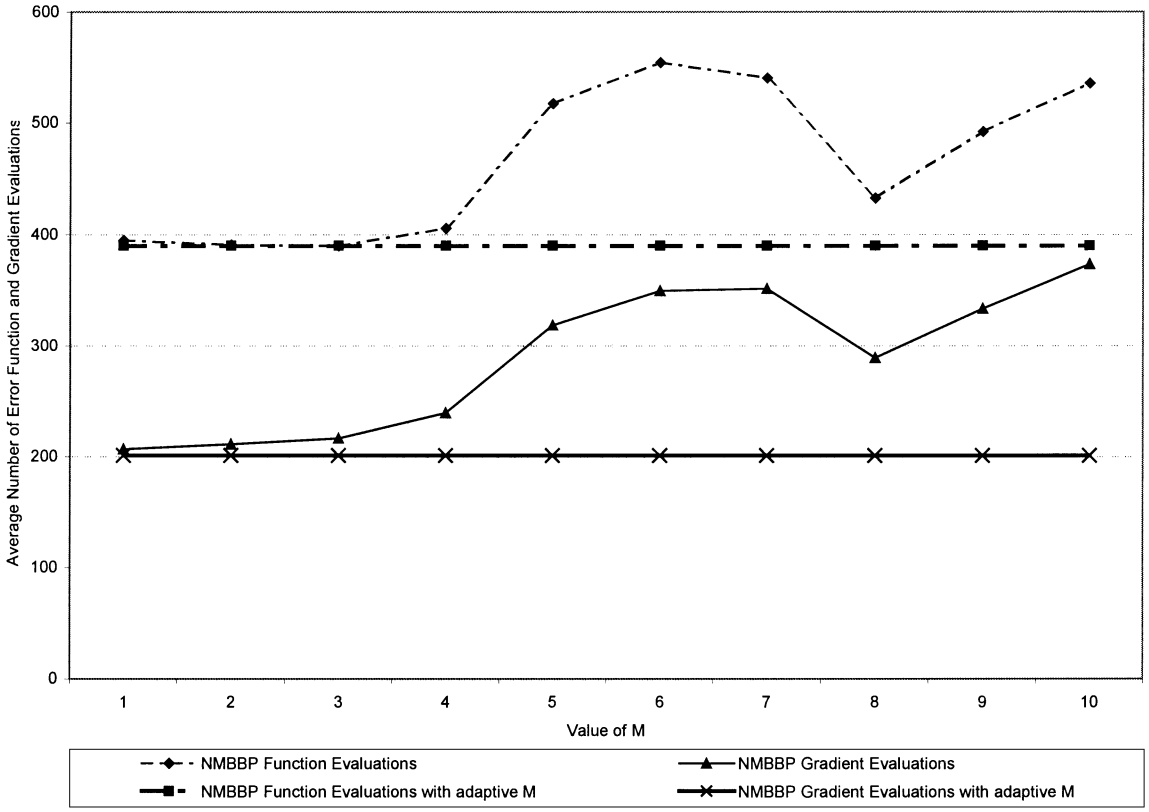


Fig. 8. Texture classification problem: Average number of error function and gradient evaluations for the NMBBP method with respect to different values of $M$.

an MLP for recognizing $8 \times 8$ pixel machine printed numerals from 0 to 9. Details on the network architecture have been provided in Section V-A-4. The termination condition for all algorithms tested is $E \leqslant 10^{-1}$.

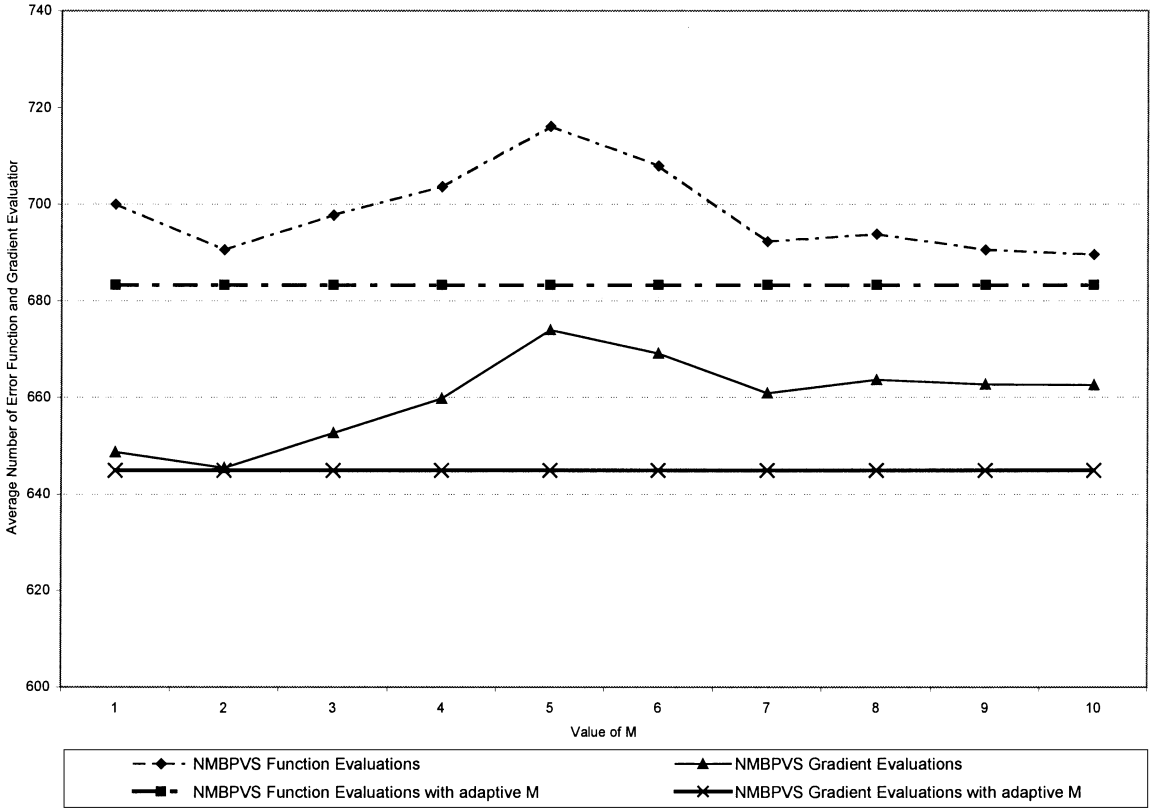


Fig. 9. Alphabetic font learning problem: Average number of error function and gradient evaluations for the NMBPVS method with respect to different values of $M$.

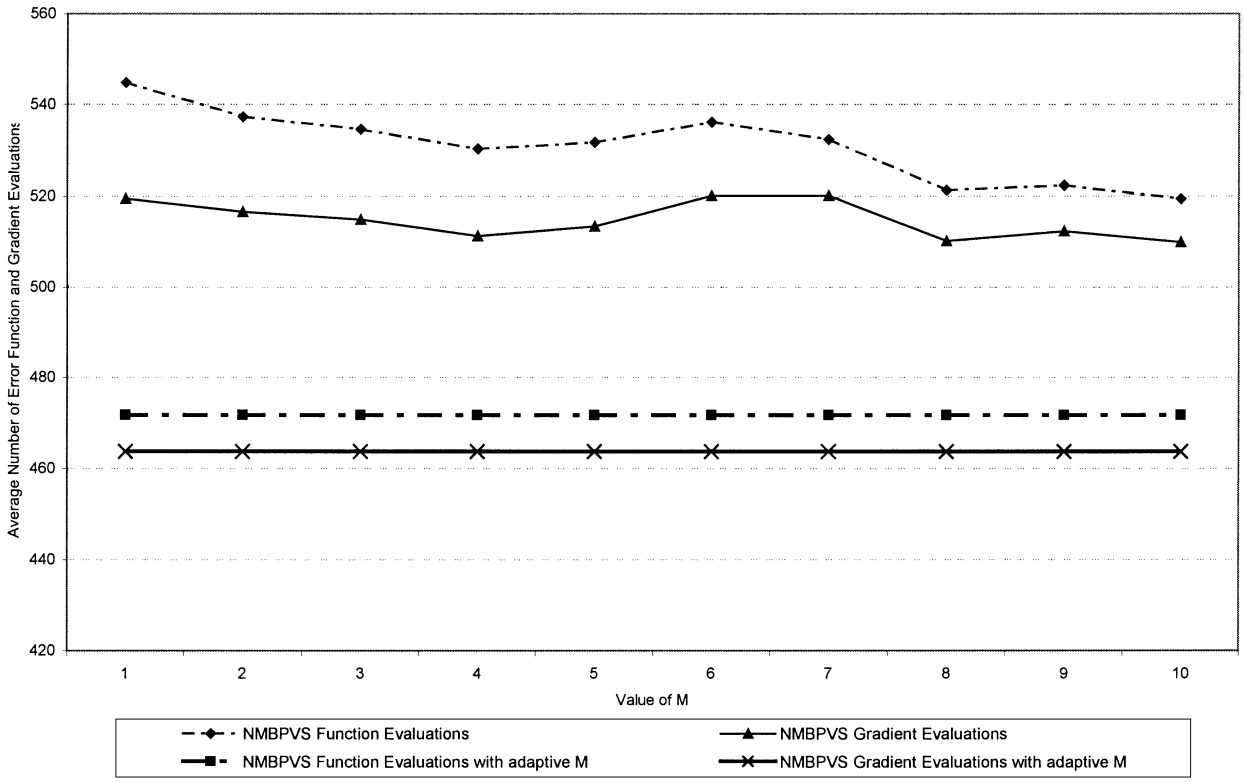


Fig. 10. Texture classification problem: Average number of error function and gradient evaluations for the NMBPVS method with respect to different values of $M$.

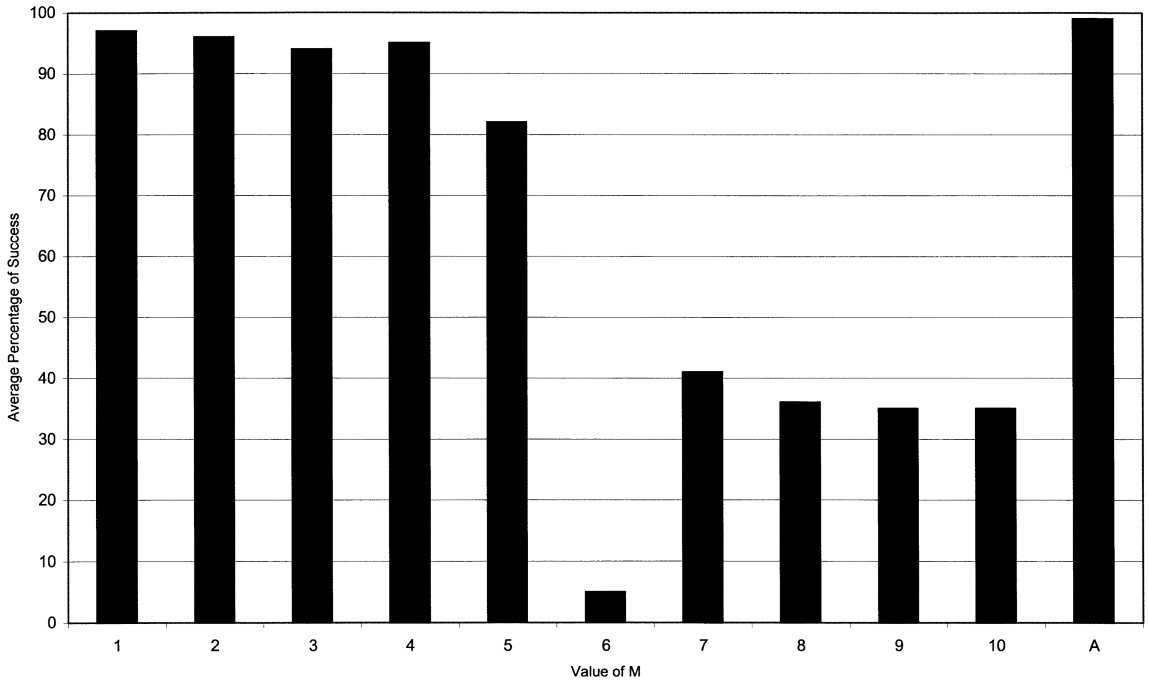


Fig. 11. Texture classification problem: Average percentage of success for the NMBBP method with respect to different values of $M$.

TABLE V
NONMONOTONE BPM APPLIED TO THE NUMERIC FONT PROBLEM

| Algorithm | Gradient Evaluation | | | Function Evaluation | | | Success |
|---|---|---|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $Min/Max$ | $\mu$ | $\sigma$ | $Min/Max$ | % |
| BPM | 560.2 | 684.9 | 239/3962 | 560.2 | 684.9 | 239/3962 | 39% |
| NMBPM | 565.9 | 429.1 | 289/2823 | 571.6 | 428.9 | 295/2827 | 97% |

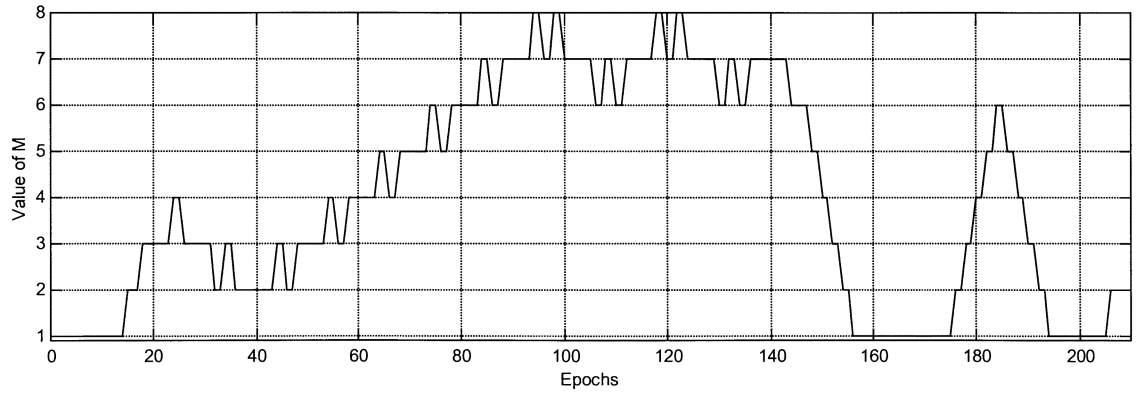


Fig. 12. XOR problem: Behavior of the NMBPM algorithm with adaptive $M^k$.

To evaluate the robustness of the proposed strategy, the learning parameters have been set intentionally to high values, i.e., $\eta = 1.2$ and $m = 0.9$. Detailed results regarding the training performance of the algorithms are presented in Table V, where $\mu$ denotes the mean number of gradient or error function evaluations required to obtain convergence, $\sigma$ the corresponding standard deviation, $\mathrm{Min/Max}$ the minimum and maximum number of gradient or error function evaluations, and *Succ.* denotes the percentage of simulations that converge to a desired minimum out of 1000.

The second experiment that we will consider is the XOR Boolean function problem. The XOR function maps two binary inputs to a single binary output. This simple Boolean function is not linearly separable (i.e., it cannot be solved by a simple mapping directly from the inputs to the output), and thus the use of extra hidden units is required to learn the task. Moreover, it is sensitive to initial weights as well as to learning rate variations, and presents a multitude of local minima with certain weight vectors. An MLP with two hidden neurons of logistic activations with biases, and one linear output neuron with bias (six weights, three biases) has been used to solve this classical problem. The weights and biases have been initialized by the Nguyen–Widrow method [49], and the learning parameters have been: $\eta = 0.4$; $m = 0.9$. The termination criterion has

TABLE VI
NONMONOTONE BPM APPLIED TO THE XOR PROBLEM

| Algorithm | Gradient Evaluation | | | Function Evaluation | | | Success |
|---|---|---|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $Min/Max$ | $\mu$ | $\sigma$ | $Min/Max$ | % |
| BPM | 230.2 | 512.8 | 13/1764 | 230.2 | 512.8 | 13/1764 | 11% |
| NMBPM | 187.8 | 365.1 | 16/1894 | 198.9 | 364.9 | 17/1903 | 80% |

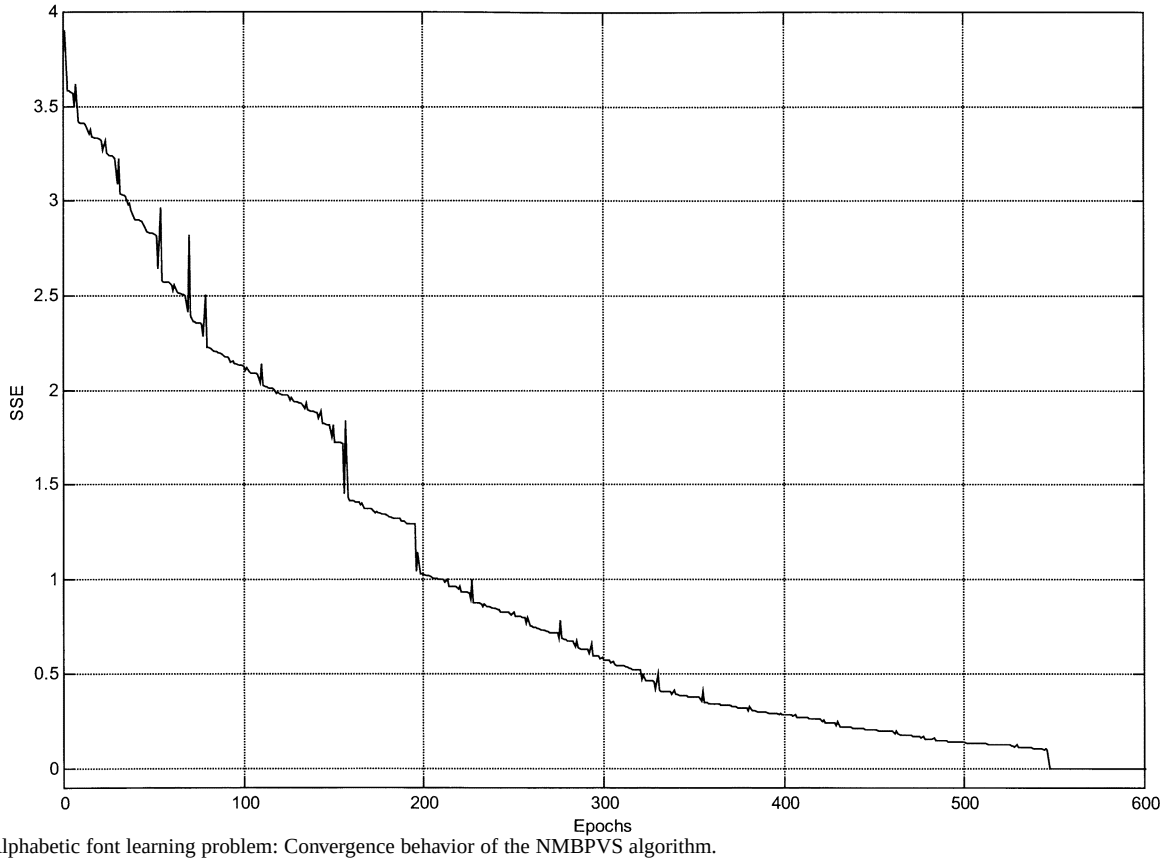


Fig. 13. Alphabetic font learning problem: Convergence behavior of the NMBPVS algorithm.

TABLE VII
NONMONOTONE BPVS APPLIED TO TEXTURE CLASSIFICATION

| Algorithm | Gradient Evaluation | | | Function Evaluation | | | Success |
|---|---|---|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $Min/Max$ | $\mu$ | $\sigma$ | $Min/Max$ | % |
| BPVS | 544.8 | 274.1 | 294/2227 | 519.5 | 257.7 | 283/2101 | 100% |
| NMBPVS | 471.7 | 116.6 | 273/888 | 463.7 | 113.7 | 268/870 | 100% |

been $E \leqslant 0.1$ within 2000 error function evaluations. The results of the simulation are shown in Table VI. The behavior of $M$ during a typical run of the NMBPM algorithm is shown in Fig. 12.

In both test cases, the use of the nonmonotone strategy significantly improves the success percentage of the BPM method. However, in the numeric font problem (see Table V), since fewer runs have converged to a desired minimum for the BPM, the algorithm reveals a lower average number of error function and gradient evaluations for the converged runs.

*2) BPVS Algorithm:* To show the effect of the proposed learning strategy on the performance of the BPVS, we apply it in two applications: the texture classification problem and the continuous function approximation problem, both mentioned previously. The results are exhibited in Tables VII and VIII. The nonmonotone strategy only marginally improves the efficiency of the BPVS method. The NMBPVS error function $E$ in a typical run is shown in Fig. 13, where the nonmonotone convergence behavior of the NMBPVS method is easily observed.

*3) BBP Algorithm:* Next, we test the proposed learning strategies in the function approximation problem. Comparative results are exhibited in Table IX. There is a noticeable improvement of the BBP algorithm when the nonmonotone learning strategy is applied; the BBP algorithm has a success percentage of 79.6%, while the nonmonotone version NMBBP has a success percentage of 92.2%.

The second experiment concerns the alphabetic font problem described above. The results are exhibited in Table X. Both the BBP and the NMBBP have 100% success, however, the nonmonotone version is clearly faster.

TABLE VIII
NONMONOTONE BPVS APPLIED TO FUNCTION APPROXIMATION

| Algorithm | Gradient Evaluation | | | Function Evaluation | | | Success |
|---|---|---|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $Min/Max$ | $\mu$ | $\sigma$ | $Min/Max$ | % |
| BPVS | 417.3 | 220.3 | 44/943 | 446.3 | 236.9 | 48/994 | 63.4% |
| NMBPVS | 429.2 | 228.9 | 64/960 | 443.9 | 238.5 | 64/991 | 66.8% |

TABLE IX
NONMONOTONE BBP APPLIED TO FUNCTION APPROXIMATION

| Algorithm | Gradient Evaluation | | | Function Evaluation | | | Success |
|---|---|---|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $Min/Max$ | $\mu$ | $\sigma$ | $Min/Max$ | % |
| BBP | 186.4 | 111.3 | 27/502 | 362.1 | 233.5 | 39/995 | 79.6% |
| NMBBP | 158.2 | 114.7 | 26/625 | 241.7 | 195.1 | 29/988 | 92.2% |

TABLE X
NONMONOTONE BBP APPLIED TO THE ALPHABETIC FONT PROBLEM

| Algorithm | Gradient Evaluation | | | Function Evaluation | | | Success |
|---|---|---|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $Min/Max$ | $\mu$ | $\sigma$ | $Min/Max$ | % |
| BBP | 169.8 | 35.9 | 90/373 | 332.6 | 70.4 | 167/758 | 100% |
| NMBBP | 119.4 | 20.5 | 73/193 | 182.0 | 33.5 | 113/309 | 100% |

### *D. Generalization Performance*

Generalization performance is a decisive factor when selecting a training algorithm. It is well known that generalization depends on the size of the weights space [17], [35], [67], the noise in the data set [8], [22], [26], [45], the size of the training set [32], and the initial weight distribution [3]. A number of techniques have been proposed to avoid overfitting and improve generalization [22], [32], such as the use of *two-fold* or *$k$-fold cross-validation* [1] and *early stopping* [58]. Results reported in the literature show that these techniques may help in practical applications. However, their success depends on user's choices, and on fine tuning or optimizing problem-dependent heuristic parameters [1], [22], [58].

The approach taken with our experiments has sought to minimize the difficulty associated with parameter selection. Our aim is to verify that the nonmonotone strategies will still do reasonably well in different types of problems without optimizing or fine tuning heuristic parameters and that they provide good generalization capability. Hence, below we examine five problems that possess different characteristics, as our goal is to verify that the nonmonotone strategies work reasonably well over a wide range of application domains. We decided not to enhance the algorithms tested with add-on techniques for improving the generalization as this would require introducing, and fine tuning or optimizing additional heuristics depending on the learning task. Despite the fact we fine tuned learning parameters of the other algorithms tested, we arbitrary set the parameters of the nonmonotone strategies to the same fixed values for all experiments, i.e., $\gamma$ was set to $10^{-5}$ and the maximum value for the adaptive nonmonotone learning horizon was set equal to ten, in order to test the robustness of our methods. Last, we used well studied problems from the UCI Repository of Machine Learning Databases [47], as well as problems studied extensively by other researchers and/or ourselves, in an attempt to reduce as much as possible biases introduced by the size of the weights space.

The first experiment concerns the well known MONK's problems [66]. These are three bipolar classification tasks, which are part of the UCI Repository of Machine Learning Databases, and are used as benchmarks for testing the generalization performance of learning algorithms. The second problem is a handwritten digits classification problem, which is also part of the UCI Repository of Machine Learning Databases, and is characterized by a real-valued training set of approximately 7500 patterns. The third problem is the texture classification studied previously. This problem is used as an example of practical application in which training data are real-valued and noisy due to low resolution in the acquisition of digital data [40]–[42]. The fourth problem concerns the machine-printed numeric font learning task described previously. In this case, the number of patterns in the training set is 45 times smaller than the dimension of the weight space [42], [64]. The fifth problem concerns the identification of malignant tissue regions extracted from standard VHS videotape recordings of endoscopy procedures. In this case, a set of 1200 training data is used. Data contain noise due to shading, shadows, lighting conditions, and reflections, etc. [29], [30].

*1) MONK's Problems [66]:* These problems rely on the artificial robot domain, in which robots are described by six different attributes. Each problem is given by a logical description of the class as follows.

**MONK-1:** (*Attribute1* $=$ *Attribute2*) OR (*Attribute5* $= 1$). This problem is in standard disjunctive normal form (DNF). A set of 124 examples were selected randomly from the data set for training, while the remaining 308 were used for the generalization testing. There were no misclassifications.

**MONK-2:** (Only two attributes $= 1$). This problem is similar to the parity problem mentioned above and is difficult to describe in DNF or conjunctive normal form (CNF). A set of 169 examples were randomly selected from the data set for training, while the rest were used for testing. Again, there was no noise.

TABLE XI
RESULTS OF GENERALIZATION IN THE MONK-3 PROBLEM

| Algorithm | Generalization |
|---|---|
| BP | 93.1% |
| BPWD | 97.2% |
| CC | 97.2% |
| NMBPVS | 100% |
| NMBBP | 100% |

**MONK-3:** (*Attribute5* $=$ 3 AND *Attribute4* $=$ 1) OR (*Attribute5* $\neq$ 4 AND *Attribute2* $\neq$ 3) with added noise. This problem is also in DNF but with 5% deliberate misclassifications in the training set, which consists of 122 examples. The remaining 310 examples were used for testing.

Each one of the six attributes can have one of three, three, two, three, four, and two values, respectively, which results in 432 possible combinations that constitute the total data set (see [66], for details). Finally, each possible value for every attribute was assigned a single bipolar input, resulting in 17 inputs. The values of the attributes are shown as follows.

- *Attribute1*: HeadShape $\in$ {round, square, octagon}.
- *Attribute2*: BodyShape $\in$ {round, square, octagon}.
- *Attribute3*: IsSmiling $\in$ {yes, no}.
- *Attribute4*: Holding $\in$ {sword, balloon, flag}.
- *Attribute5*: JacketColor $\in$ {red, yellow, green, blue}.
- *Attribute6*: HasTie $\in$ {yes, no}.

For example, the robot with: *Attribute1* $=$ round, *Attribute2* $=$ square, *Attribute3* $=$ *yes*, *Attribute4* $=$ flag, *Attribute5* $=$ blue, and *Attribute6* $=$ no, is encoded as: (1, $-1$, $-1$, $-1$, 1, $-1$, 1, $-1$, $-1$, $-1$, 1, $-1$, $-1$, $-1$, 1, $-1$, 1).

We have tested the nonmonotone algorithms against the well known BP, BP with weight decay (BPWD), and cascade correlation (CC) methods. In our simulation, we have used the same network topologies as those found in [66] for the BP method.

All the tested methods exhibited 100% classification success in the MONK-1 and MONK-2 problems, but in MONK-3, where there are 5% deliberate misclassifications, the networks generated by BP, BPWD, and CC seem to fail to capture the concept embedded in the training data, and fit to the noise instead. Table XI shows that the NMBPVS and NMBBP algorithms are excellent generalizers, and manage to correctly classify all the input patterns in the MONK-3 problem.

*2) Pen-Based Recognition of Handwritten Digits Problem:* The pen-based recognition of handwritten digits classification problem is also part of the UCI Repository of Machine Learning Databases [47]. In this experiment, a digit database has been assembled by collecting 250 samples from 44 independent writers. The samples written by 30 writers are used for training, and the rest are used for testing. The training set consists of 7494 real-valued samples and the test set of 3498 real-valued samples. The architecture found to exhibit better average performance was a network with 50 hidden neurons. Thus, a 16–50–10 MLP, having hidden neurons based on logistic activations and output neurons based on linear activation functions, was trained. The training procedure stopped when the MLP exhibited 2% misclassifications in the training set. The average success percentage of classification for each algorithm in 100 independent tests was: BP $=$ 90.93%; BPM $=$ 91.08%; ABP $=$ 92.02%; FR $=$ 96.76%; PR $=$ 97.11%; PR–FR $=$ 97.25%; NMBPVS $=$ 98.25%; NMBBP $=$ 98.39%.

*3) Texture Classification Problem:* The texture classification problem, mentioned in a previous subsection, is chosen to compare and evaluate the training algorithms, as an example of practical application, using continuous-valued training data that contain random noise. On average, BP trained networks with a 16–8–12 architecture perform better than others according to our previous experiments [40], [41]. Thus, 1000 trained MLPs of this architecture were tested for their generalization capability, using test patterns from 20 subimages of size 128 $\times$ 128, randomly selected from each image. To evaluate the average generalization performance of the MLPs the $\max$ rule was used, i.e., a test pattern is considered to be correctly classified if the corresponding output neuron has the greatest value among the output neurons. The average success percentage of classification for each algorithm is: BP $=$ 90.0%; BPM $=$ 90.0%; ABP $=$ 93.5%; FR $=$ 92.0%; PR $=$ 92.6%; PR–FR $=$ 93.5%; NMBPVS $=$ 93.6%; NMBBP $=$ 93.6%. The nonmonotone algorithms provided marginal improvement with respect to PR–FR but required the least average number of FE and GE to converge.

*4) Numeric Font Problem:* Numerals from zero to nine in eight points standard helvetica font form the training patterns for this application example. A 64–6–10 architecture was used as this was found to exhibit better average performance than other architectures trained with the BP algorithm [42], [64]. After being trained, with BP, BPM, ABP, FR, PR, PR–FR, NMBPVS and NMBBP, as in Section V-A4, MLPs were tested for their generalization capability using helvetica italic. Note that, the test patterns in italic have 6–14 bit reversed from the training patterns, and that 100 MLPs were trained for each case. To evaluate the average generalization performance the $\max$ rule was used.

The BP and BPM trained MLPs achieved similar generalization capability as the ABP trained ones, but more epochs were necessary in order to converge. The same holds for the FR and PR methods. Thus, in Fig. 14, only the performance of the ABP, PR, PR–FR, NMBPVS, and NMBBP is illustrated.

This figure shows the number of trained MLPs (out of 100 runs) that correctly recognize the numeric symbols $0, 1, \ldots, 9$ with respect to the total number of correctly recognized numeric symbols. For example, 32 out of 100 NMBBP trained MLPs correctly recognize six out of the ten numeric symbols $0, 1, \ldots, 9$. On average, there is a higher number of nonmonotone trained MLPs that correctly recognize five, six, seven, eight, nine, and ten numeric symbols out of the ten in the testing phase. For example, approximately 50% of the NMBPVS and NMBBP trained MLPs correctly recognize six or more of the numeric symbols $0, 1, \ldots, 9$. On the other hand, approximately 35% of the PR–FR trained MLPs exhibit a similar performance.

*5) Abnormalities Detection in Endoscopy Video Sequences:* Detecting malignant regions in video sequences is a difficult task due to variations of the environmental conditions during the endoscopy procedure. Textures from normal and abnormal tissue samples have been randomly chosen from four

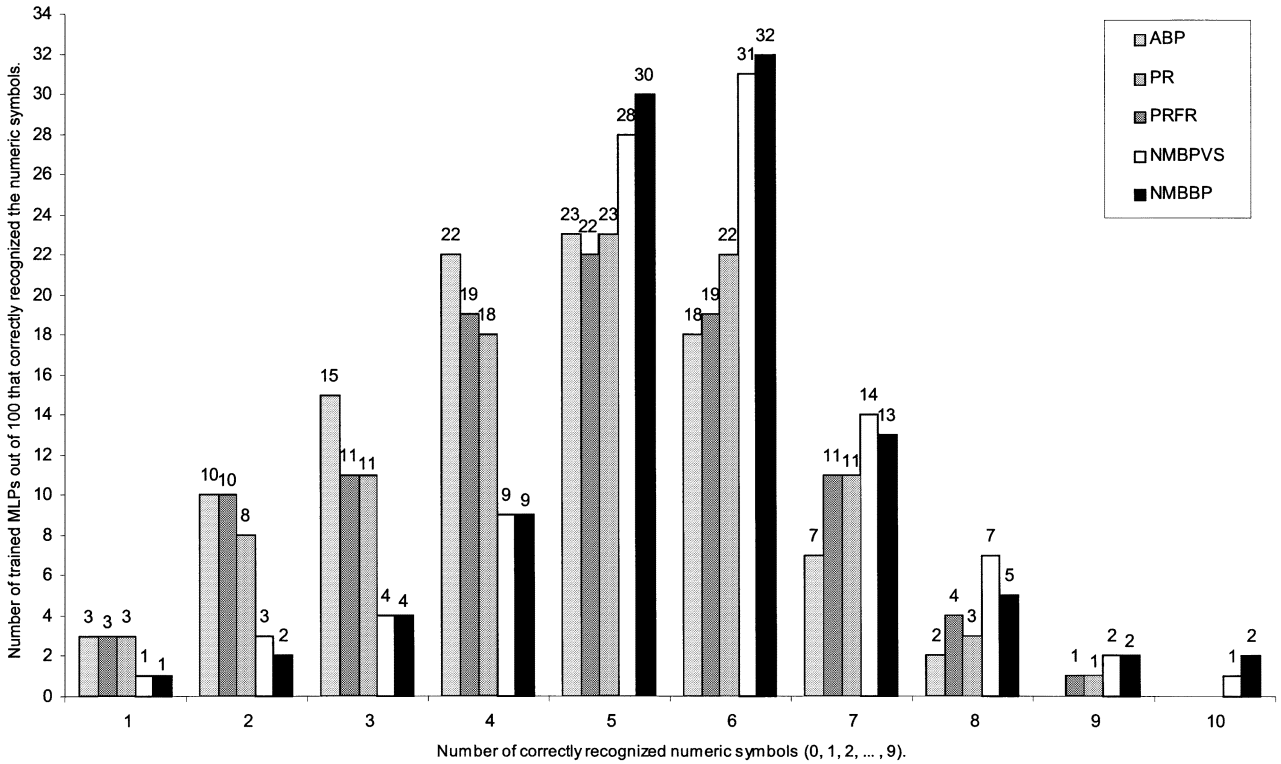


Fig. 14. Numeric font learning problem: Number of trained MLPs out of 100 that correctly recognized the numeric symbols $0, 1, \ldots, 9$ in helvetica italic.

TABLE XII
RESULTS FOR THE ABNORMALITIES DETECTION PROBLEM

| Algorithm | | $Min$ | $\mu$ | $Max$ | $\sigma$ | $Success$ |
|---|---|---|---|---|---|---|
| BP | | 7697 | 8505.5 | 9314 | 1143.39 | 20% |
| BPM | | 5685 | 8500.0 | 9315 | 952.58 | 32% |
| ABP | | 453 | 734.1 | 1055 | 249.37 | 99% |
| NMBPVS | (FE) | 261 | 374.7 | 515 | 129.08 | 100% |
| | (GE) | 166 | 231.6 | 328 | 107.81 | |
| NMBBP | (FE) | 263 | 656.3 | 955 | 227.24 | 100% |
| | (GE) | 151 | 376.6 | 561 | 197.43 | |

TABLE XIII
GENERALIZATION IN THE ABNORMALITIES DETECTION PROBLEM

| Algorithm | BP | BPM | ABP | NMBBP | NMBPVS |
|---|---|---|---|---|---|
| Generalization | 78.09% | 78.09% | 79.10% | 83.91% | 85.12% |

video frames of the same sequence, and used for training the network to discriminate between malignant and normal regions.

To generate the training set the cooccurrence matrices have been used. More specifically, the colonoscopic image was separated into windows of size 16 pixels by 16 pixels. Then the cooccurrence matrices algorithm was used to gather information regarding each pixel in an image window [29]–[31]. Cooccurrence matrices, [25], represent the spatial distribution and the dependence of the gray levels within a local area (see [29] and [30] for further technical details).

The feature vectors contain 16 elements each and therefore the first layer of the MLPs will consist of 16 neurons. 100 MLPs with 16 inputs, 30 hidden, and two output neurons have been trained to discriminate between normal and abnormal image regions using 1200 randomly selected patterns from four video frames. This network architecture was identified as providing on average the best results for the classical BP following preliminary experiments reported in [29], [30], [73], and [74]. The training procedure stopped when the MLPs exhibited 3% misclassifications on the training set. Table XII summarizes the convergence performance of the tested algorithms for simulations that reached solution, out of 100 trials.

To test the generalization performance of the trained MLPs approximately 16 000 test patterns have been created. This test set constitutes the whole image region in each of the four frames and contains normal and abnormal samples. In Table XIII, the generalization capability of the algorithms is exhibited.

## VI. CONCLUSION

Deterministic nonmonotone learning strategies for MLP training were proposed in this paper. According to this approach, the error function value must satisfy a nonmonotone criterion with respect to the maximum error function value of the $M$ previous epochs, which constitute the nonmonotone learning horizon.

A procedure for the adaptation of the nonmonotone learning horizon based on the local estimation of the Lipschitz constant was proposed. The experiments indicate that the use of an adaptive $M$ at each iteration helps to reduce the number of gradient and error function evaluations required to obtain convergence.

The nonmonotone strategies can be incorporated in any batch training algorithm, providing stable learning and, therefore, a greater possibility of good performance. The simulation results suggest that the use of the nonmonotone strategies significantly accelerates the convergence of the first-order training algorithms as measured by the number of error function and gradient

evaluations, and the average CPU time for convergence. In addition, the nonmonotone training algorithms lead to good quality solutions, in the sense that final weight vectors provide on the average improved generalization capability with no need for fine-tuning problem-dependent heuristic parameters.

The nonmonotone algorithms were compared against some well known conjugate gradient training algorithm, which apply inexact line search techniques to ensure the monotone decrease of the learning error. Numerical evidence shows that the nonmonotone strategies improve the efficiency and effectiveness of the first-order methods. It is worth noting that, in certain cases, the nonmonotone methods exhibit faster, or equally fast, convergence than the conjugate gradient methods as shown by the average CPU time needed for convergence.

## ACKNOWLEDGMENT

The authors wish to thank the editors and a referee for their constructive comments and useful suggestions.

## REFERENCES

[1] S. Amari, N. Murata, K.-R. Müller, M. Finke, and H. H. Yang, "Asymptotic statistical theory of overtraining and cross-validation," *IEEE Trans. Neural Networks*, vol. 8, pp. 985–996, Sept. 1997.

[2] L. Armijo, "Minimization of functions having Lipschitz-continuous first partial derivatives," *Pacific J. Math.*, vol. 16, pp. 1–3, 1966.

[3] A. Atiya and C. Ji, "How initial conditions affect generalization performance in large networks," *IEEE Trans. Neural Networks*, vol. 8, pp. 448–451, Mar. 1997.

[4] J. Barzilai and J. M. Borwein, "Two point step size gradient methods," *IMA J. Numer. Anal.*, vol. 8, pp. 141–148, 1988.

[5] R. Battiti, "Accelerated backpropagation learning: Two optimization methods," *Complex Syst.*, vol. 3, pp. 331–342, 1989.

[6] ——, "First- and second-order methods for learning: Between steepest descent and Newton's method," *Neural Comput.*, vol. 4, pp. 141–166, 1992.

[7] S. Becker and Y. Le Cun, "Improving the convergence of the back-propagation learning with second order methods," in *Proc. 1988 Connectionist Models Summer School*, D. S. Touretzky, G. E. Hinton, and T. J. Sejnowski, Eds. San Mateo, CA: Morgan Kaufmann, 1988, pp. 29–37.

[8] C. M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Comput.*, vol. 7, pp. 108–116, 1995.

[9] P. Brodatz, *Textures—A Photographic Album for Artists and Designers*. New York: Dover, 1966.

[10] A. Cauchy, "Méthode générale pour la résolution des systèmes d'équations simultanées," *Comp. Rend. Acad. Sci. Paris*, vol. 25, pp. 536–538, 1847.

[11] L. W. Chan and F. Fallside, "An adaptive training algorithm for backpropagation networks," *Comput. Speech Language*, vol. 2, pp. 205–218, 1987.

[12] J. E. Dennis and J. J. Moré, "Quasi-Newton methods, motivation and theory," *SIAM Rev.*, vol. 19, pp. 46–89, 1977.

[13] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Philadelphia, PA: SIAM, 1996.

[14] J. L. Elman, E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett, "The shape of change," in *Rethinking Innateness: A Connectionist Perspective on Development*. Cambridge, MA: MIT Press, 1997, ch. 6.

[15] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," in *Proc. 1988 Connectionist Models Summer School*, D. S. Touretzky, G. E. Hinton, and T. J. Sejnowski, Eds. San Mateo, CA: Morgan Kaufmann, 1988, pp. 38–51.

[16] A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Philadelphia, PA: SIAM, 1990.

[17] O. Fujita, "Statistical estimation of the number of hidden units for feedforward neural networks," *Neural Networks*, vol. 11, pp. 851–859, 1998.

[18] J. C. Gilbert and J. Nocedal, "Global convergence properties of conjugate gradient methods for optimization," *SIAM J. Optimization*, vol. 2, pp. 21–42, 1992.

[19] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. New York: Academic, 1981.

[20] A. A. Goldstein, "Cauchy's method of minimization," *Numer. Math.*, vol. 4, pp. 146–150, 1962.

[21] L. Grippo, F. Lampariello, and S. Lucidi, "A nonmonotone line search technique for Newton's method," *SIAM J. Numer. Anal.*, vol. 23, pp. 707–716, 1986.

[22] A. Gupta and S. M. Lam, "Weight decay backpropagation for noisy data," *Neural Networks*, vol. 11, pp. 1127–1137, 1998.

[23] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Boston, MA: PWS, 1996.

[24] R. Haralick, K. Shanmugan, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man. Cybern.*, vol. SMC-3, pp. 610–621, 1973.

[25] R. M. Haralick, "Statistical and structural approaches to texture," *Proc. IEEE*, vol. 67, pp. 786–804, 1979.

[26] L. Holmstrom and P. Koistinen, "Using additive noise in backpropagation training," *IEEE Trans. Neural Networks*, vol. 3, pp. 24–38, Jan. 1992.

[27] H. C. Hsin, C. C. Li, M. Sun, and R. J. Sclabassi, "An adaptive training algorithm for back-propagation neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 512–514, Apr. 1995.

[28] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295–307, 1988.

[29] S. A. Karkanis, D. K. Iakovidis, D. E. Maroulis, G. D. Magoulas, and N. G. Theofanous, "Tumor recognition in endoscopic video images using artificial neural network architectures," in *Proc. 26th Euromicro Conf.*, Maastricht, The Netherlands, September 5–7, 2000, pp. 423–429.

[30] S. A. Karkanis, G. D. Magoulas, D. K. Iakovidis, D. A. Karras, and D. E. Maroulis, "Evaluation of textural feature extraction schemes for neural network-based interpretation of regions in medical images," in *Proc. IEEE Int. Conf. Image Processing (ICIP-2001)*, vol. 1, Thessaloniki, Greece, 2001, pp. 281–284.

[31] S. Karkanis, G. D. Magoulas, and N. Theofanous, "Image recognition and neuronal networks: Intelligent systems for the improvement of imaging information," *Minimally Invasive Therapy Allied Technol.*, vol. 9, pp. 225–230, 2000.

[32] G. N. Karystinos and D. A. Pados, "On overfitting, generalization, and randomly expanded training sets," *IEEE Trans. Neural Networks*, vol. 11, pp. 1050–1057, Sept. 2000.

[33] C. M. Kuan and K. Hornik, "Convergence of learning algorithms with constant learning rates," *IEEE Trans. Neural Networks*, vol. 2, pp. 484–488, Mar. 1991.

[34] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3rd ed. New York: McGraw-Hill, 2000.

[35] S. Lawrence, C. L. Giles, and A. C. Tsoi, "What size neural network gives optimal generalization? Convergence properties of backpropagation," Univ. Maryland Tech. Rep. CS-TR-3617, 1996.

[36] Y. Le Cun, P. Y. Simard, and B. A. Pearlmutter, "Automatic learning rate maximization by on-line estimation of the Hessian's eigenvectors," in *Advances in Neural Information Processing Systems 5*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds. San Mateo, CA: Morgan Kaufmann, 1993, pp. 156–163.

[37] Y. Lee, S. H. Oh, and M. W. Kim, "An analysis of premature saturation in backpropagation learning," *Neural Networks*, vol. 6, pp. 719–728, 1993.

[38] R. Liu, G. Dong, and X. Ling, "A convergence analysis for neural networks with constant learning rates and nonstationary inputs," in *Proc. 34th Conf. Decision Contr.*, New Orleans, 1995, pp. 1278–1283.

[39] C. G. Looney, *Pattern Recognition Using Neural Networks*. New York: Oxford University Press, 1997.

[40] G. D. Magoulas, S. A. Karkanis, D. A. Karras, and M. N. Vrahatis, "Comparison study of textural descriptors for training neural network classifiers," Int. J. Comput. Res., Oct. 2002, to be published.

[41] G. D. Magoulas, V. P. Plagianakos, and M. N. Vrahatis, "Hybrid methods using evolutionary algorithms for on-line training," in *Proc. IEEE Int. Joint Conf. Neural Networks*, Washington, DC, 2001, pp. 2218–2223.

[42] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis, "Effective back-propagation with variable stepsize," *Neural Networks*, vol. 10, pp. 69–82, 1997.

[43] ——, "Improving the convergence of the back-propagation algorithm using learning rate adaptation methods," *Neural Computation*, vol. 11, pp. 1769–1796, 1999.

[44] G. D. Magoulas, M. N. Vrahatis, T. N. Grapsa, and G. S. Androulakis, "Neural network supervised training based on a dimension reducing method," in *Mathematics of Neural Networks: Models, Algorithms and Applications*, S. W. Ellacot, J. C. Mason, and I. J. Anderson, Eds: Kluwer, 1997, pp. 245–249.

[45] K. Matsuoka, "Noise injection into inputs in backpropagation learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 436–440, Mar. 1992.
[46] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525–533, 1993.
[47] P. M. Murphy and D. W. Aha. (1994) UCI repository of machine learning databases. Univ. California, Dept. Inform. Comput. Sci., Irvine, CA. [Online]. Available: http://www.ics.uci.edu/ mlearn/MLRepository.html
[48] E. Mizutani and S. E. Dreyfus, "On complexity analysis of supervised MLP-learning for algorithmic comparisons," in *Proc. Int. Joint Conf. Neural Networks*, Washington, DC, 2001, pp. 347–352.
[49] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural network by choosing initial values of the adaptive weights," in *IEEE Proc. 1st Int. Joint Conf. Neural Networks*, vol. 3, 1990, pp. 21–26.
[50] J. Nocedal, "Theory of algorithms for unconstrained optimization," *Acta Numerica*, vol. 1, pp. 199–242, 1992.
[51] P. P. Ohanian and R. C. Dubes, "Performance evaluation for four classes of textural features," *Pattern Recognition*, vol. 25, pp. 819–833, 1992.
[52] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic, 1970.
[53] D. B. Parker, "Optimal algorithms for adaptive networks: Second order back-propagation, second order direct propagation, and second order Hebbian learning," in *Proc. IEEE Int. Conf. Neural Networks*, 1987, pp. 593–600.
[54] M. Pfister and R. Rojas, "Speeding-up backpropagation—A comparison of orthogonal techniques," in *Proc. Joint Conf. Neural Networks*, Nagoya, Japan, 1993, pp. 517–523.
[55] V. P. Plagianakos, D. G. Sotiropoulos, and M. N. Vrahatis, "Automatic adaptation of learning rate for backpropagation neural networks," in *Recent Advances in Circuits and Systems*, N. E. Mastorakis, Ed. Singapore: World Scientific, 1998, pp. 337–341.
[56] V. P. Plagianakos, M. N. Vrahatis, and G. D. Magoulas, "Nonmonotone methods for backpropagation training with adaptive learning rate," in *Proc. IEEE Int. Joint Conf. Neural Networks*, vol. 3, Washington, DC, 1999, pp. 1762–1767.
[57] E. Polak, *Optimization: Algorithms and Consistent Approximations*. New York: Springer-Verlag, 1997.
[58] L. Prechelt, "Automatic early stopping using cross validation: Quantifying the criteria," *Neural Networks*, vol. 11, pp. 761–767, 1998.
[59] M. Raydan, "The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem," *SIAM J. Optimization*, vol. 7, pp. 26–33, 1997.
[60] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The Rprop algorithm," in *Proc. IEEE Int. Conf. Neural Networks*, San Francisco, CA, 1993, pp. 586–591.
[61] A. K. Rigler, J. M. Irvine, and T. P. Vogl, "Rescaling of variables in backpropagation learning," *Neural Networks*, vol. 4, pp. 225–229, 1991.
[62] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, vol. 1, pp. 318–362.
[63] F. Silva and L. Almeida, "Acceleration techniques for the back-propagation algorithm," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1990, vol. 412, pp. 110–119.
[64] A. Sperduti and A. Starita, "Speed up learning and network optimization with extended back-propagation," *Neural Networks*, vol. 6, pp. 365–383, 1993.
[65] J. Strang and T. Taxt, "Local frequency features for texture classification," *Pattern Recognition*, vol. 27, pp. 1397–1406, 1994.
[66] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufmann, S. Keller, I. Kononenko, J. Kreuziger, R. S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang, "The MONK's problems: A performance comparison of different learning algorithms," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep., CMU-CS-91-197, 1991.
[67] N. K. Treadgold and T. D. Gedeon, "Exploring constructive cascade networks," *IEEE Trans. Neural Networks*, vol. 10, pp. 1335–1350, Nov. 1999.
[68] P. P. Van der Smagt, "Minimization methods for training feedforward neural networks," *Neural Networks*, vol. 7, pp. 1–11, 1994.
[69] M. N. Vrahatis, G. S. Androulakis, J. N. Lambrinos, and G. D. Magoulas, "A class of gradient unconstrained minimization algorithms with adaptive stepsize," *J. Comput. Appl. Math.*, vol. 114, pp. 367–386, 2000.
[70] M. N. Vrahatis, G. D. Magoulas, and V. P. Plagianakos, "Globally convergent modification of the Quickprop method," *Neural Processing Lett.*, vol. 12, pp. 159–169, 2000.
[71] T. P. Vogl, J. K. Mangis, J. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method," *Biol. Cybern.*, vol. 59, pp. 257–263, 1988.
[72] R. L. Watrous, "Learning algorithms for connectionist networks: Applied gradient of nonlinear optimization," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 2, 1987, pp. 619–627.
[73] S. A. Karkanis, D. K. Iakovidis, D. E. Maroulis, G. D. Magoulas, and N. G. Theofanous, "Tumor recognition in endoscopic video images using artificial neural network architectures," in *Proc. 26th Euromicro Conf.*, F. Vajda, Ed. Los Alamitos, CA: IEEE Press, 2000.
[74] S. A. Karkanis, G. D. Magoulas, D. K. Iakovidis, D. A. Karras, and D. E. Maroulis, "Evaluation of textural feature extraction schemes for neural network-based interpetation of regions in medical images," in *Proc. IEEE Int. Conf. Image Processing (ICIP-2001)*. Los Alamitos, CA: IEEE Press, 2001.

**Vassilis P. Plagianakos** was born in Githio, Greece, in 1973. He received the Diploma degree in mathematics from the Department of Mathematics, University of Patras, Greece, in 1996. He is currently pursuing the Ph.D. degree in mathematics at the same department.

Since 1996, he has been a Research Assistant with the Computational Intelligence Group, Department of Mathematics, University of Patras and participated in several research projects funded by public and private organizations. His research interests are in the areas of neural-network training, evolutionary algorithms, optimization and parallel algorithms.

Mr. Plagianakos is a Student Member of the IEEE Neural Networks Society.

**George D. Magoulas** (M'02) was born in Athens, Greece, in 1966. He received the Bachelor's degree in electrical and computer engineering and the Ph.D. degree in neural network learning, both from the Department of Electrical and Computer Engineering of the University of Patras in Greece.

From 1990 to 1993 and from 1997 to 1998, he worked in Greek industry. He participated in several projects, and developed software tools for the design and simulation of fuzzy logic controllers and neuro-fuzzy controllers for real-time control of cement plants and for embedded automotive applications. In 1998, he took up the post of Research Fellow in the Department of Informatics and Telecommunications, University of Athens, Greece, where he participated in EU-funded projects, followed in 1999, by a postdoctoral fellowship from the Greek State Scholarship Foundation (I.K.Y.) in the Department of Mathematics at the University of Patras, Greece. In 2000, he was appointed to his first lectureship in the Department of Information Systems and Computing at Brunel University, West London, U.K. His research interests include methods for learning and evolution, intelligent optimization, bioinformatics and real-world problem solving.

Dr. Magoulas is a Member of the Operational Research Society, the Technical Chamber of Greece, and the Hellenic Artificial Intelligence Society.

**Michael N. Vrahatis** received the Bachelor's and Ph.D. degrees in mathematics from the University of Patras, Greece, in 1978 and 1982, respectively.

He was a Visiting Rresearch Fellow at the Department of Mathematics, Cornell University, Ithaca, NY, from 1987 to 1988 and a Visiting Professor to the INFN (Istituto Nazionale di Fisica Nucleare), Bologna, Italy, in 1992, 1994, and 1998; the Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium, in 1999; the Department of Ocean Engineering, Design Laboratory, Massachusetts Institute of Technolog, Cambridge, in 2000, and the Collaborative Research Center "Computational Intelligence" (SFB 531) at the Department of Computer Science, University of Dortmund, Dortmund, Germany, in 2001. He has been a Visiting Researcher at CERN (European Organization of Nuclear Research), Geneva, Switzerland, in 1992, and at the Institut National de Recherche en Informatique et en Automatique (INRIA), France, in 1998. Since August 2000, he has been a Professor at the Department of Mathematics, University of Patras, Greece. He is the author of more than 150 publications in both pure and applied mathematics, including optimization, neural network training, genetic and evolutionary algorithms, data mining, cryptography and artificial intelligence. He has been principal investigator of several research grants from the European Union and the Hellenic Ministry of Industry, Energy and Technology. He is among the founders of the University of Patras Artificial Intelligence Research Center (UPAIRC), established in 1997, where currently he serves as Director.