Pergamon

Neural Networks, Vol. 8, No. 2, pp. 237–249, 1995

0893-6080/95 $9.50 + .00

0893-6080(94)00067-0

CONTRIBUTED ARTICLE

An Efficient Constrained Learning Algorithm With Momentum Acceleration

STAVROS J. PERANTONIS AND DIMITRIS A. KARRAS

Institute of Informatics and Telecommunications, National Research Center "Demokritos"

(*Received* 20 *April* 1993; *revised accepted* 17 *June* 1994)

Abstract—*An algorithm for efficient learning in feedforward networks is presented. Momentum acceleration is achieved by solving a constrained optimization problem using nonlinear programming techniques. In particular, minimization of the usual mean square error cost function is attempted under an additional condition for which the purpose is to optimize the alignment of the weight update vectors in successive epochs. The algorithm is applied to several benchmark training tasks (exclusive-or, encoder, multiplexer, and counter problems). Its performance, in terms of learning speed and scalability properties, is evaluated and found superior to the performance of reputedly fast variants of the back-propagation algorithm in the above benchmarks.*

Keywords—Feedforward neural networks, Supervised learning, Momentum acceleration, Nonlinear programming, Constraints, Lagrange multipliers.

1. INTRODUCTION

Multilayer feedforward neural networks (MFNN) have been the subject of intensive research efforts because of their interesting learning and generalization abilities. Of particular importance is the rigorous theoretical establishment that these networks are universal approximators (Hornik, Stinchcombe, & White, 1989; Funahashi, 1989), once properly trained. The problem of devising efficient algorithms for training MFNNs is thus of central importance in neural network research and has been thoroughly studied in recent years. Following the back–propagation (BP) algorithm and its momentum acceleration variant (Rumelhart, Hinton, & Williams, 1986a,b), a multitude of supervised learning algorithms have been devised with the aim of improving the learning speed and generalization capability of these networks. In particular, methods originating from the field of numerical analysis [second order (Parker, 1987; Becker & le Cun, 1988) and line search, conjugate gradient (Kramer & Sangiovanni–Vincentelli, 1988), and quasi-Newton (Watrous, 1987) methods] and from the field of optimal filtering [extended Kalman algorithm (Singhal & Wu, 1989)], as well as heuristic optimization techniques that perform a search in the weight space [delta-bar-delta (Jacobs, 1988) and quickprop (Fahlman, 1988)], have been proposed.

A common objective of these algorithms is to adapt the synaptic weights until the activation of the network's output layer nodes matches prespecified values–targets. Apart from this *sine qua non* condition, some algorithms incorporate in their formulation additional information about learning in MFNNs. For example, attempts to increase learning speed by imposing additional conditions aimed at helping the hidden nodes to play a more active role during training (Grossman, 1990; Grossmann, Meir, & Domany, 1990; Rohwer, 1990; Krogh, Thorbergsson, & Hertz, 1990), as well as attempts to improve generalization by enabling the decay of redundant weights (Weigend, Rumelhart, & Huberman, 1991), have been reported in the literature.

Along this line of research, the authors have proposed methods for incorporating useful information in the learning algorithm in the form of additional conditions—apart from the demand for minimization of the cost function—that must be satisfied during learning. Techniques of nonlinear programming have been utilized to solve the resulting constrained optimization problems. As specific examples, Algorithms for Learning Efficiently with Constrained Optimization techniques (ALECO) have been proposed, which incorporate information about the desirable behavior of hidden units. These algorithms exhibit better learning

Requests for reprints should be sent to Dr. Stavros J. Perantonis, Institute of Informatics and Telecommunications, National Research Center "Demokritos," 153 10 Aghia Paraskevi, Athens, Greece; E-mail: sper@iit.nrcps.ariadne-t.gr

properties than the BP algorithm and variants thereof (Karras & Perantonis, 1993; Perantonis & Karras, 1993; Varoufakis, Perantonis, & Karras, 1993; Karras, Perantonis, & Varoufakis, 1993, 1994).

Among this multitude of learning algorithms, back propagation with momentum acceleration (BPMA) (Rumelhart et al., 1986a,b) remains one of the most popular learning paradigms for MFNNs, mainly because of its faster convergence than the BP method in a variety of problems and because of its computational simplicity. The incorporation of momentum in the BP algorithm has been extensively studied, especially from an experimental point of view (Fahlman, 1988; Tesauro & Janssens, 1988; Jacobs, 1988; Minai & Williams, 1990; Tollenaere, 1990). It is only recently, however, that some theoretical background to this intrinsically heuristic method has been provided (Sato, 1991; Hagiwara, 1992).

The purpose of this paper is to establish a link between the use of momentum in MFNN learning on the one hand, and constrained optimization learning techniques on the other. Motivated by the BPMA algorithm, we discuss how the use of momentum can be optimized using constrained learning techniques. A modified algorithm for constrained learning with momentum (ALECO-2) ensues with substantially improved learning capabilities compared not only to the BPMA algorithm, but also to other popular and reputedly fast learning algorithms (quickprop and delta-bar-delta) in a variety of binary benchmark problems.

This paper is organized as follows. In Section 2, the BPMA formalism is reviewed and its links to constrained learning are discussed. In Section 3, the new constrained learning algorithm with momentum is derived. Sections 4, 5, and 6 contain experimental work. In particular, Section 4 describes the experiments conducted to test the performance of the algorithm and compare it with that of other supervised learning algorithms; experimental results are presented in Section 5 and discussed in Section 6. Finally, in Section 7, conclusions are drawn and future research goals are set.

2. LEARNING WITH MOMENTUM ACCELERATION

Consider the standard MFNN architecture with one layer of input, M layers of hidden, and one layer of output nodes. The nodes in each layer receive input from all nodes in the previous layer. The network node outputs are denoted by $O_{ip}^{(m)}$. Here the superscript (m) labels a layer within the structure of the neural network ($m = 0$ for the input layer, $m = k$ for the kth hidden layer, $m = M + 1$ for the output layer), i labels a node within a layer, and p labels the input patterns. The synaptic weights are denoted by $w_{ij}^{(m)}$, where m, j correspond, respectively, to the layer and the node toward which the synapse is directed, and i corresponds to the node in the previous layer from which the synapse emanates. Keeping in mind the iterative nature of learning algorithms, we shall denote the value of node outputs and weights at the current epoch and at the last (immediately preceding) epoch by the subscripts c and l, respectively.

The ultimate goal of a supervised learning algorithm, viz. matching the network outputs to prespecified target values T_{ip}, can be achieved through minimization of the cost function

$$E = \tfrac{1}{2} \sum_{ip} \varepsilon_{ip},\ \varepsilon_{ip} = (T_{ip} - O_{ip}^{(M+1)})^2. \tag{1}$$

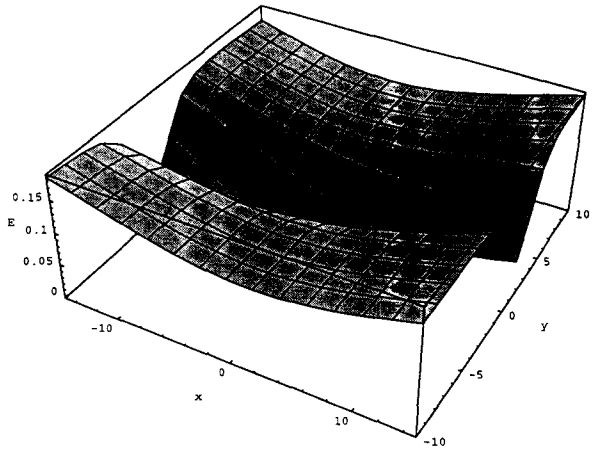
In the BPMA algorithm (off-line version) minimization of E is attempted using the following rule for updating the weights:

$$dw_{ij}^{(m)} = -\varepsilon \left. \frac{\partial E}{\partial w_{ij}^{(m)}} \right|_c + \alpha (w_{ij}^{(m)}|_c - w_{ij}^{(m)}|_l). \tag{2}$$

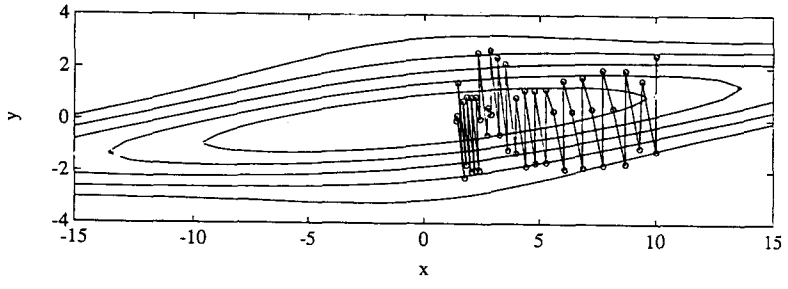
Thus, the current weight update vector is a linear combination of the gradient vector and the weight update vector in the immediately preceding epoch.

The BPMA algorithm is inherently heuristic in nature, although attempts have been made to invest it with theoretical background by taking into account information from the behavior of the weights in more than one epoch (Sato, 1991; Hagiwara, 1992). Thus, in BPMA the mathematical rigor of gradient descent—where a lot of information is available in the form of convergence theorems (Goldstein, 1965; Fletcher, 1980)—is compromised; in return, it is expected that improved speed can be achieved by filtering out high-frequency variations of the error surface in the weight space (Rumelhart et al., 1986a).

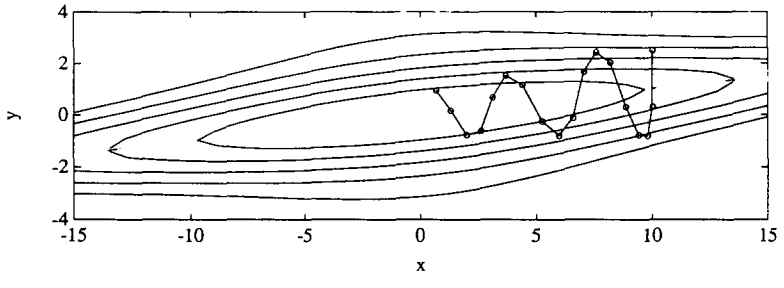
A good example of relatively successful negotiation of high-frequency variations by BPMA is movement along long narrow troughs that are flat in one direction and steep in surrounding directions. These features are often exhibited by cost function landscapes in various small- and large-scale problems solved by MFNN (Sutton, 1986; Hush, Horne, & Salas, 1992). In such landscapes, the cost function exhibits significant eccentricity and high-frequency variation is present in the direction perpendicular to that of the trough. It is well known that gradient descent proper is highly inefficient in locating minima in such landscapes (Rao, 1984) because it settles into zigzag paths and is hopelessly slow. In neural network applications, failure to converge to the global minimum can sometimes be attributed to zigzag wandering in the bottom of very shallow, steep-sided valleys (Hertz, Krogh, & Palmer, 1991). An illustrative example of such undesirable behavior is given by Hush et al., (1992). Supplementing gradient descent with momentum acceleration represents a compromise between the need to decrease the cost function at each epoch and the need to proceed along relatively



(a)



(b)



(c)

FIGURE 1. (a) Cost function landscape with a long, narrow trough. (b) Contour plot of the cost function and zigzag path followed by BP, which reaches the minimum in 45 epochs. (c) Smoother path followed by BPMA, reaching the minimum in 17 epochs. Initial conditions and algorithm parameters are given in the text.

smooth paths in the weight space. The formalism favors configurations where the current and previous weight update vectors are partially aligned, thus avoiding zigzag paths and accelerating learning.

It is instructive to provide visual evidence of the improvement achieved by incorporating momentum in the BP formalism. This is possible in simple two-dimensional problems. Consider, for example, a network with two input nodes, one layer of weights, and one output node without bias, corresponding to the following cost function E of the weights x and y:

$$E(x, y) = \tfrac{1}{2}[g(ax + by) - T_1]^2 + \tfrac{1}{2}[g(cx + dy) - T_2]^2. \quad (3)$$

Here g is the logistic function $g(x) = 1/(1 + \exp(-x))$. The values $a = -0.1$, $b = -0.02$, $c = 0.1$, $d = -1.0$, $T_1 = 0.5$, $T_2 = 0.5$ are chosen to create a

long trough with the minimum at $x = 0, y = 0$, as shown in Figure 1. The objective is to reach the minimum starting from the initial conditions $x_0 = 10.0$, $y_0 = 2.5$ within a tolerance of 10^{-3} for E. Gradient descent with relatively low values of ε is hopelessly slow in the trough, whereas best performance is achieved with large values of ε, leading to zigzag paths. Figure 1a shows the path obtained with $\varepsilon = 76.0$, which reaches the minimum in 45 epochs. Using momentum acceleration leads to partial alignment of successive weight update vectors and to a smoother path that follows the direction of the trough more closely. As a result, faster convergence is achieved, as shown in Figure 1b, where the minimum is reached in 17 epochs with $\varepsilon = 24.0$, $\alpha = 0.9$.

Motivated by the analysis of BPMA made so far, we suggest that still better results could be obtained using an iterative algorithm that would maximize the alignment of successive weight update vectors without compromising the need for a decrease of the cost function at each epoch. This would allow more efficient negotiation of cost function landscapes involving long, steep-sided troughs. Thus, the proposed algorithm (ALECO-2) should solve *for each epoch* the following constrained optimization problem:

- *Maximize* the function

$$\Phi = \sum_{ijm} (w_{ij}^{(m)} - w_{ij}^{(m)}|_c)(w_{ij}^{(m)}|_c - w_{ij}^{(m)}|_l) \qquad (4)$$

  to achieve optimal alignment of successive weight vector updates.
- Lower the cost function E by a specified amount δE. After a sufficient number of epochs, the accumulated changes to the non-negative cost function should suffice to achieve the desired input-output relation.

The proposed algorithm is an iterative procedure whereby the weights are changed by small amounts $dw_{ij}^{(m)}$ at each iteration so that the quadratic form

$$\sum_{ijm} dw_{ij}^{(m)} \cdot dw_{ij}^{(m)} \qquad (5)$$

takes on a prespecified value $(\delta P)^2$. Thus, at each epoch, the search for an optimum new point in the weight space is restricted to a small hypersphere centered at the point defined by the current weight vector. If δP is small enough, the changes to E and Φ induced by changes in the weights can be approximated by the first differentials dE and $d\Phi$. The problem then amounts to determining, for given values of δP and δE, the values of $dw_{ij}^{(m)}$, so that the maximum value of $d\Phi$ is attained. Similar problems where Φ has an explicit functional dependence on the node activations only (not the weights, as is the case here) have been solved (Karras & Perantonis, 1993; Perantonis & Karras, 1993; Varoufakis et al., 1993; Karras et al., 1993, 1994) by closely following the optimal control method proposed by Bryson and Denham (1962). In this case, where Φ exhibits an explicit functional dependence on the weights, a modification of this method is required. The solution, based on methods of nonlinear programming, is presented in the next section.

3. DERIVATION OF ALECO-2

Maximization of $d\Phi$ is attempted with respect to variations in $w_{ij}^{(m)}$ and $O_{ip}^{(m)}$. In the language of nonlinear programming, the synaptic weights correspond to decision variables and the node outputs correspond to state (solution) variables (Beightler, Phillips, & Wilde, 1979). These quantities must satisfy the state equations, that is, the constraints describing the network architecture

$$f_{jp}^{(m)}(\boldsymbol{O}, \boldsymbol{w}) = g\left(\sum_i w_{ij}^{(m)} O_{ip}^{(m-1)}\right) - O_{jp}^{(m)} = 0. \qquad (6)$$

Here g is the logistic function $g(x) = 1/(1 + \exp(-x))$ and biases are treated as weights emanating from nodes of constant, pattern-independent activation equal to 1. In addition, the following two constraints must be satisfied:

$$dE = \delta E \qquad (7)$$

$$\sum_{ijm} dw_{ij}^{(m)} \cdot dw_{ij}^{(m)} = (\delta P)^2. \qquad (8)$$

This constrained maximization problem is solved by introducing suitable Lagrange multipliers. Hence, to take account of the architectural constraints, we construct the functions

$$e = E + \sum_{jpm} \lambda_E^{jp(m)} f_{jp}^{(m)} \qquad (9)$$

$$\phi = \Phi + \sum_{jpm} \lambda_\Phi^{jp(m)} f_{jp}^{(m)} \qquad (10)$$

where the λ_E and λ_Φ are Lagrange multipliers to be determined in due course. Consider the differentials

$$de = \sum_{jpm} \left.\frac{\partial e}{\partial O_{jp}^{(m)}}\right|_c dO_{jp}^{(m)} + \sum_{ijm} \left.\frac{\partial e}{\partial w_{ij}^{(m)}}\right|_c dw_{ij}^{(m)} \qquad (11)$$

$$d\phi = \sum_{jpm} \left.\frac{\partial \phi}{\partial O_{jp}^{(m)}}\right|_c dO_{jp}^{(m)} + \sum_{ijm} \left.\frac{\partial \phi}{\partial w_{ij}^{(m)}}\right|_c dw_{ij}^{(m)}. \qquad (12)$$

We choose the λ_E and λ_Φ to eliminate all dependence of de and $d\phi$ on the $O_{jp}^{(m)}$:

$$\left.\frac{\partial e}{\partial O_{jp}^{(m)}}\right|_c = 0, \quad \left.\frac{\partial \phi}{\partial O_{jp}^{(m)}}\right|_c = 0. \qquad (13)$$

This leads to closed formulas for determining the Lagrange multipliers. From eqns (1), (4), (6), (9), (10), and (13) we readily obtain

$$\lambda_E^{jp(M+1)} = O_{jp}^{(M+1)}|_c - T_{jp} \qquad (14)$$

$$\lambda_E^{ip(m)} = \sum_j \lambda_E^{jp(m+1)} w_{ij}^{(m+1)} O_{jp}^{(m+1)}|_c (1 - O_{jp}^{(m+1)}|_c),$$

$$m = 1, 2, \ldots, M \qquad (15)$$

$$\lambda_{\Phi}^{jp(m)} = 0, \quad m = 1, 2, \ldots, M+1 \tag{16}$$

for all nodes j and patterns p.

The Lagrange multipliers can thus be determined in the following systematic way. Multipliers corresponding to the output layer are evaluated. Multipliers of the mth layer are readily determined once the ones corresponding to the $(m+1)$th layer have been evaluated. This procedure can be considered as a back propagation of the Lagrange multiplier values.

Differentiating eqns (9) and (10) with respect to the synaptic weights and having eliminated all dependence on the state variables, we obtain the following equations for points satisfying the architectural constraints:

$$dE = de = \sum_{ijm} J_{ijm} dw_{ij}^{(m)}, \quad d\Phi = d\phi = \sum_{ijm} F_{ijm} dw_{ij}^{(m)} \tag{17}$$

with

$$J_{ijm} = \sum_{p} \lambda_E^{jp(m)} O_{jp}^{(m)}|_c (1 - O_{jp}^{(m)}|_c) O_{ip}^{(m-1)}|_c \tag{18}$$

$$F_{ijm} = \left.\frac{\partial \Phi}{\partial w_{ij}^{(m)}}\right|_c = w_{ij}^{(m)}|_c - w_{ij}^{(m)}|_l. \tag{19}$$

We now introduce new Lagrange multipliers λ_1 and λ_2 to take account of the remaining constraints in the problem [eqns (7) and (8)]

$$d\phi = d\Phi = \sum_{ijm} F_{ijm} dw_{ij}^{(m)} + \lambda_1 \left(\delta E - \sum_{ijm} J_{ijm} dw_{ij}^{(m)} \right) + \lambda_2 \left[(\delta P)^2 - \sum_{ijm} dw_{ij}^{(m)} dw_{ij}^{(m)} \right]. \tag{20}$$

Note that the quantities multiplying λ_1 and λ_2 are equal to zero by eqns (17) and (8) and that δP and δE are known quantities. We obtain maximum change in Φ at each iteration of the algorithm by ensuring that

$$d^2\Phi = \sum_{ijm} (F_{ijm} - \lambda_1 J_{ijm} - 2\lambda_2 dw_{ij}^{(m)}) d^2 w_{ij}^{(m)} = 0 \tag{21}$$

$$d^3\Phi = -2\lambda_2 \sum_{ijm} d^2 w_{ij}^{(m)} \cdot d^2 w_{ij}^{(m)} < 0. \tag{22}$$

To satisfy eqn (21) we set

$$dw_{ij}^{(m)} = -\frac{\lambda_1}{2\lambda_2} J_{ijm} + \frac{1}{2\lambda_2} F_{ijm}. \tag{23}$$

In effect, weight updates are formed at each epoch as a linear combination of the cost function derivative J_{ijm} with respect to the corresponding weight [see eqn (17)] and of the weight update F_{ijm} at the immediately preceding epoch [see eqn (19)]. This weight update rule is similar to that of BPMA. However, unlike BPMA, where the coefficients of F_{ijm} and J_{ijm} are constant, in ALECO-2 the coefficients are chosen adaptively. To see this, we use eqns (8), (21), and (17) to obtain

$$\lambda_2 = \frac{1}{2}\left[\frac{I_{EE}(\delta P)^2 - (\delta E)^2}{I_{\Phi\Phi} I_{EE} - I_{E\Phi}^2} \right]^{-1/2},$$

$$\lambda_1 = (I_{E\Phi} - 2\lambda_2 \delta E)/I_{EE} \tag{24}$$

where

$$I_{EE} = \sum_{ijm} (J_{ijm})^2 \tag{25}$$

$$I_{E\Phi} = \sum_{ijm} J_{ijm} F_{ijm} \tag{26}$$

$$I_{\Phi\Phi} = \sum_{ijm} (F_{ijm})^2 \tag{27}$$

and the positive square root value was chosen for λ_2 to ensure maximum (rather than minimum) $d\Phi$ [relation (22)].

Note the bound $|\delta E| \leq \delta P \sqrt{I_{EE}}$ set on the value of δE by eqn (24), which forces us to choose δE adaptively. The simplest choice for adapting δE, namely

$$\delta E = -\xi \delta P \sqrt{I_{EE}}, \quad 0 < \xi < 1 \tag{28}$$

is most attractive because of its learning convergence properties. Indeed, as in BP, it is possible to show for small enough δP that the algorithm converges to global or local minima of the cost function of eqn (1). To see this, we use eqns (25), (11), and (17) to rewrite eqn (28) as

$$\delta E = -\xi \delta P \left[\sum_{ijm} (\partial e / \partial w_{ij}^{(m)})^2 \right]^{1/2}. \tag{29}$$

Hence, it suffices to show that for a given positive real number η there exists an epoch number ν_0, so that $|\partial e/\partial w_{ij}^{(m)}| < \eta$ for all subsequent epochs. Indeed, in the opposite case, an infinity of changes in E at least equal to $-\xi\eta\delta P$ would accumulate and drive E to minus infinity as learning progressed. This is not possible, because E is bounded from below by zero.

In short, eqn (23) describes the weight updating formula of ALECO-2, which optimizes the weight steps in each epoch and converges to minima of the cost function. Taking into account eqn (28) we are left with two free learning parameters δP and ξ, which should be adjusted to achieve optimum performance.

Before testing the performance of ALECO-2 on specific benchmark tests, we return to the two-dimensional cost function of eqn (3) and illustrate in Figure 2 the ability of ALECO-2 to negotiate cost function landscapes with long, steep-sided troughs. The behavior of ALECO-2 in the landscape of Figure 1 is shown, starting from the same initial conditions $x_0 = 10.0$, $y_0 = 2.5$. The learning parameters $dP = 1.1$ and $\xi = 0.3$ were used. Note that optimal alignment of successive weight update vectors leads to a path that is smoother and more closely aligned to the axis of the trough than gradient descent (with or without momentum). As a result, the minimum is reached within the allowed tolerance in just 10 epochs.

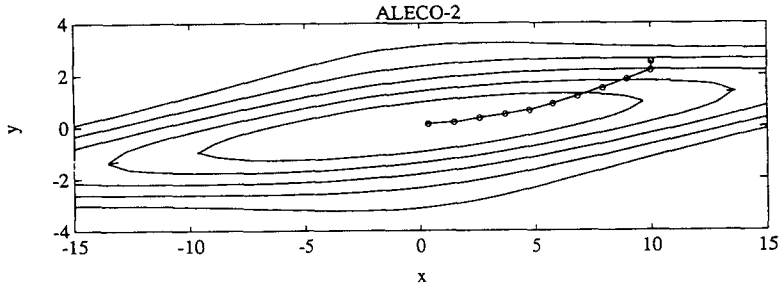


FIGURE 2. Path followed by ALECO-2 in the cost function landscape of Figure 1a, with the same initial conditions as in Figure 1b and c. The minimum is reached in just 10 epochs. Algorithm parameters are given in the text.

4. ORGANIZATION OF EXPERIMENTS

In evaluating the performance of a learning algorithm, three key issues should be addressed: the learning speed of the algorithm; its scalability properties (i.e., its ability to cope with large network architectures and problems with a large number of training patterns); and the generalization capabilities of networks trained using this algorithm. In the remaining sections of this paper we present experimental work carried out to test learning speed and scalability properties, and comment briefly on generalization properties, a full study of which is deferred for papers currently under preparation.

We compare ALECO-2 not only with BPMA (both the off-line version referred to in the previous sections of this work, as well as the well-known on-line version), but also with other supervised learning algorithms that reputedly improve its performance significantly. We have selected some representative, well-known, and reputedly fast algorithms that have fixed and predefined model architecture—so we have excluded algorithms like cascade correlation (Fahlman & Lebiere, 1990). These algorithms are the quickprop learning rule (Fahlman, 1988) and delta-bar-delta (Jacobs, 1988). Our simulation program is based on the one referred by Pao (1989). We also used the public domain quickprop simulation program created by Fahlman in Common Lisp and translated into C by Regier.

There is no wide consensus about benchmark selection for testing learning speed and scalability. Nevertheless, binary benchmark problems—encoder, counter (van Ooyen & Nienhuis, 1992), multiplexer (Jacobs, 1988)—are the ones that most authors are in favor of. Among them, encoder problems are used by an increasing number of authors (Fahlman, 1988; Schmidhuber, 1988; Tollenaere, 1990; van Ooyen & Nienhuis, 1992) as seeming to be acceptable in terms of generalization. XOR, the well known and popular benchmark, is also frequently used for historical reasons (Minsky & Papert, 1969). Thus, in our experiments we use XOR and rather small-size counter (4-5-5 architecture), multiplexer (6-6-1), and encoder (8-3-8) problems to test learning speed. In general, these are the benchmarks used by the authors of the algorithms to which we compare ALECO-2. In this way we ensure performance evaluation of ALECO-2 in the environment proved to be best for the other algorithms. We also use large-scale benchmarks with either a large number of synaptic weights and biases (64-6-64 and 256-8-256 encoders) or a large number of patterns in the training set (11-11-1 multiplexer with 2048 input patterns) to test scalability properties. For the small-scale benchmarks we ran our program on the small SUN-sparcstation network of our laboratory, whereas for the large-scale benchmarks we used the Convex supercomputer and the Silicon Graphics Crimson workstation at NRCPS "DEMOKRITOS."

For reasons of uniformity and fair algorithm comparison, the following factors were used for all algorithms. Random initial weights with uniform distribution in $[-0.5,0.5]$ were selected. The cost function of eqn (1) was used. Patterns were presented to the networks in a fixed order, sequentially indexed in their categories. The sigmoid-logistic function $g(x) = 1/(1 + \exp(-x))$ was used as activation function in our MFNN. We adopted Fahlman's suggestion of adding to the derivative of the logistic function a small constant S'. This was found to accelerate all five algorithms. Finally, we adopted the 0.4–0.6 convergence criterion reported by Fahlman (1988) for all algorithms and benchmarks.

Each algorithm has its own learning parameters and a fair comparison requires that the best parameter values be chosen for each algorithm and benchmark (Tollenaere, 1990; van Ooyen & Nienhuis, 1992). The learning parameters of the algorithms we have compared in this experimental study, as defined by the authors who proposed them, are:

- *BPMA:* the learning rate ε, the momentum factor α (Rumelhart et al., 1986a,b).
- *Quickprop:* the learning rate ε, the momentum factor α, the maximum growth factor μ, and the weight decay term ω (Fahlman, 1988).
- *Delta-Bar-Delta:* the learning rate ε, the learning rate increment κ, the learning rate proportion decrement φ, the base of the exponential average of the

TABLE 1
Experimental Results for the 4-5-5 Counter Problem

Experiment	Type: COUNTER		Categories: 5	Input Samples: 16	Ref: van Ooyen & Nienhuis, 1992
MFNN System Architecture	Input Units: 4		Hidden Layers: 1	Hidden Units: 5	Output Units: 5
Algorithms	ALECO-2	On BP	Off BP	Quickprop	Delta-Bar-Delta
Learning Parameters	$\delta P = 1.2$ $\xi = 0.85$	$\varepsilon = 0.50$ $\alpha = 0.70$	$\varepsilon = 0.50$ $\alpha = 0.5$	$\varepsilon = 3.00$, $\mu = 1.75$ $\omega = -0.0001$, $\alpha = 0.0$	$\varepsilon = 0.20$, $\kappa = 0.09$ $\varphi = 0.12$, $\theta = 0.70$ $\alpha = 0.70$
Allowed Epochs	3000	3000	3000	3000	3000
Trials	1000	1000	1000	1000	1000
Successes (%)	86.4	85.2	90.2	92.5	89.0
Failures (%)	13.6	14.8	9.8	7.5	11.0
Mean	**97.45**	**247.57**	**520.10**	**603.18**	**227.92**
Stdv	111.25	76.25	504.28	517.51	241.86
Maximum	1740	992	2965	2921	2053
Minimum	**46**	**155**	**148**	**78**	**53**

derivatives θ, and the momentum factor α (Jacobs, 1988).

- *ALECO-2:* the parameters δP and ξ defined in Section 3.

For each algorithm, the relevant parameters are carefully adjusted to achieve the best possible performance. Results are considered optimum when small minima and averages for the number of epochs needed to complete each benchmark task are obtained, subject to the condition that at least 70% of the experimental trials have passed the 0.4–0.6 convergence criterion.

In MFNN training algorithms, trials are started by random initialization of the weights. The search for a minimum varies in terms of difficulty at each trial. Therefore, it is very important to test our algorithms for a sufficient number of trials to ensure an adequate level of statistical significance for our results and a fair comparison of different training algorithms.

In this work, a test of statistical significance was performed on the average number of epochs needed to successfully complete a training task. It should be emphasized that for a specific training task the number of successful training trials needed to obtain a certain level of statistical significance for this average can depend heavily on the training algorithm. We denote by $\mathcal{M}$ and σ the experimental sample average and standard deviation of the number of epochs in successful trials, respectively. An estimate of the minimum number ν of training epochs needed to obtain an experimental average differing from the true average of the distribution less than a fraction γ of $\mathcal{M}$ with probability greater than b can be obtained using the Central Limit Theorem of Probability Theory (Papoulis, 1965). An estimate proportional to $(\sigma/\mathcal{M})^2$ is obtained:

$$\nu \approx \left(\frac{\sigma}{\mathcal{M}}\right)^2 \frac{2}{\gamma^2} [\mathrm{erf}^{-1}(b)]^2,$$

$$\text{where} \quad \mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-u^2)\, du. \quad (30)$$

A short derivation of this formula is given in the Appendix. Because different training algorithms can yield significantly differing values of $(\sigma/\mathcal{M})^2$, the number of trials needed to obtain a certain level of statistical significance will vary from algorithm to algorithm for the same benchmark. Taking into account eqn (30), we have performed enough experimental trials, using new initial weights, for each learning algorithm and benchmark to ensure at least 99% probability that the experimental average number of epochs in successful trials differed from the true average by less than 10%.

5. PRESENTATION OF EXPERIMENTAL RESULTS

Complete results about the performance of all algorithms are reported in Tables 1–7, each table corresponding to one of the benchmarks. In each table sufficient information is given to allow other researchers to reproduce our results (taking into account the factors common for all algorithms and benchmarks reported in Section 4). Thus, a brief benchmark description is given (benchmark identification, number of patterns and categories, reference where a full description can be found); the architecture of the MFNN that is called upon to solve the benchmark is described; the learning parameters used for each algorithm are reported; the maximum allowed number of epochs before declaration of failure is shown; finally, the number of trials performed for each algorithm, starting from different initial weights, is given.

TABLE 2
Experimental Results for the 6-6-1 Multiplexer Problem

Experiment	Type: ENCODER		Categories: 8	Input Samples: 8	Ref: Fahlman, 1988
MFNN System Architecture	Input Units: 8		Hidden Layers: 1	Hidden Units: 3	Output Units: 8
Algorithms	ALECO-2	On BP	Off BP	Quickprop	Delta-Bar-Delta
Learning Parameters	$\delta P = 2.0$ $\xi = 0.75$	$\varepsilon = 2.50$ $\alpha = 0.50$	$\varepsilon = 1.50$ $\alpha = 0.50$	$\varepsilon = 5.00$, $\mu = 1.75$ $\omega = -0.0001$, $\alpha = 0.0$	$\varepsilon = 0.50$, $\kappa = 0.45$ $\varphi = 0.15$, $\theta = 0.40$ $\alpha = 0.80$
Allowed Epochs	1000	1000	1000	1000	1000
Trials	1000	1000	1000	1000	1000
Successes (%)	100.0	100.0	100.0	100.0	99.9
Failures (%)	0.0	0.0	0.0	0.0	0.1
Mean	**37.81**	**172.06**	**145.77**	**83.26**	**68.34**
Stdv	15.58	79.50	53.62	97.51	64.08
Maximum	120	517	370	946	941
Minimum	**17**	**25**	**56**	**16**	**25**

Detailed results are shown, including the percentage of successful and unsuccessful trials, as well as the mean, maximum, minimum, and standard deviation of the number of epochs required to solve the problems for the selected number of trials with different weight initializations. In principle, learning speed can be evaluated using either the concept of epoch, which is widely accepted, or similar ones (Fahlman, 1988) or the concept of computational complexity as partially involved in the work of Shah, Palmieri, and Datum (1992). Here we have chosen a scheme based on the number of epochs, rather than computational complexity, motivated by the following considerations:

1. The epoch, apart from being a convenient and generally accepted unit for measuring learning time (Fahlman, 1988), is also compatible with the concept of the "100-step program" constraint (Feldman, 1985) extended in the training procedure. In biological learning, the number of "training set" presentations involved in a learning task is a small number, rather than a number of the order of thousands. Moreover, we can assume that whenever exponential time is involved in computations within the same epoch, neural networks (the biological ones, as well as the models we investigate) may be able to provide speed at the cost of an excessive network size (Abu-Mostafa, 1986).
2. Nevertheless, implementation of current generation neural networks on serial computers calls for algorithms that can afford us with acceptable overall training times. In this respect, it is important to note

TABLE 3
Experimental Results for the 8-3-8 Encoder Problem

Experiment	Type: MULTIPLEXER		Categories: 2	Input Samples: 64	Ref: Jacobs, 1988
MFNN System Architecture	Input Units: 6		Hidden Layers: 1	Hidden Units: 6	Output Units: 1
Algorithms	ALECO-2	On BP	Off BP	Quickprop	Delta-Bar-Delta
Learning Parameters	$\delta P = 1.8$ $\xi = 0.85$	$\varepsilon = 1.50$ $\alpha = 0.0$	$\varepsilon = 0.30$ $\alpha = 0.80$	$\varepsilon = 0.5$, $\mu = 2.5$ $\omega = -0.0001$, $\alpha = 0.0$	$\varepsilon = 0.20$, $\kappa = 0.09$ $\varphi = 0.12$, $\theta = 0.50$ $\alpha = 0.80$
Allowed Epochs	2000	2000	2000	2000	2000
Trials	2000	1000	1000	1000	2000
Successes (%)	97.4	99.6	84.0	96.1	82.1
Failures (%)	2.6	0.4	16.0	3.9	17.9
Mean	**88.37**	**166.47**	**124.33**	**236.04**	**135.76**
Stdv	138.09	74.74	114.66	230.60	204.72
Maximum	1971	1145	996	1968	1959
Minimum	**23**	**86**	**41**	**48**	**28**

TABLE 4
Experimental Results for the XOR Problem

Experiment	Type: XOR		Categories: 2	Input Samples: 4	Ref: Minsky & Papert, 1969
MFNN System Architecture	Input Units: 2		Hidden Layers: 1	Hidden Units: 2	Output Units: 1
Algorithms	ALECO-2	On BP	Off BP	Quickprop	Delta-Bar-Delta
Learning Parameters	$\delta P = 1.4$ $\xi = 0.85$	$\varepsilon = 0.90$ $\alpha = 0.70$	$\varepsilon = 4.50$ $\alpha = 0.80$	$\varepsilon = 2.00$, $\mu = 1.20$ $\omega = -0.0001$, $\alpha = 0.0$	$\varepsilon = 7.0$, $\kappa = 0.25$ $\varphi = 0.12$, $\theta = 0.70$ $\alpha = 0.80$
Allowed Epochs	2000	2000	2000	2000	2000
Trials	5000	2000	1000	5000	1000
Successes (%)	80.5	73.5	91.9	80.2	92.7
Failures (%)	19.5	26.5	8.1	19.8	7.3
Mean	**48.37**	**175.12**	**73.88**	**130.60**	**66.02**
Stdv	93.77	215.96	79.16	272.95	72.59
Maximum	981	1978	1176	1999	1245
Minimum	**8**	**24**	**25**	**26**	**26**

that the CPU time required to complete an epoch is similar for all algorithms quoted in this paper. Indeed, all these algorithms require the evaluation of the derivatives J_{ijm} of the cost function with respect to the weights. The number of operations per weight needed to complete this evaluation is proportional to the number of patterns in the training set [cf. eqn (18)]. Once the J_{ijm} have been evaluated, only a relatively small number of additional operations per weight independent of the number of training patterns is needed to complete an update [for ALECO-2 cf. eqns (19), (23)–(27)]. Although this additional computational burden is less for off-line BPMA than all other algorithms quoted here, it takes much less CPU time than the calculation of the J_{ijm} in all these algorithms. This is confirmed by the actual CPU times measured in our binary benchmarks. Utilizing optimized codes for use with a serial computer, the following ratios of CPU times needed to complete an epoch in ALECO-2 and off-line BPMA have been measured on a SUN-sparc2 workstation, accurate to two decimal places: 1.06 for the 4-5-5 counter, 1.02 for the 6-6-1 multiplexer, 1.09 for the 8-3-8 encoder, 1.17 for XOR, 1.02 for the 64-6-64 encoder, 1.00 for the 256-8-256 encoder, and 1.00 for the 11-11-1 multiplexer.

In the literature, there exist two approaches for reporting the number of epochs required by learning algorithms to complete training tasks, reported by Jacobs (1988) and Fahlman (1988), respectively. The first approach favors algorithms that give better mean value of epochs whereas the second, using the restart procedure, favors algorithms that give better minima of the number of epochs. Which of these is a better measure of learning speed performance depends heavily on the shape of the distribution of epochs needed to success-

TABLE 5
Experimental Results for the 64-6-64 Encoder Problem

Experiment	Type: ENCODER		Categories: 64	Input Samples: 64	Ref: Fahlman, 1988
MFNN System Architecture	Input Units: 64		Hidden Layers: 1	Hidden Units: 6	Output Units: 64
Algorithms	ALECO-2	On BP	Off BP	Quickprop	Delta-Bar-Delta
Learning Parameters	$\delta P = 1.5$ $\xi = 0.50$	$\varepsilon = 0.50$ $\alpha = 0.70$	No convergence	$\varepsilon = 4.00$, $\mu = 1.75$ $\omega = -0.0001$, $\alpha = 0.0$	No convergence
Allowed Epochs	1000	1000	1000	1000	1000
Trials	100	100	100	100	100
Successes (%)	100.0	100.0	0.0	98.0	0.0
Failures (%)	0.0	0.0	100.0	2.0	100.0
Mean	**158.72**	**276.83**		**402.09**	
Stdv	13.94	35.80		143.97	
Maximum	202	412		984	
Minimum	**128**	**213**		**124**	

TABLE 6
Experimental Results for the 256-8-256 Encoder Problem

Experiment	Type: ENCODER		Categories: 256	Input Samples: 256	Ref: Fahlman, 1988
MFNN System Architecture	Input Units: 256		Hidden Layers: 1	Hidden Units: 8	Output Units: 256
Algorithms	ALECO-2	On BP	Off BP	Quickprop	Delta-Bar-Delta
Learning Parameters	$\delta P = 1.8$ $\xi = 0.50$	$\varepsilon = 0.40$ $\alpha = 0.60$	No convergence	No convergence	No convergence
Allowed Epochs	1000	1000	1000	1000	1000
Trials	50	50	50	50	50
Successes (%)	100.0	100.0	0.0	0.0	0.0
Failures (%)	0.0	0.0	100.0	100.0	100.0
Mean	**293.07**	**476.68**			
Stdv	15.88	89.96			
Maximum	323	699			
Minimum	**257**	**358**			

fully complete a task. For example, it is clearly desirable that an algorithm exhibit a small standard deviation to mean value ratio for this distribution; it is then reliable in its performance as regards learning speed and the mean value is an adequate parameter by which to judge performance. On the other hand, if this ratio is large, the distribution will extend far beyond its peak towards infinity and the peak can be closer to the minimum number of epochs than to the mean; this minimum may then be considered as a better measure of learning speed, provided that the user of the algorithm is prepared to disregard the trouble caused by trials in which an excessive number of epochs is needed to achieve convergence. With these thoughts in mind, we try to be as informative as possible in reporting our results and include in Tables 1–7 detailed information about the distribution of epochs required to successfully complete each task. The mean and minimum of the distribution of epochs are highlighted, to help the reader focus his/her attention.

6. DISCUSSION OF EXPERIMENTAL RESULTS

Evidently, ALECO-2 outperforms off-line BPMA in all four small-scale benchmarks in terms of learning speed. Ratios of the average number of epochs needed to solve the tasks using the two algorithms range from 5.34 in the 4-5-5 counter problem to 1.41 in the 6-6-1 multiplexer problem, always in favor of ALECO-2. The corresponding ratios for the minimum number of epochs needed to solve these tasks range from 3.29 in the 8-3-8 encoder to 1.78 in the 6-6-1 multiplexer. Moreover, our method also achieves much faster learning than online BPMA, quickprop, and delta-bar-delta: much better averages are obtained than both algorithms for all

TABLE 7
Experimental Results for the 11-11-1 Multiplexer Problem

Experiment	Type: MULTIPLEXER		Categories: 2	Input Samples: 2048	Ref: Jacobs, 1988
MFNN System Architecture	Input Units: 11		Hidden Layers: 1	Hidden Units: 11	Output Units: 1
Algorithms	ALECO-2	On BP	Off BP	Quickprop	Delta-Bar-Delta
Learning Parameters	$\delta P = 0.8$ $\xi = 0.50$	$\varepsilon = 0.60$ $\alpha = 0.70$	No convergence	No convergence	No convergence
Allowed Epochs	1000	1000	1000	1000	1000
Trials	100	100	100	100	100
Successes (%)	100.0	94.0	0.0	0.0	0.0
Failures (%)	0.0	6.0	100.0	100.0	100.0
Mean	**140.25**	**279.68**			
Stdv	54.26	91.68			
Maximum	481	548			
Minimum	**82**	**160**			

benchmarks; the minimum number of epochs is much better than that achieved by delta-bar-delta and on-line BPMA in all cases and better or, at worst, comparable to that achieved by quickprop. Compared to the other methods regarding the percentage of successful trials, our method has mixed fortunes. It can be said that none of the algorithms examined shows a clear advantage over the others regarding convergence ability in the small-scale benchmarks.

Results in the three large-scale benchmarks can be summarized as follows:

- Concerning convergence ability, we notice that off-line BP and delta-bar-delta cannot converge at all in the benchmarks tried within the specified limit of epochs until failure, whereas quickprop converges only in the 64-6-64 encoder. By contrast, on-line BP and ALECO-2 exhibit good convergence ability in all three tasks tried, with ALECO-2 slightly outperforming on-line BP in the 11-11-1 multiplexer problem.
- Concerning learning speed, measured either by the average or the minimum of the distribution of epochs needed to successfully complete a task, ALECO-2 clearly outperforms its closest rival (on-line BP) in all the tasks by, approximately, a factor of 2.
- ALECO-2 exhibits a relatively small standard deviation in the distribution of epochs needed to successfully complete a task, thus exhibiting reliability of performance regarding learning speed.

These results are in compliance with work by Fogelman Soulie (1991) reporting that on-line BP exhibits very good learning abilities in large-scale problems. Moreover, they demonstrate the emergence of an excellent new learning algorithm, ALECO-2, for large-scale networks and problems.

Practical guidelines can be given for selecting optimal values for the learning parameters δP and ξ: for all small-scale benchmarks, similar performances were recorded with $0.5 < \xi < 0.9$ and $1.0 < \delta P < 2.0$, indicating that results are not very sensitive to the exact values of the parameters. Following the example of Jacobs (1988), we performed additional runs of ALECO-2 using common values ($\delta P = 1.5$ and $\xi = 0.85$) for all four small-scale benchmarks. Deterioration of the mean number of epochs, compared to the optimal values shown in Tables 1–4, was never more than 30%. Larger-scale problems are more sensitive to the selection of dP, but not to the selection of ξ for which a value around 0.5 works well for all benchmarks.

Finally, we add a few words regarding the generalization ability of ALECO-2. Preliminary results in large-scale optical character recognition problems indicate that the improvement in MFNN learning speed achieved by ALECO-2 has no adverse effect on generalization ability (Gatos, Karras, & Perantonis, 1993). In these problems, the classification accuracy achieved by ALECO-2 was superior to that achieved by on-line BPMA. We plan to carry out extensive tests of the generalization ability of ALECO-2 in OCR problems, to fully confirm these preliminary results. Moreover, in the same spirit of constrained learning, it is possible to augment the algorithm with weight elimination techniques (Weigend et al., 1991) that will hopefully further improve its generalization ability without adverse effect on its learning speed.

7. CONCLUSION AND PROSPECTS

A learning algorithm for MFNNs was proposed incorporating momentum acceleration in its formalism and exhibiting the following attractive features:

- Solid theoretical background, based on rigorous nonlinear programming techniques.
- Proved convergence to global or local minima of the mean squared error cost function for small enough learning step. This property is shared with the BP algorithm, but not with some of its descendants of heuristic origin.
- Faster learning than the BPMA algorithm, from which it is inspired, and from other reputedly fast learning algorithms.
- Good scalability properties.

However, the most attractive feature of this algorithm is its potential for further improvement. In the same framework for constrained learning, it is possible to augment it with further information about learning in MFNNs, including methods for constructing suitable internal representations (Karras & Perantonis, 1993; Karras et al., 1993) and weight elimination techniques. It is the concerted incorporation of such detailed information into the same algorithm that will hopefully lead to increasingly efficient MFNN training schemes combining fast learning, good scalability properties, and powerful generalization capabilities.

REFERENCES

Abu-Mostafa, Y. S. (1986). Neural networks for computing? *AIP Conference Proceedings 151,* Snowbird, UT, 1–6.

Becker, S., & le Cun, Y. (1988). Improving the convergence of back propagation learning with second-order methods. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), *Proceedings of the Connectionist Models Summer School* (pp. 29–37). San Mateo: Morgan Kaufmann.

Beightler, C. S., Phillips, D. T., & Wilde, D. J. (1979). *Foundations of optimization.* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall.

Bryson, A. E., & Denham, W. F. (1962). A steepest-ascent method for solving optimum programming problems. *Journal of Applied Mechanics,* **29,** 247–257.

Fahlman, S. E. (1988). Faster learning variations on back-propagation: An empirical study. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), *Proceedings of the Connectionist Models Summer School* (pp. 38–51). San Mateo: Morgan Kaufmann.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* (Vol. 2, pp. 524–532). San Mateo: Morgan Kaufmann.

Feldman, J. A. (1985). Connectionist models and their applications: Introduction. *Cognitive Science*, **9**, 1–2.

Fletcher, R. (1980). *Practical methods of optimization* (Vol. 1). New York: Wiley.

Fogelman Soulie, F. (1991). Neural network architectures and algorithms: A perspective. In T. Kohonen, K. Makisara, O. Simula, & J. Kangas (Eds.), *Artificial neural networks* (pp. 605–615). New York: Elsevier.

Funahashi, K-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, **2**, 183–192.

Gatos, B., Karras, D., & Perantonis, S. (1993). Optical character recognition using novel feature extraction and neural network classification techniques. *Proceedings of Workshop on Neural Networks: Techniques and Applications*, Liverpool, UK, 65–72.

Goldstein, A. A. (1965). On steepest descent. *Journal of SIAM Contributions Series A*, **3**(1), 147–151.

Grossman, T. (1990). The CHIR algorithm for feed forward networks with binary weights. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* (Vol. 2, pp. 516–523). San Mateo: Morgan Kaufmann.

Grossman, T., Meir, R., & Domany, E. (1990). Learning by choice of internal representations. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* (Vol. 1, pp. 73–80). San Mateo: Morgan Kaufmann.

Hagiwara, M. (1992). Theoretical derivation of momentum term in back propagation. *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, **1**, 682–686.

Hertz, J., Krogh, A., & Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Reading, MA: Addison–Wesley.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, 359–366.

Hush, D. R., Horne, B., & Salas, J. M. (1992). Error surfaces for multilayer perceptrons. *IEEE Transactions on Systems, Man, and Cybernetics*, **22**(5), 1152–1161.

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, **1**, 295–307.

Karras, D. A., & Perantonis, S. J. (1993). Efficient constrained training algorithm for feedforward networks. *IEEE Transactions on Neural Networks* (under revision).

Karras, D. A., Perantonis, S. J., & Varoufakis, S. J. (1993). Constrained learning: A new approach to pattern classification. *Proceedings of the World Conference on Neural Networks*, Oregon, USA, **4**, 235–238.

Karras, D. A., Perantonis, S. J., & Varoufakis, S. J. (1994). An efficient constrained learning algorithm for optimal linear separability of the internal representations. *Proceedings of World Congress on Computational Intelligence*, Orlando, USA, 285–289.

Kramer, A. H., & Sangiovanni–Vincentelli (1988). Efficient parallel learning algorithms for neural networks. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* (pp. 40–48). San Mateo: Morgan Kaufmann.

Krogh, A., Thorbergsson, G. I., & Hertz, J. A. (1990). A cost function for internal representations. In Touretzky, D. S. (Ed.), *Advances in neural information processing systems* (Vol. 2, pp. 733–740). San Mateo: Morgan Kaufmann.

Minai, A. A., & Williams, R. D. (1990). Back-propagation heuristics: A study of the extended delta-bar-delta algorithm. *Proceedings of the International Joint Conference on Neural Networks*, San Diego, **I**, 595–600.

Minsky, M., & Papert, S. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Pao, Y-H. (1989). *Adaptive pattern recognition and neural networks*. Reading, MA: Addison-Wesley.

Papoulis, A. (1965). *Probability, random variables, and stochastic processes*. Tokyo: Mc Graw-Hill Kogakusha.

Parker, D. B. (1987). Optimal algorithms for adaptive networks: Second order back propagation, second order direct propagation and second order Hebbian learning. *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, 593–600.

Perantonis, S. J., & Karras, D. A. (1993). A fast constrained learning algorithm based on the construction of suitable internal representations. *Proceedings of the International Joint Conference on Neural Networks*, Nagoya, Japan, 536–539.

Rao, S. S. (1984). *Optimization theory and applications*. New Delhi: Wiley Eastern.

Rohwer, R. (1990). The 'moving targets' training algorithm. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* (Vol. 2, pp. 558–565). San Mateo: Morgan Kaufmann.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (pp. 318–362). Cambridge, MA: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature*, **323**, 533–536.

Sato, A. (1991). An analytical study of the momentum term in a back-propagation algorithm. In T. Kohonen, K. Makisara, O. Simula, & J. Kangas (Eds.), *Artificial neural networks* (pp. 617–622). New York: Elsevier.

Schmidhuber, J. (1988). *Accelerated learning in back propagation nets*. (Techn. Rep.). Institut für Informatik, Technische Universität München.

Shah, S., Palmieri, F., & Datum, M. (1992). Optimal filtering algorithms for fast learning in feedforward neural networks. *Neural Networks*, **5**, 779–787.

Singhal, S., & Wu, L. (1989). Training feed-forward networks with the extended Kalman filter. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1187–1190.

Sutton, R. S. (1986). Two problems with back-propagation and other steepest-descent learning procedures for networks. *Proceedings of the Eighth Annual Conference of Cognitive Science Society*, 823–831.

Tesauro, G., & Janssens, B. (1988). Scaling relationships in back propagation learning. *Complex Systems*, **2**, 39–44.

Tollenaere, T. (1990). SuperSAB: Fast adaptive back propagation with good scaling properties. *Neural Networks*, **3**, 561–573.

van Ooyen, A., & Nienhuis, B. (1992). Improving the convergence of the back propagation algorithm. *Neural Networks*, **5**, 465–471.

Varoufakis, S., Perantonis, S., & Karras, D. (1993). A class of efficient learning algorithms for feedforward networks based on constrained optimization techniques. *Proceedings of NEURONET' 93*, Prague, Czech Republic, 1–9.

Watrous, R. L. (1987). Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization. *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, **2**, 619–627.

Weigend, A. S., Rumelhart, D. E., & Huberman, B. A. (1991). Generalization by weight elimination with application to forecasting. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* (pp. 875–882). San Mateo: Morgan Kaufmann.

APPENDIX

Let us denote by e_i the number of epochs recorded in the ith successful trial. The expected value of any of the e_i represents the "true" average $\bar{\mathcal{M}}$ of the epoch distribution. Moreover, let Σ denote the standard deviation of any of the e_i. According to the Central Limit Theorem, the distribution of the sample average number of epochs $\mathcal{M} = (e_1 + e_2 + \ldots + e_\nu)/\nu$ for a sufficiently large number ν of independent trials tends to the normal distribution $N(\bar{\mathcal{M}}, \Sigma/\sqrt{\nu})$. It follows that $(\mathcal{M} - \bar{\mathcal{M}})\sqrt{\nu}\Sigma^{-1}$ tends to the normal distribution $N(0, 1)$. It is required that the probability

$$\mathcal{P}[|\mathcal{M} - \bar{\mathcal{M}}| \leq \gamma\bar{\mathcal{M}}] = \mathcal{P}[|\mathcal{M} - \bar{\mathcal{M}}|\sqrt{\nu}\Sigma^{-1} \leq \gamma\bar{\mathcal{M}}\sqrt{\nu}\Sigma^{-1}] \quad (31)$$

be greater or equal to b. Therefore,

$$\frac{2}{\sqrt{2\pi}} \int_0^{\gamma\bar{\mathcal{M}}\sqrt{\nu}\Sigma^{-1}} \exp(-t^2/2)\, dt \geq b, \quad (32)$$

which can be rewritten as

$$\operatorname{erf}\left(\frac{\gamma\bar{\mathcal{M}}\sqrt{\nu}}{\sqrt{2}\Sigma}\right) \geq b. \quad (33)$$

By substituting $\bar{\mathcal{M}}$ and Σ by their experimentally determined estimates $\mathcal{M}$ and σ, we readily obtain the experimental estimate given by eqn (30) for the minimum number $\nu_{\min}$ of trials required to achieve the desired level of statistical significance for $\mathcal{M}$.