Neural, Parrallel & Scientific Computations 8 (2000) 147-168

A Class of Adaptive Learning Rate Algorithms Derived by One–dimensional Subminimization Methods

G.D. Magoulas[1,3] and M.N. Vrahatis[2,3]

[1]Department of Information Systems and Computing, Brunel University
Uxbridge UB8 3PH, United Kingdom

[2]Department of Mathematics,University of Patras
GR-261.10 Patras, Greece

[3]University of Patras Artificial Intelligence Research Center-UPAIRC

ABSTRACT: A new class of adaptive learning rate algorithms for backpropagation networks is presented. The algorithms of this class exploit the parallelism inherent in the evaluation of the gradient of the error function to compute a different learning rate for each weight. To this end, one–dimensional subminimization is applied along each weight direction. The algorithms of this class are analyzed as composite nonlinear Jacobi methods applied to the gradient of the error function. Simulations are conducted to evaluate the convergence behavior of two training algorithms of this class and to compare them with several popular training methods.

Keywords - feedforward neural networks, gradient descent, back-propagation algorithm, minimization methods, nonlinear Jacobi method, parallel algorithms.

1. INTRODUCTION

The efficient supervised training of Feedforward Neural Networks (FNNs) is a subject of considerable ongoing research and numerous algorithms have been proposed to this end. The special case of batch training of an FNN is consistent with the theory of unconstrained optimization, since the information from all the training set is used. It can be viewed as the minimization of the error function E; that is to find a minimizer $w^* = (w_1^*, w_2^*, \dots, w_n^*) \in \mathbb{R}^n$, 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 over the entire training set. This minimization corresponds to updating the weights by epoch and, in order to be successful, it requires a sequence of weight

Received Febuary 1, 2000

iterates $\{w^k\}_{k=0}^{\infty}$, where k indicates epochs, which converges to a minimizer w^*. The rapid computation of such a minimizer is a rather difficult task since, in general, the number of network variables is large and the corresponding nonconvex error function possesses multitudes of local minima and has broad flat regions adjoined with narrow steep ones.

Let us consider the family of gradient based training algorithms having the iterative form:

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

where w^k is the current weight vector, d^k is a search direction, and η^k is the learning rate. Various choices of the direction d^k give rise to distinct algorithms. A broad class of methods uses the search direction $d^k = -\nabla E(w^k)$, where the gradient vector $\nabla E(w)$ is obtained by means of back-propagation of the error through the layers of the network (Rumelhart *et al.*, 1986). The most popular training algorithm of this class, named batch Back-Propagation (BP), minimizes the error function using the following steepest descent method with a constant, heuristically chosen, learning rate η:

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

A small value for the learning rate is usually chosen, $0 < \eta < 1$, in order to secure the convergence of the BP training algorithm and to avoid oscillations in a direction where the error function is steep. However, it is well known that this approach tends to be inefficient. This happens, for example, when the search space contains long ravines that are characterized by sharp curvature across them and a gently slopping floor (Rumelhart *et al.*, 1986; Jacobs, 1988). In addition, the use of a constant learning rate introduces difficulties in obtaining convergence of back-propagation training algorithms; see for example the work of Kuan *et al.*, (1991) and of Liu *et al.*, (1995). On the other hand, there are theoretical results that guarantee the convergence when the learning rate is constant. In this case the learning rate is proportional to the inverse of the Lipschitz constant which, in practice, is not easily available (Armijo, 1966; Magoulas *et al.*, 1997a).

Several heuristic methods have been suggested to dynamically adapt the learning rate during training as an alternative to the constant learning rate strategy with the aim to accelerate the convergence (Chan *et al.*, 1987; Vogl *et al.*, 1988; Battiti, 1989). A different approach is to exploit the local shape of the error surface as described by the direction cosines (Hsin *et al.*, 1995) or the local estimation of the Lipschitz constant (Magoulas *et al.*, 1997a). Alternatively, 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 (Parker, 1987; Watrous, 1987; Battiti, 1992; Møller, 1993; Van der Smagt, 1994; Magoulas *et al.*, 1997b). However, these training algorithms are computationally intensive for FNNs with several hundred weights. Furthermore, it is not certain that the extra computational cost speeds up the minimization process for nonconvex functions when the initial points are far from a minimizer (Dennis and Moré, 1977; Nocedal, 1992), as is usually the case with the neural network training problem (Battiti, 1992).

In this paper we present a class of training algorithms which are derived by one–dimensional subminimization methods. A different learning rate for each weight is adapted by means of one–dimensional subminimization. This approach help us to develop algorithms which provide fast training without fluctuations and a greater possibility of good performance. These characteristics are considered useful in neural network applications.

The paper is organized as follows. In Section 2 the well known class of training algorithms that employ a heuristically determined adaptive learning rate for each weight is presented and the advantages as well as the disadvantages of these algorithms are discussed. The new class of training algorithms is introduced in Section 3 and its convergence properties are investigated in Section 4. Experiments to evaluate and compare the performance of two algorithms of this class with several other training methods are presented in Section 5. The paper ends, in Section 6, with conclusions.

2. TRAINING WITH A HEURISTICALLY DETERMINED LEARNING RATE FOR EACH WEIGHT

The use of a different learning rate for each weight allows us to find the proper learning rate that compensates for the small magnitude of the gradient in a flat weight direction in order to avoid slow convergence, and dampens a large weight change in a steep weight direction in order to avoid oscillations. Moreover, this approach exploits the parallelism inherent in the evaluation of $\nabla E(w)$ by the BP algorithm. Various batch-type BP training algorithms with an adaptive learning rate for each weight have been suggested in the literature (Jacobs, 1988; Fahlman, 1989; Silva and Almeida, 1990; Pfister and Rojas, 1993; Riedmiller and Braun, 1993). Following this approach Eq. (3) is reformulated to the following scheme:

$$w^{k+1} = w^k - \operatorname{diag}\{\eta_1^k, \ldots, \eta_n^k\}\, \nabla E(w^k), \tag{4}$$

where $\operatorname{diag}\{\eta_1^k, \ldots, \eta_n^k\}$ defines the diagonal matrix with elements $\{\eta_1^k, \ldots, \eta_n^k\}$.

The algorithms that follow the above scheme try to decrease the error by searching a local minimum with small weight steps. These steps are usually constraint by problem-dependent heuristic parameters in an attempt to avoid oscillations, and to ensure minimization of the error function in each weight direction. However, this approach usually results in a trade-off between the convergence speed and the stability of the training algorithm. For example, the ***delta-bar-delta*** method (Jacobs, 1988) or the ***quickprop*** method (Fahlman, 1989) introduce additional highly problem-dependent heuristic coefficients to alleviate the stability problems.

Heuristically chosen learning rate lower and upper bounds are also suggested with the aim to avoid the usage of an extremely small, or large, learning rate component, which misguides the resultant search direction. The learning rate lower bound helps to avoid unsatisfactory convergence rate, while the learning rate upper bound limits the influence of a large learning rate component on the resultant search direction, and depends on the shape of the error function.

A well known difficulty of this approach is that the use of inappropriate heuristic values for a weight direction misguides the resultant search direction. In such cases, these training algorithms cannot exploit the global information obtained by taking into consideration all the directions. This is the case of many well known training algorithms that employ heuristics for properly tuning the adaptive learning rates (Jacobs, 1988; Fahlman, 1989; Silva and Almeida, 1990; Pfister and Rojas, 1993; Riedmiller and Braun, 1993) and no guarantee is provided that the weight updates will converge to a minimizer of E. Recently, a modification of the Fahlman's method has been proposed which exhibits improved convergence characteristics (Vrahatis *et al.*, 2000a). In certain cases the aforementioned methods, although initially developed for batch training, can be used for on-line training by minimizing a pattern-based error measure.

3. LEARNING RATE ADAPTATION BY SUBMINIMIZATION IN EACH WEIGHT DIRECTION

It is well known that a minimizer w^* of a continuous differentiable function E should satisfy the necessary conditions:

$$\nabla E(w^*) = \Theta^n = (0, 0, \ldots, 0). \tag{5}$$

Eq. (5) represents a set of n nonlinear equations which must be solved to obtain w^* (Rao, 1992). Therefore, one approach to the minimization of the error function E is to seek the solutions of the set of Eq. (5) by including a provision to ensure that the solution found does, indeed, correspond to a local minimizer. This is equivalent to solving the following system of equations:

$$\begin{aligned} \partial_1 E(w_1, w_2, \ldots, w_n) &= 0, \\ \partial_2 E(w_1, w_2, \ldots, w_n) &= 0, \\ &\vdots \\ \partial_n E(w_1, w_2, \ldots, w_n) &= 0, \end{aligned} \tag{6}$$

where $\partial_i E(w_1, \ldots, w_i, \ldots, w_n)$ denotes the partial derivative of E with respect to the ith weight.

Next, we consider the class of *nonlinear Jacobi methods* applied to System (6). These methods are widely used for the numerical solution of a system of nonlinear equations. The main feature of the nonlinear Jacobi process is that it is a parallel algorithm (Ortega and Rheinboldt, 1970), i.e. it applies a parallel update of the variables.

Starting from an arbitrary initial weight vector $w^0 \in \mathcal{D}$, one can subminimize at the kth epoch the function:

$$E(w_1^k, \ldots, w_{i-1}^k, w_i, w_{i+1}^k, \ldots, w_n^k), \tag{7}$$

along the ith direction and obtain the corresponding subminimizer $\hat{w}_i$. Obviously for the subminimizer $\hat{w}_i$ holds:

$$\partial_i E(w_1^k, \ldots, w_{i-1}^k, \hat{w}_i, w_{i+1}^k, \ldots, w_n^k) = 0. \tag{8}$$

This is a one-dimensional subminimization because all other components of the weight vector, except the ith, are kept constant. Then, the ith weight is updated according to the equation:

$$w_i^{k+1} = w_i^k + \tau_k(\hat{w}_i - w_i^k), \tag{9}$$

for some relaxation factor τ_k. The error function in (7) is subminimized in parallel for all i.

Various composite nonlinear Jacobi training algorithms can be obtained depending on the one–dimensional minimization method applied. It is worth noticing that the number of the iterations of the subminimization method is related to the requested accuracy in obtaining the subminimizer approximations. Thus, significant computational effort is needed in order to find very accurate approximations of the subminimizer in each weight direction at each epoch. Moreover, this computational effort is increased for FNNs with several hundred weights. On the other hand, it is not certain that this large computational effort speeds up the minimization process for nonconvex functions when the current weight vector is not close to a minimizer w^*. Thus, we propose to obtain $\hat{w}_i$ by minimizing the function (7) with one iteration of a minimization method. Note that this practice is also suggested for the iterative solution of nonlinear equations (Ortega and Rheinboldt, 1970; Voigt, 1971).

By properly tuning the relaxation factor τ_k we can obtain better weight updates because this factor defines the length of the minimization step along the resultant search direction. Thus, we are able to avoid temporary oscillations and/or to enhance the rate of convergence when the current weight vector is far from a minimizer.

Below we synthesize three adaptive learning rate algorithms of this class. These algorithms employ a different learning rate for each weight based on traditional one-dimensional minimization methods. The first one requires only the sign of the gradient values, while the other two exploit both the function and gradient values.

3.1. The multi-step Jacobi-modified bisection method

In order to compute a minimizer approximation $\hat{w}_i$ in the interval $[a_i, b_i]$ we use the following iterative formula (Vrahatis, 1988a; Vrahatis, 1988b):

$$w_i^{p+1} = w_i^p + \operatorname{sgn} \partial_i E(w^0) \operatorname{sgn} \partial_i E(w^p)\, h_i/2^{p+1}, \quad p = 0, 1, \dots, \tag{10}$$

where sgn defines the well known triple valued sign and $w_i^0 = a_i$; $h_i = b_i - a_i$. Of course, the iterations (10) converge to $\hat{w}_i \in (a_i, b_i)$ if for some w_i^p, $p = 1, 2, \dots,$ the following condition holds:

$$\operatorname{sgn} \partial_i E(w^0) \operatorname{sgn} \partial_i E(w^p) = -1.$$

To ensure that $\hat{w}_i$ is a subminimizer along the ith weight direction, we choose the endpoints a_i and b_i in such a way that the ith component of the gradient vector at the left endpoint a_i has negative value, or, the ith component of the gradient vector at the right endpoint b_i has positive value. In order that this condition be fulfilled we choose the endpoints by means of the following relation:

$$a_i = w_i^k - \frac{1}{2}\Big\{1 + \operatorname{sgn} \partial_i E(w^k)\Big\} h_i - \operatorname{sgn} \partial_i E(w^k)\beta, \qquad b_i = a_i + h_i,$$

where β is a small positive real number which has dependence on the machine precision.

The maximum number ν of the iterations of the sequence (10), which are required to obtain an approximate minimizer w_i^* at each epoch along the ith direction, is related to a predefined accuracy $\delta \in (0, 1)$, and it is given by (Magoulas *et al.*, 1996; Vrahatis *et al.*, 1996):

$$\nu = \lceil \log_2 (h_i \delta^{-1}) \rceil,$$

where $\lceil \cdot \rceil$, defines the ceiling function.

Thus, the parameter $\hat{w}_i$ in the weight update equation (9) is the approximation of the subminimizer obtained by (10). For the implementation and the good performance characteristics of (10) in neural networks training with imprecision see (Magoulas *et al.*, 1996).

3.2. The one–step Jacobi–Newton method and an "approximated scheme"

A straightforward implementation of the one-step Jacobi-Newton iteration leads to the following weight update equation:

$$w_i^{k+1} = w_i^k - \tau_k \frac{\partial_i E(w^k)}{\partial_{ii}^2 E(w^k)}. \tag{11}$$

In order to avoid the calculation of the second derivative we propose the following scheme which utilizes the notion of the local Lipschitz constant (Magoulas *et al.*, 1999):

$$\Lambda_i^k = |\partial_i E(w^k) - \partial_i E(w^{k-1})| / |w_i^k - w_i^{k-1}|, \tag{12}$$

where w^k and w^{k-1} is a pair of consecutive weight updates at the kth iteration. Thus, the weights are updated according to the relation:

$$w_i^{k+1} = w_i^k - \tau_k \left\{ \frac{|\partial_i E(w^k) - \partial_i E(w^{k-1})|}{|w_i^k - w_i^{k-1}|} \right\}^{-1} \partial_i E(w^k). \tag{13}$$

In the steep regions of the error surface Eq. (13) uses a small value for the learning rate in order to guarantee convergence. On the other hand, when the error surface has flat regions, a large learning rate is used to accelerate the convergence. Note that in Eq. (13) the partial derivative with respect to the ith weight $\partial_i E(w^{k-1})$ is used. Thus, initial learning rates are needed to start the learning procedure.

3.3. A modified one–step Jacobi–Newton method

The problem of minimizing the error function E along the ith direction at the kth epoch:

$$\eta_i^* = \min_{\eta_i \geq 0} E(w^k + \eta_i e_i), \tag{14}$$

where e_i indicates the ith column of the identity matrix, is equivalent to seeking the value of η_i^* that minimizes the one-dimensional function:

$$\phi_i(\eta_i) = E(w^k + \eta_i e_i). \tag{15}$$

Since in neural network training $E(w) \geq 0$, for all $w \in \mathbb{R}^n$, then the point w^* with $E(w^*) = 0$ minimizes $E(w)$. Therefore the subminimization problem (14) can be handled by applying properly a root finding procedure to the equation:

$$\phi_i(\eta_i) = 0, \tag{16}$$

in order to obtain an approximation $\hat{\eta}_i$ of η_i^*. To this end, using one step of the Newton's method (see the discussion in the previous and in the next section for a justification of the one step procedure) we obtain:

$$\eta_i^1 = \eta_i^0 - \frac{\phi_i(\eta_i^0)}{\phi_i'(\eta_i^0)}. \tag{17}$$

Since $\eta_i^0 = 0$ we get:

$$\hat{\eta}_i = -\frac{\phi_i(\eta_i^0)}{\phi_i'(\eta_i^0)}. \tag{18}$$

From Eq. (15) we have:

$$\phi_i(\eta_i^0) = E(w^k + \eta_i^0 e_i) = E(w^k) \tag{19}$$

and

$$\phi_i'(\eta_i^0) = \nabla E(w^k)^\top e_i = \partial_i E(w^k). \tag{20}$$

Thus, Eq. (18) is reformulated as:

$$\hat{\eta}_i = -\frac{E(w^k)}{\partial_i E(w^k)}. \tag{21}$$

The value of $\hat{\eta}_i$ corresponds to the difference $(\hat{w}_i - w_i^k)$ of Eq. (9) and is calculated in parallel for all weight directions $(i = 1, ..., n)$ at each epoch.

Consequently, Eq. (9) takes the form:

$$w_i^{k+1} = w_i^k - \tau_k \frac{E(w^k)}{\partial_i E(w^k)}. \tag{22}$$

The iterative scheme (22) takes into consideration information from both, the error function and the magnitude of the gradient components. When the gradient magnitude is small, the local shape of E in this direction is flat, otherwise it is steep. The value of the error function indicates how close to the global minimizer this local shape is. The above pieces of information help the iterative scheme (22) to escape from flat regions with high error values, which are located far from a desired minimizer. Note also that Eq. (22) does not need any initial learning rates.

4. CONVERGENCE ANALYSIS

The convergence analysis is studied under appropriate assumptions and provides useful insight into the new class. First, we recall two concepts which will be used in our convergence analysis. Then, we provide a local convergence analysis and we propose strategies for developing globally convergent adaptive learning rate algorithms.

4.1. The property A^π

Young in 1971 discovered a class of matrices described as having *property A* that can be partitioned into block-tridiagonal form, possibly after a suitable permutation (Young, 1971).

Definition 1 (Axelsson, 1996): The matrix A has the property A^π if A can be permuted by PAP^{T} into a form that can be partitioned into block-tridiagonal form, that is,

$$PAP^{\mathrm{T}} = \begin{bmatrix} D_1 & L_1^{\mathrm{T}} & & & \mathcal{O} \\ L_1 & D_2 & L_2^{\mathrm{T}} & & \\ & \ddots & \ddots & \ddots & \\ & & L_{r-2} & D_{r-1} & L_{r-1}^{\mathrm{T}} \\ \mathcal{O} & & & L_{r-1} & D_r \end{bmatrix},$$

where the matrices D_i, $i = 1, \dots, r$ are nonsingular. For an algorithm which transforms a symmetric matrix to tridiagonal form see (p.335 in (Stewart, 1973)).

4.2. The root-convergence factor

It is useful for any root finding iterative procedure to have a measure of the rate of its convergence. In our case, we are interested in how fast the training algorithms of the new class, denoted in general by $\mathcal{P}$, converge to w^*. A measure of the rate of convergence is obtained by taking appropriate roots of successive errors. To this end we use the following definition.

Definition 2 (Ortega and Rheinboldt, 1970): Let $\{w^k\}_{k=0}^{\infty}$ be any sequence that converges to w^*. Then the number

$$R\{w^k\} = \lim_{k\to\infty} \sup \|w^k - w^*\|^{1/k}, \tag{23}$$

is the *root-convergence factor*, or *R-factor* of the sequence of the weights. If the iterative procedure $\mathcal{P}$ converges to w^* and $C(\mathcal{P}, w^*)$ is the set of all sequences generated by $\mathcal{P}$ which convergence to w^*, then

$$R(\mathcal{P}, w^*) = \sup\{R\{w^k\}; \{w^k\} \in C(\mathcal{P}, w^*)\}, \tag{24}$$

is the *R-factor* of $\mathcal{P}$ at w^*.

4.3. Local convergence

In this subsection we focus on the local convergence behavior of the new class of algorithms. The objective is to show that there is a neighborhood of a minimizer of the error function for which convergence to the minimizer can be guaranteed. To this end, we recall some qualitative results which provide the theoretical convergence justification of this class of algorithms (Vrahatis *et al.*, 2000b).

Theorem 1: Let $E\colon \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable in an open neighborhood $\mathcal{S}_0 \subset \mathcal{D}$ of a point $w^* \in \mathcal{D}$ for which $\nabla E(w^*) = \Theta^n$ and the Hessian, $H(w^*)$ is positive definite with the property A^π. Then there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$ such that any sequence $\{w^k\}_{k=0}^{\infty}$ generated by the nonlinear Jacobi process $\mathcal{P}$ converges to w^* which minimizes E and $R(\mathcal{P}, w^*) < 1$.

Proof: Following the proof of Theorem 3.6 in (Vrahatis *et al.*, 2000b), the necessary and sufficient conditions for the point w^* to be a local minimizer of the function E are satisfied by the hypothesis $\nabla E(w^*) = \Theta^n$ and the assumption of positive definitiveness of the Hessian at w^* (see for example (Ortega and Rheinboldt, 1970)). Finding such a point is equivalent to obtaining the corresponding solution $w^* \in \mathcal{D}$ of the following equation:

$$\nabla E(w) = \Theta^n, \tag{25}$$

which minimizes $E(w)$ or, equivalently, to solve iteratively, in parallel, the system of equations (6) by applying the nonlinear Jacobi process and employing any one-dimensional method for the subminimization process.

Next we consider the decomposition of $H(w^*)$ into its diagonal, strictly lower-triangular and strictly upper-triangular parts:

$$H(w^*) = D(w^*) - L(w^*) - L^{\top}(w^*). \tag{26}$$

Since, $H(w^*)$ is symmetric and positive definite, then $D(w^*)$ is positive definite (Varga, 1962). Moreover, since $H(w^*)$ has the property A^π, the eigenvalues of

$$\Phi(w^*) = D(w^*)^{-1}\left[L(w^*) + L^{\top}(w^*)\right], \tag{27}$$

are real and the spectral radius of $\Phi(w^*)$ is $\rho(\Phi(w^*)) = \varrho < 1$ (Axelsson, 1996); then there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$, such that, for any initial weight vector $w^0 \in \mathcal{S}$, there is a sequence $\{w^k\}_{k=0}^{\infty} \subset \mathcal{S}$ which satisfies the composite methods defined by Eqs. (8)–(9) such that $\lim_{k\to\infty} w^k = w^*$ and $R(\mathcal{P}, w^*) = \varrho < 1$ (Ortega and Rheinboldt, 1970; Voigt, 1971). Thus the Theorem is proved. □

The proof of the above Theorem shows that the asymptotic rate of convergence of the algorithms is not enhanced if one takes more than one iteration of the one-dimensional subminimization method. It only depends on the value of the spectral radius ϱ. The theorem is applicable to any training algorithm that adapts a different learning rate for each weight using one-dimensional subminimization methods.

The local convergence analysis has been developed under appropriate assumptions and provides useful insight into the new class. However, in practice, neural network users want a guarantee that a training algorithm will reduce the error at each epoch and that the error will not fluctuate. Particularly, neural network practitioners are interested in techniques that will satisfy the above mentioned requirements when the initial weights are far from the neighborhood of a minimizer. Therefore we would like any iterative scheme of the new class to generate weight iterates that achieve a sufficient reduction in the error function at each epoch. Only these weight iterates will be accepted. Searching for an acceptable weight vector rather than a minimizer along the current search direction usually reduces the number of function evaluations per epoch and the same goes for the total number of function evaluations required to successfully train the FNN. This is due to the fact that training starts away from a local minimum of the error function and exact minimization steps along the search direction do not usually help, because of the nonlinearity of the error function. On the other hand, when the current iterate w^k is close to the minimizer a "better" approximator of the minimizer w^{k+1} can be found without much difficulty. This issues are investigated below in the framework of the global convergence properties of the new class.

4.4. Global convergence

By a globally convergent algorithm we mean an algorithm with the property that for any initial weight vector the weight sequence converges to a local minimizer of the error function (see Kelley, 1995; Dennis and Schnabel, 1996). In order to ensure global convergence of the new class of algorithms the following assumptions are needed (Kelley, 1995; Dennis and Schnabel, 1996):

a) The error function E is a real-valued function defined and continuous everywhere in $\mathbb{R}^n$, bounded below in $\mathbb{R}^n$,

b) for any two points w and $v \in \mathbb{R}^n$, ∇E satisfies the Lipschitz condition

$$\|\nabla E(w) - \nabla E(v)\| \leq L\|w - v\|, \tag{28}$$

where $L > 0$ denotes the Lipschitz constant.

The effect of the above assumptions is to place an upper bound on the degree of the nonlinearity of the error function and to ensure that the first derivatives are continuous in w. If these assumptions are fulfilled any algorithm of this class can be made globally convergent by determining the learning rates in such a way that the error function is exactly minimized along the current search direction at each epoch. To this end an iterative search, which is often expensive in terms of error function evaluations, is required. To alleviate this situation it is preferable to determine the learning rates so that the error function is sufficiently decreased at each epoch, accompanied by a significant change in the value of w. The following conditions, associated with the names of Armijo, Goldstein, Price and Wolfe (Ortega and Rheinboldt, 1970), are used to formulate the above ideas and to define a criterion of acceptance of any weight iterate:

$$E(w^{k+1}) - E(w^k) \leq \sigma_1 \tau_k \nabla E(w^k)^{\mathsf{T}} d^k, \tag{29}$$

$$\nabla E(w^{k+1})^{\mathsf{T}} \nabla E(w^k) \geq \sigma_2 \nabla E(w^k)^{\mathsf{T}} d^k, \tag{30}$$

where $0 < \sigma_1 < \sigma_2 < 1$ and d^k denotes the search direction and depends on the training algorithm. By setting $\tau_k = 1$ and by using appropriate values for the learning rates we seek to satisfy the conditions (29)–(30): the first condition ensures that the error function is reduced with every epoch and the second condition prevents the learning rate from becoming too small. Furthermore, these conditions can be used to enhance any training algorithm with tuning techniques that are able to handle arbitrarily large learning rates. Note that the value $\sigma_1 = 0.5$ is usually suggested in the literature (Armijo, 1966; Nocedal, 1992).

A simple technique to tune the length of the minimization step, so that it satisfies Conditions (29)–(30) at each epoch, is to decrease the learning rates by a reduction factor $1/q$, where $q > 1$ (Ortega and Rheinboldt, 1970). This has the effect that each learning rate is decreased by the largest number in the sequence $\{q^{-m}\}_{m=1}^{\infty}$, so that the condition (29) is satisfied. We remark here that the selection of q 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 rates by modest amounts, such as $1/2$, while others might respond well to a more aggressive learning rate reduction. Note that reducing η_i too much can be costly as well, since the total number of epochs will be increased. Consequently, when seeking to satisfy the condition (29) it is important to ensure that the learning rates are not reduced unnecessarily so that the condition (30) is not satisfied. Instead of checking the condition (30) it is possible to place a lower bound on the acceptable values of each learning rate: the learning rate lower bound is related to the desired accuracy in obtaining the final weights and helps to avoid unsatisfactory convergence rate. This bound has the same theoretical effect as the condition (30) and ensures global convergence (Dennis and Schnabel, 1996). The value $q = 2$ is usually suggested in the literature (Armijo, 1966) and, indeed, it has been found to work without problems in the experiments reported in the next section. We have also set the learning rate lower bound to 10^{-5}.

Another approach to perform learning rate reduction is to estimate the appropriate reduction factor at each epoch. This is achieved by modeling the decrease in the magnitude of the gradient vector as the learning rates are reduced. To this end, quadratic and cubic interpolations, which exploit the available information about the error function are suggested. Relative techniques have been proposed by Battiti (1989), Dennis and Schnabel (1996) and Looney (1997).

5. EXPERIMENTAL STUDY

In this section we give comparative results for six batch training algorithms: Back-propagation with constant learning rate (BP); Back-propagation with constant learning rate and constant Momentum (Rumelhart *et al.*, 1986), named BPM; Adaptive Back-propagation with adaptive momentum (ABP) proposed by Vogl *et al.* (1988); Rprop (Riedmiller and Braun, 1993); the *Approximated one-step Jacobi-Newton method* of Eq. (13), named AJN and the *Modified one-step Jacobi-Newton method* of Eq. (22), named MJN. The FNNs have been implemented in Matlab (Demuth and Beale, 1992) and 1000 simulations have been run in each test case.

The selection of initial weights is very important in FNN training (Wessel and Barnard, 1992). A well known initialization heuristic for FNNs is to select the weights with uniform probability from the interval $(-1, 1)$. This weight range has been used in several experimental studies (see Hirose *et al.*, 1991; Hoehfeld and Fahlman 1992, Pearlmutter 1992; Riedmiller and Braun, 1993).

The values of the learning parameters used in each problem are shown in Table 1. The initial learning rate has been the same for all algorithms tested; it has been carefully chosen so that the BP algorithm rapidly converges without oscillating towards a global minimum. Then, all the other learning parameters have been tuned by trying different values and comparing the number of successes that has been exhibited by 3 simulation runs that have been started from the same initial weights. However, if an algorithm has exhibited the same number of successes out of 3 runs for two different parameter combinations, then the average number of epochs has been checked and the combination that has provided the fastest convergence has been chosen.

To obtain the best possible convergence, the momentum term m in BPM is normally adjusted by trial and error or even by some kind of random search (Schaffer *et al.*, 1992). Since the optimal value is highly dependent on the learning task,.no general strategy has been developed to deal with this problem. Thus, the optimal value of m is experimental, but depends on the learning rate chosen. In our experiments, we have tried 9 different values for the momentum ranging from 0.1 to 0.9 and we have run 3 simulations combining all these values with the best available learning

Table 1: Learning parameters used in the experiments

Algorithm	8×8 font	Texture classification	Vowel spotting
BP	$\eta^0 = 1.2$	$\eta^0 = 0.001$	$\eta^0 = 0.0034$
BPM	$\eta^0 = 1.2$	$\eta^0 = 0.001$	$\eta^0 = 0.0034$
	$m = 0.9$	$m = 0.9$	$m = 0.7$
ABP	$\eta^0 = 1.2$	$\eta^0 = 0.001$	$\eta^0 = 0.0034$
	$m = 0.1$	$m = 0.9$	$m = 0.1$
	inc=1.05	$inc = 1.05$	$inc = 1.07$
	$dec = 0.7$	$dec = 0.5$	$dec = 0.8$
	$ratio = 1.04$	$ratio = 1.04$	$ratio = 1.04$
Rprop	$\eta^0 = 1.2$	$\eta^0 = 0.001$	$\eta^0 = 0.0034$
	$u = 1.3$	$u = 1.05$	$u = 1.3$
	$d = 0.7$	$d = 0.6$	$d = 0.7$
	$lb = 10^{-5}$	$lb = 10^{-5}$	$lb = 10^{-5}$
	$ub = 1$	$ub = 1$	$ub = 1$
AJN	$\eta^0 = 1.2$	$\eta^0 = 0.001$	$\eta^0 = 0.0034$
	$\tau = 1$	$\tau = 1$	$\tau = 1$
	$lb = 10^{-5}$	$lb = 10^{-5}$	$lb = 10^{-5}$
	$ub = 1$	$ub = 1$	$ub = 1$
MJN	$\tau = 1$	$\tau = 1$	$\tau = 1$
	$ub = 1$	$ub = 1$	$ub = 1$

rate for the BP.

Much effort has been made to properly tune the learning rate increment and decrement factors *inc*, *u*, *dec* and *d*. To be more specific, different values in steps of 0.05 to 2 have been tested for the learning rate increment factor, while different values between 0.1 and 0.9, in steps of 0.05, have been tried for the learning rate decrement factor. The error ratio parameter, denoted *ratio* in Table 1, has been set equal to 1.04. This value is generally suggested in the literature (Vogl *et al.*, 1988) and, indeed, it has been found to work better than others tested. The lower and upper learning rate bound, *lb* and *ub*, respectively, have been chosen so as to avoid unsatisfactory convergence rates (Riedmiller and Braun, 1993). All the combinations of these parameter values have been tested on 3 simulation runs starting from the same initial weights. The combination that has exhibited the best number of successes out of 3 runs has finally been chosen. If two different parameter combinations have exhibited the same number of successes (out of 3), then the combination with the smallest average number of epochs has been chosen.

A consideration that is worth mentioning is the difference between gradient and error function evaluations at each epoch: for the BP, the BPM, the ABP and the Rprop one gradient evaluation and one error function evaluation are necessary at each epoch; for AJN and MJN there is a number of additional error function evaluations

Table 2: Comparative results for the numeric font learning problem

Algorithm	Gradient Evaluation			Function Evaluation			Success
	μ	σ	Min/Max	μ	σ	Min/Max	%
BP	14489	2783.7	9421/19947	14489	2783.7	9421/19947	66
BPM	10142	2943.1	5328/18756	10142	2943.1	5328/18756	54
ABP	1975	2509.5	228/13822	1975	2509.5	228/13822	91
Rprop	289	189.1	56/876	289	189.1	56/876	90
AJN	159	24.1	119/245	581	105.9	404/884	100
MJN	1361	648.8	553/5286	3708	3041.4	555/26024	100

when the condition (29) is not satisfied. Thus, we compare the algorithms in terms of both gradient and error function evaluations. However, the reader has to consider the fact that a gradient evaluation is more costly than an error function evaluation. For example, Møller suggests to count a gradient evaluation three times more than an error function evaluation (Møller, 1993).

5.1. Numeric font learning problem

The first experiment refers to the training of a 64-6-10 FNN (444 weights, 16 biases) for recognizing 8×8 printed numerals from 0 to 9 (Sperduti and Starita, 1993). The network is based on neurons of the logistic activation model. Numerals are given in a finite sequence of input–output pairs (Inp_p, Out_p) where Inp_p are the binary 64-dimensional input vectors determining the 8×8 binary pixel and Out_p are 10-dimensional binary output vectors, for $p = 0, \ldots, 9$ determining the corresponding numerals. An error value $E \leq 10^{-3}$ has been chosen as termination condition for all algorithms tested.

Detailed results regarding the training performance of the algorithms are presented in Table 2, where μ denotes the mean number of gradient or error function evaluations required to obtain convergence, σ the corresponding standard deviation, Min/Max the minimum and maximum number of gradient or error function evaluations, and % denotes the percentage of successful simulations out of 1000 runs.

Obviously, the number of gradient evaluations is equal to the number of error function evaluations for the BP, the BPM, the ABP and the Rprop. AJN has the smallest average number of gradient evaluations which is considered very important in practice. Rprop requires less function evaluations by it reveals a smaller number of successful runs than AJN. Both AJN and MJN provide a greater possibility of successful training: they exhibit a 100% of success in the 1000 simulation runs. Regarding the performance of the MJN, it is better that the BP and the BPM (without needing an initial learning rate) and comparable to the overall performance of the

ABP which needs fine tuning five parameters.

5.2. Texture classification problem

The second experiment is a texture classification problem. A total of 12 Brodatz texture images (Brodatz, 1966): 3, 5, 9, 12, 15, 20, 51, 68, 77, 78, 79, 93 (see Figure 1) of size 512 $\times$ 512 has been acquired by a scanner at 150dpi. From each texture image 10 subimages of size 128 $\times$ 128 have been randomly selected and the co-occurrence method has been applied (Haralick *et al.*, 1973). This method evaluates a series of matrices that describe the spatial variation of gray-level values within a local area. In this way, the relative frequencies of gray-level pairs of pixels at certain relative displacements are computed and stored in a matrix form. In our simulations, four co-occurrence matrices have been computed for each sample area, with a displacement of one pixel and angles of 0°, 45°, 90° and 135°. 10 sixteenth-dimensional training patterns have been created from each image. A 16-8-12 FNN (224 weights, 20 biases) with logistic activations has been trained to classify the patterns to 12 texture types. A classification error *CE*$<$ 3% has been used as termination condition.

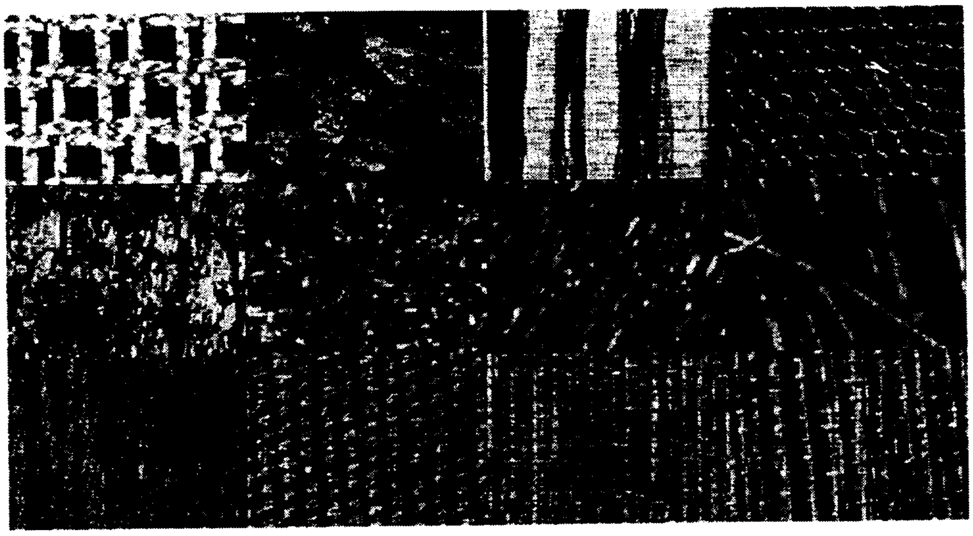

Figure 1: Twelve texture patterns obtained from digitizing the 'Brodatz Album' images 20, 5, 51, 3, 12, 9, 93, 15, 68, 77, 78 and 79.

The results of the training phase are shown in Table 3, where the abbreviations are as in Table 2. The successfully trained FNNs are tested for their generalization capability using patterns from 20 subimages of the same size randomly selected from each image. The average success rate of classification is: BP= 90%; BPM= 90%;

ABP= 93.5%; Rprop= 93.6%; AJN= 94.1%; MJN= 93%.

In general, AJN outperforms all other methods tested. MJN significantly outperforms BP and BPM in the number of gradient and error function evaluations as well as in the percentage of successful simulations. ABP exhibits very good performance, however it requires fine tuning four additional learning parameters. Rprop is also faster than MJN needing tuning five learning parameters, but has smaller percentage of success than MJN.

5.3. Vowel spotting problem

Vowel spotting provides a preliminary acoustic labeling of speech, which can be very important for both speech and speaker recognition procedures. A 15-15-1 FNN (240 weights and 16 biases), based on neurons of hyperbolic tangent activations, is trained as speaker independent using labeled training data from a large number of speakers from the TIMIT database (Fisher *et al.*, 1987). The sampled speech data are segmented into 30ms frames with a 15ms sliding window in overlapping mode. After applying a Hamming window, each frame is analyzed using the Perceptual Linear Predictive (PLP) speech analysis technique to obtain the characteristic features of the signal. The choice of the proper features is based on the work of (Sirigos *et al.*, 1995) which proposes a 15-dimensional feature vector for each frame. The feature vectors are classified into $\{-1,+1\}$ for the non-vowel/vowel model.

The FNN is part of a text-independent speaker identification and verification system which is based on using only the vowel part of the signal. The fact that the system uses only the vowel part of the signal makes the cost of mistakenly accepting a non-vowel and considering it as a vowel much more than the cost of rejecting a vowel and considering it as a non-vowel. A mistaken decision regarding a non-vowel will produce unpredictable errors to the speaker classification module of the system that uses the response of the FNN and is trained only with vowels (Fakotakis and Sirigos, 1996).

Table 3: Comparative results for the texture classification problem

Algorithm	Gradient Evaluation			Function Evaluation			Success
	μ	σ	Min/Max	μ	σ	Min/Max	%
BP	15839	3723.3	8271/25749	15839	3723.3	8271/25749	96
BPM	12422	2912.1	4182/18756	12422	2912.1	4182/18756	94
ABP	560	270.4	310/2052	560	270.4	310/2052	100
Rprop	703	2112.6	82/18750	703	2112.6	82/18750	88
AJN	382	129.6	217/1242	591	200.9	343/1930	100
MJN	791	512.3	417/5318	2185	1405.9	1116/14006	100

Table 4: Comparative results for the vowel spotting problem

Algorithm	Gradient Evaluation			Function Evaluation			Success
	μ	σ	Min/Max	μ	σ	Min/Max	%
BP	905	1067.5	393/6686	905	1067.5	393/6686	63
BPM	802	1852.2	381/9881	802	1852.2	381/9881	57
ABP	1146	1374.4	302/6559	1146	1374.4	302/6559	73
Rprop	296	584.3	79/3000	296	584.3	79/3000	80
AJN	169	90.4	108/520	545	175.8	315/1092	82
MJN	362	358.5	96/1580	1756	1692.1	459/7396	64

Thus, in order to minimize the false acceptance error rate which is more critical than the false rejection error rate, we polarize the training procedure by taking 317 non-vowel patterns and 43 vowel patterns. The training terminates when the classification error is less than 2%. After training, the generalization capability of the successfully trained FNNs is examined with 769 feature vectors taken from different utterances and speakers. The results of the training phase are shown in Table 4, where the abbreviations are as in Table 2.

The performance of the FNNs that have been trained using adaptive methods is evaluated in terms of the average improvement on the error rate percentage, that has been achieved by BP trained FNNs, that is 9%. Thus, FNNs trained with BPM do not improve the error rate achieved by the BP, thus the improvement is 0%. The average improvement achieved by the adaptive methods is: ABP= 0.1%; Rprop= 1.3%; AJN= 1.3%; MJN= 1%. The above results show that the increased training speed achieved by the adaptive learning rate methods does not affect their generalization capability.

AJN exhibits the best performance. MJN compares favorably on the number of gradient evaluations to BP, BPM and ABP without using an initial learning rate. It also improves the error rate achieved by the BP by 1%. ABP exhibits the slowest convergence but has a reliable performance regarding the number of successful simulations. Note that Rprop provides fast training and has very good performance regarding the generalization capability.

6. CONCLUSIONS

A class of gradient-based training algorithms has been presented. These algorithms apply one-dimensional subminimization methods to adapt the rate of learning and provide accelerated training without oscillation by ensuring that the error function is sufficiently decreased with every iteration. In addition, they ensure global convergence, i.e. convergence to a local minimizer of the error function from any starting point, and provide stable learning and, therefore, a greater possibility of good performance.

Two algorithms of this class have been tested and compared with several training algorithms. Their efficiency has been numerically confirmed by the experiments presented in this paper.

7. ACKNOWLEDGEMENT

George D. Magoulas was supported by a postdoctoral fellowship from the Greek State Scholarship Foundation (I.K.Y.).

REFERENCES

1. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1, D. E. Rumelhart, & J. L. McClelland, eds., pp. 318–362. MIT Press, Cambridge, Massachusetts.
2. Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, vol. 1, pp.295–307.
3. Kuan, C.M., & Hornik, K. (1991). Convergence of learning algorithms with constant learning rates. *IEEE Transactions on Neural Networks*, vol. 2, pp.484–488.
4. Liu, R., Dong, G., & Ling, X. (1995). A convergence analysis for neural networks with constant learning rates and non-stationary inputs. In *Proceedings of the 34th Conference on Decision & Control*, New Orleans, pp.1278-1283.
5. Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, vol. 16, pp.1–3.
6. Magoulas, G. D., Vrahatis, M. N., & Androulakis, G. S. (1997a). Effective back–propagation with variable stepsize. *Neural Networks*, vol. 10, pp.69–82.
7. Chan, L. W., & Fallside, F. (1987). An adaptive training algorithm for back–propagation networks. *Computers, Speech & Language*, vol. 2, pp.205–218.
8. Vogl, T. P, Mangis, J. K., Rigler, J. K., Zink, W. T, & Alkon, D. L. (1988). Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, vol. 59, pp.257–263.
9. Battiti, R. (1989). Accelerated backpropagation learning: two optimization methods. *Complex Systems*, vol. 3, pp.331–342.
10. Hsin, H.-C., Li, C.-C., Sun, M., & Sclabassi, R.J. (1995). An adaptive training algorithm for back–propagation neural networks. *IEEE Transactions on System, Man and Cybernetics*, vol. 25, pp.512–514.
11. Parker, D. B. (1987). Optimal Algorithms for Adaptive Networks: Second Order Back-Propagation, Second Order Direct Propagation, and Second Order Hebbian Learning. In *Proceedings of the IEEE International Conference on Neural Networks*, vol. 2, pp.593–600. Piscataway, New Jersey: IEEE Press.
12. Watrous, R. L. (1987). Learning Algorithms for Connectionist Networks: Applied Gradient of Nonlinear Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*,vol. 2, pp.619–627. Piscataway, New Jersey: IEEE Press.

13. Battiti, R. (1992). First- and second-order methods for learning: between steepest descent and Newton's method. *Neural Computation*, vol. 4, pp.141-166.

14. Møller, M. F. (1993). A scaled conjugate gradient algorithm, for fast supervised learning. *Neural Networks*, vol. 6, pp.525–533.

15. Van der Smagt, P. P. (1994). Minimization Methods for training feedforward neural networks. *Neural Networks*, vol. 7, pp.1–11.

16. Magoulas, G. D., Vrahatis, M. N., Grapsa, T. N., & Androulakis, G. S. (1997b). 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, & I. J. Anderson, eds., pp. 245–249, Kluwer.

17. Dennis, J. E., & Moré, J. J. (1977). Quasi-Newton methods, motivation and theory. *SIAM Review*, vol. 19, pp.46–89.

18. Nocedal, J. (1992). Theory of algorithms for unconstrained optimization. *Acta Numerica*, pp.199–242.

19. Fahlman, S. E. (1989). Faster-learning variations on back-propagation: an empirical study. In *Proceedings of the 1988 Connectionist Models Summer School*, D.S. Touretzky, G.E. Hinton, & T.J. Sejnowski, eds., pp. 38–51. San Mateo: Morgan Kaufmann.

20. Silva, F., & Almeida, L. (1990). Acceleration techniques for the back-propagation algorithm. *Lecture Notes in Computer Science*, vol. 412, pp.110–119. Berlin: Springer-Verlag.

21. Pfister, M., & Rojas, R. (1993). Speeding-up backpropagation -A comparison of orthogonal techniques. In *Proceedings of the Joint Conference on Neural Networks*, Nagoya, Japan, pp.517–523. Piscataway, New Jersey: IEEE Press.

22. Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the Rprop algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, pp.586–591. Piscataway, New Jersey: IEEE Press.

23. Vrahatis, M.N., Magoulas, G.D. & Plagianakos, V.P. (2000a). Globally convergent modification of the Quickprop method, *Neural Processing Letters*, to appear vol. 12, No. 2, October 2000.

24. Rao, S.S. (1992). *Optimization Theory and Applications.* New Delhi: Wiley Eastern Limited.

25. Ortega, J.M., & Rheinboldt, W.C. (1970). *Iterative Solution of Nonlinear Equations in Several Variables.* New York:Academic Press.

26. Voigt, R.G. (1971). Rates of convergence for a class of iterative procedures. *SIAM Journal of Numerical Analysis*, vol. 8, pp.127–134.

27. Vrahatis, M. N. (1988a). Solving systems of nonlinear equations using the nonzero value of the topological degree. *ACM Transactions Mathematical Software*, vol. 14, pp.312–329 .

28. Vrahatis, M. N. (1988b). CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations. *ACM Transactions Mathematical Software*, vol. 14, pp.330–336.

29. Magoulas, G. D., Vrahatis, M. N., & Androulakis, G. S. (1996). A new method in neural network supervised training with imprecision. In *Proceedings of the IEEE 3rd International Conference on Electronics, Circuits and Systems*, pp. 287–290. Piscataway, New Jersey: IEEE Press.

30. Vrahatis, M. N., Androulakis, G. S., & Manousakis, G.E. (1996). A new unconstrained optimization method for imprecise function and gradient values. *Journal of Mathematical Analysis & Applications*, vol. 197, pp.586–607.

31. Magoulas, G. D., Vrahatis, M. N., & Androulakis, G. S. (1999). Improving the convergence of

the back-propagation algorithm using learning rate adaptation methods, *Neural Computation*, vol. 11, pp.1769–1796.

32. Young, D. (1971). Iterative methods for solving partial difference equations of elliptic type. *Transactions American Mathematical Society*, vol. 76, pp.92–111.
33. Axelsson, O. (1996). *Iterative Solution Methods.* New York: Cambridge University Press.
34. Stewart G.W., (1973). *Introduction to Matrix Computations.* New York: Academic Press.
35. Vrahatis, M.N., Androulakis, G.S., Lambrinos, J.N. & Magoulas, G.D. (2000b). A class of gradient unconstrained minimisation algorithms with adaptive stepsize. *Journal of Computational and Applied Mathematics*, vol. 114, pp.367–386.
36. Varga, R. (1962). *Matrix Iterative Analysis.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc.
37. Kelley, C.T. (1995). *Iterative Methods for Linear and Nonlinear Equations.* Philadelphia: SIAM publications.
38. Dennis, J. E., & Schnabel, R. B. (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Philadelphia: SIAM publications.
39. Looney, C. G. (1997). *Pattern Recognition Using Neural Networks.* New York: Oxford University Press.
40. Demuth, H., & Beale, M. (1992). *Neural Network Toolbox User's Guide.* Natick, Massachusetts: The MathWorks Inc.
41. Wessel, L.F., & Barnard, E. (1992). Avoiding false local minima by proper initialization of connections. *IEEE Transactions Neural Networks*, vol. 3, pp.899–905.
42. Hirose, Y., Yamashita, K., & Hijiya, S. (1991). Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, vol. 4, pp.61–66.
43. Hoehfeld, M., & Fahlman, S.E. (1992). Learning with limited numerical precision using the cascade-correlation algorithm. *IEEE Transactions on Neural Networks*, vol. 3, pp.602–611.
44. Pearlmutter, B. 1992. Gradient descent: second-order momentum and saturating error. In *Advances in Neural Information Processing Systems 4*, J.E. Moody, S.J. Hanson, & R.P. Lippmann, eds., pp. 887–894. San Mateo: Morgan Kaufmann.
45. Schaffer, J., Whitley, D., & Eshelman, L. (1992). Combinations of genetic algorithms and neural networks: a survey of the state of the art. In *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pp.1–37. Los Alamitos: IEEE Computer Society Press.
46. Sperduti, A., & Starita, A. (1993). Speed up learning and network optimization with extended back-propagation. *Neural Networks*, vol. 6, pp.365–383.
47. Brodatz P. *Textures- a Photographic Album for Artists and Designer.* New York: Dover.
48. Haralick, R., Shanmugan, K., & Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions System, Man & Cybernetics*, vol. 3, pp.610-621.
49. Sirigos, J., Fakotakis, N., & Kokkinakis, G. (1995). A comparison of several speech parameters for speaker independent speech recognition and speaker recognition. In *Proceedings of the 4th European Conference of Speech Communications and Technology.*
50. Fisher, W., Zue, V., Bernstein, J., & Pallet, D. (1987). An acoustic-phonetic data base. *Journal of Acoustical Society of America*, Suppl. A, vol. 81, pp.581–592.
51. Fakotakis, N., & Sirigos, J. (1996). A high-performance text-independent speaker recognition system based on vowel spotting and neural nets. In *Proceedings of the IEEE International Conference on Acoustic Speech and Signal Processing*, vol. 2, pp.661–664. Piscataway, New Jersey: IEEE Press.