Brill Academic Publishers
P.O. Box 9000, 2300 PA Leiden,
The Netherlands

Lecture Series on Computer and Computational Sciences
Volume 1, 2005, pp. 1-15

Parameter Optimization Algorithm with Improved Convergence Properties for Adaptive Learning

G.D. Magoulas[†1], M.N. Vrahatis[‡2]

[†]School of Computer Science and Information Systems,
Birkbeck University of London, London WC1E 7HX, UK

[‡]Computational Intelligence Laboratory, Department of Mathematics,
University of Patras, GR-26110 Patras, Greece

Abstract: The error in an artificial neural network is a function of adaptive parameters (weights and biases) that needs to be minimized. Research on adaptive learning usually focuses on gradient algorithms that employ problem–dependent heuristic learning parameters. This fact usually results in a trade–off between the convergence speed and the stability of the learning algorithm. The paper investigates gradient–based adaptive algorithms and discusses their limitations. It then describes a new algorithm that does not need user–defined learning parameters. The convergence properties of this method are discussed from both theoretical and practical perspective. The algorithm has been implemented and tested on real life applications exhibiting improved stability and high performance.

Keywords: Feedforward neural networks, Supervised training, Back-propagation algorithm, Heuristic learning parameters, Non-linear Jacobi process, Globally convergent algorithms, Local convergence analysis.

Mathematics Subject Classification: 65K10, 49D10, 68T05, 68G05

1 Introduction

Let us first define the notation we will use in the paper. We use a unified notation for the weights. Thus, for a feedforward neural network (FNN) with a total of n weights, $\mathbb{R}^n$ is the n–dimensional real space of column weight vectors w with components $w_1, w_2, \ldots, w_n$ and w^* is the optimal weight vector with components $w_1^*, w_2^*, \ldots, w_n^*$; E is the batch error measure defined as the sum–of–squared–differences error function over the entire training set; $\partial_i E(w)$ denotes the partial derivative of $E(w)$ with respect to the ith variable w_i; $g(w) = \Big(g_1(w), \ldots, g_n(w)\Big)$ defines the gradient $\nabla E(w)$ of the sum-of-squared-differences error function E at w, which is computed by applying the chain rule on the layers of an FNN (see [48]), while $H = [H_{ij}]$ defines the Hessian $\nabla^2 E(w)$ of E at w. Also, throughout this paper $\mathrm{diag}\{e_1, \ldots, e_n\}$ defines the $n \times n$ diagonal matrix with elements $e_1, \ldots, e_n$, $\Theta^n = (0, 0, \ldots, 0)$ denotes the origin of $\mathbb{R}^n$ and $\rho(A)$ is the spectral radius of matrix A.

The Back-Propagation (BP) algorithm [48] is widely recognized as a powerful tool for training FNNs. It minimizes the error function using the Steepest Descent (SD) method [15] with constant learning rate η:

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

[1]E-mail: gmagoulas@dcs.bbk.ac.uk
[2]E-mail: vrahatis@math.upatras.gr

The SD method requires the assumption that E is twice continuously differentiable on an open neighborhood $\mathcal{S}(w^0)$, where $\mathcal{S}(w^0) = \{w : E(w) \leq E(w^0)\}$ is bounded, for some initial weight vector w^0. It also requires that η is chosen to satisfy the relation $\sup \|H(w)\| \leq \eta^{-1} < \infty$ in the level set $\mathcal{S}(w^0)$ [16, 17]. The approach adopted in practice is to apply a small constant learning rate value ($0 < \eta < 1$) in order to secure the convergence of the BP training algorithm and avoid oscillations in a direction where the error function is steep. However, this approach considerably slows down training since, in general, a small learning rate may not be appropriate for all the portions of the error surface. Furthermore, it affects the convergence properties of training algorithms (see [25, 29]). Nevertheless, there are theoretical results that guarantee convergence when the learning rate is constant. This happens when the learning rate is proportional to the inverse of the Lipschitz constant which, in practice, is not easily available [2, 31].

Attempts to adaptive learning are usually based on the following approaches: (i) start with a small learning rate and increase it exponentially, if successive iterations reduce the error, or rapidly decrease it, if a significant error increase occurs [4, 56], (ii) start with a small learning rate and increase it, if successive iterations keep gradient direction fairly constant, or rapidly decrease it, if the direction of the gradient varies greatly at each iteration [8] and (iii) for each weight an individual learning rate is given, which increases if the successive changes in the weights are in the same direction and decreases otherwise. The well known *delta-bar-delta* method [18] and Silva and Almeida's method [49] follow this approach. Another method, named *quickprop*, has been presented in [11]. Quickprop is based on independent secant steps in the direction of each weight. Riedmiller and Braun in 1993 proposed the *Rprop* algorithm. The algorithm updates the weights using the learning rate and the sign of the partial derivative of the error function with respect to each weight. Note that all these adaptation methods employ heuristic learning parameters in an attempt to secure converge of the BP algorithm to a minimizer of E and avoid oscillations.

A different approach is to exploit the local shape of the error surface as described by the direction cosines or the Lipschitz constant. In the first case the learning rate is a weighted average of the direction cosines of the weight changes at the current and several previous successive iterations [23], while in the second case the learning rate is an approximation of the Lipschitz constant [31].

A variety of approaches adapted from numerical analysis have also been applied, in an attempt to use not only the gradient of the error function but also the second derivative in constructing efficient supervised training algorithms to accelerate the learning process. However, training algorithms that apply nonlinear conjugate gradient methods, such as the Fletcher–Reeves or the Polak–Ribiere methods [34, 53], or variable metric methods, such as the Broyden–Fletcher–Goldfarb–Shanno method [5, 58], or even Newton's method [6, 39], are computationally intensive for FNNs with several hundred weights: derivative calculations as well as subminimization procedures (for the case of nonlinear conjugate gradient methods) and approximations of various matrices (for the case of variable metric and quasi-Newton methods) are required. Furthermore, it is not certain that the extra computational cost speeds up the minimization process for nonconvex functions when far from a minimizer, as is usually the case with the neural network training problem [5, 9, 35]. Thus, the development of improved gradient-based BP algorithms receives significant attention of neural network researchers and practitioners.

The training algorithm introduced in this paper does not use a user–defined initial learning rate, instead it self–determinates the search direction and the learning rates at each epoch. It provides stable learning and robustness to oscillations. The paper is organized as follows. In Section 2 the class of adaptive learning algorithms that employ a different learning rate for each weight is presented and the advantages as well as the disadvantages of these algorithms are discussed. The new algorithm is introduced in Section 3 and its convergence properties are investigated in Section 4. Experimental results are presented in Section 5 to evaluate and compare the performance of the new algorithms with several other BP methods. The paper ends, in Section 6, with concluding remarks.

2 Adaptive Learning and the Error Surface

The eigensystem of the Hessian matrix can be used to determine the shape of the error function E in the neighborhood of a local minimizer [1, 18]. Thus, studying the sensitivity of the minimizer to small changes by approximating the error function with a quadratic one, it is known that, in a sufficiently small neighborhood of w^*, the directions of the principal axes of the corresponding elliptical contours (n–dimensional ellipsoids) will be given by the eigenvectors of $H(w^*)$, while the lengths of the axes will be inversely proportional to the square roots of the corresponding eigenvalues. Furthermore, a variation along the eigenvector corresponding to the maximum eigenvalue will cause the largest change in E, while the eigenvector corresponding to the minimum eigenvalue gives the least sensitive direction. Therefore, a value for the learning rate which yields a large variation along the eigenvector corresponding to the maximum eigenvalue may result in oscillations. On the other hand, a value for the learning rate which yields a small variation along the eigenvector corresponding to the minimum eigenvalue may result in small steps along this direction and thus, in a slight reduction of the error function. In general, a learning rate appropriate for any one weight direction is not necessarily appropriate for other directions.

Various adaptive learning algorithms with a different learning rate for each weight have been suggested in the literature [11, 18, 33, 41, 46, 49]. This approach allows us to find the proper learning rate that compensates for the small magnitude of the gradient in a flat direction in order to avoid slow convergence, and dampens a large weight change in a steep direction in order to avoid oscillations. Moreover, it exploits the parallelism inherent in the evaluation of $E(w)$ and $g(w)$ by the BP algorithm.

Following this approach Eq. (1) is reformulated to the following scheme:

$$w^{k+1} = w^k - \operatorname{diag}\{\eta_1^k, \ldots, \eta_n^k\}\, g(w^k). \tag{2}$$

The weight vector in Eq. (2) is not updated in the direction of the negative of the gradient; instead, an alternative adaptive search direction is obtained by taking into consideration the weight change, evaluated by multiplying the length of the search step, i.e. the value of the learning rate, along each weight direction by the partial derivative of $E(w)$ with respect to the corresponding weight, i.e. $-\eta_i \partial_i E(w)$. In other words, these algorithms try to decrease the error in each direction, by searching the local minimum with small weight steps. These steps are usually constraint by problem–dependent heuristic parameters, in order to ensure subminimization of the error function in each weight direction.

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, the training algorithms with an adaptive learning rate for each weight 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 heuristic parameters for properly tuning the adaptive learning rates [11, 18, 33, 41, 46, 49] and no guarantee is provided that the weight updates will converge to a minimizer of E. In certain cases the aforementioned methods, although originally developed for batch training, can be used for on-line training by minimizing a pattern–based error measure.

3 A Theoretical Derivation of the Adaptive Learning Process

An adaptive learning rate algorithm (see Eq. (2)) seeks for a minimum w^* of the error function and generates with every training epoch a discretized path in the nth dimensional weight space. The limiting value of this path, $\lim_{k\to\infty} w^k$, corresponds to a stationary point of $E(w)$. This path depends on the values of the learning rates chosen in each epoch. Appropriate learning rates help to avoid convergence to a saddle point or a maximum. In the framework of Eq. (2) the learning process can theoretically be interpreted as follows.

Starting from an arbitrary initial weight vector $w^0 \in \mathcal{D}$ (a specific domain of E), the training algorithm subminimizes, at the kth epoch, in parallel, the n one–dimensional function:

$$E(w_1^k, \ldots, w_{i-1}^k, w_i, w_{i+1}^k, \ldots, w_n^k). \tag{3}$$

First, each function is minimized along the direction i and the corresponding subminimizer $\hat{w}_i$ is obtained. Obviously for this $\hat{w}_i$

$$\partial_i E(w_1^k, \ldots, w_{i-1}^k, \hat{w}_i, w_{i+1}^k, \ldots, w_n^k) = 0. \tag{4}$$

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

$$w_i^{k+1} = \hat{w}_i. \tag{5}$$

In order to be consistent with Eq. (2) only a single iteration of the one–dimensional method in each weight direction is proposed. 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, the computational effort for the subminimization method 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 algorithm is away from a minimizer w^* [37]. Thus, we propose to obtain $\hat{w}_i$ by minimizing Eq.(3) with one iteration of a minimization method.

The problem of minimizing the error function E along the ith direction

$$\min_{\eta_i \geq 0} E(w + \eta_i \cdot e_i) \tag{6}$$

is equivalent to the minimization of the one–dimensional function

$$\phi_i(\eta) = E(w + \eta_i e_i). \tag{7}$$

Since we have n directions we consider n one–dimensional functions $\phi_i(\eta)$. Note that according to experimental work [61] these functions can be approximated for certain learning tasks with quadratic functions in the neighborhood of $\eta \geq 0$. In general, we can also use the formulation $\phi(\eta) = E(w + \eta d)$, where d is the search direction vector and $\phi^{'}(\eta) = \nabla E(w + \eta d)^{\top} d$.

In our case, we want at the kth epoch to find the learning rate η_i that minimizes $\phi_i(\eta)$ along the ith direction. Since $E(w) \geq 0$, $\forall w \in \mathbb{R}^n$ then the w^*, such that $E(w^*)$, minimizes $E(w)$. Thus, by applying one iteration of the Newton method to the one–dimensional equation $\phi_i(\eta) = 0$ we obtain:

$$\eta_i^1 = \eta_i^0 - \frac{\phi_i(\eta^0)}{\phi_i^{'}(\eta^0)}. \tag{8}$$

But $\eta_i^0 = 0$ and $\phi_i^{'}(\eta^0) = g(w)^{\top} d_i$. Since, $d_i = e_i$ the Eq.(8) is reformulated as

$$\eta_i = -\frac{E(w^k)}{\partial_i E(w^k)}. \tag{9}$$

This is done at the kth epoch in parallel, for all weight directions to evaluate the corresponding learning rates. Then, Eq.(5) takes the form

$$w_i^{k+1} = w_i^k - \frac{E(w^k)}{\partial_i E(w^k)}. \tag{10}$$

Eq.(10) constitutes the weight update formula of the new BP training algorithm with an adaptive learning rate for each weight.

The iterative scheme (10) 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 (10) to escape from flat regions with high error values, which are located far from a desired minimizer.

4 Convergence Analysis

First, we recall two concepts which will be used in our convergence analysis.

1) The Property A^{π}: Young [60] has discovered a class of matrices described as having *property A* that can be partitioned into block–tridiagonal form, possibly after a suitable permutation. In Young's original presentation, the elements of a matrix $A = [a_{ij}]$ are partitioned into two groups. In general, any partitioning of an n–dimensional vector $x = (x^{(1)}, \ldots, x^{(m)})$ into block components $x^{(p)}$ of dimensions n_p, $p = 1, \ldots, m$ (with $\sum_{p=1}^{m} n_p = n$) is uniquely determined by a partitioning $\pi = \{\pi_p\}_{p=1}^{m}$ of the set of the first n integers, where π_p contains the integers $s_p + 1, \ldots, s_p + n_p$, $s_p = \sum_{j=1}^{k-1} n_j$. The same partitioning π also induces a partitioning of any $n \times n$ matrix A into block matrix components A_{ij} of dimensions $n_i \times n_j$. Note that the matrices A_{ii} are square.

Definition 1 [3]: The matrix A has the property A^{π} if A can be permuted by $PAP^{\top}$ into a form that can be partitioned into block–tridiagonal form, that is,

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

where the matrices D_i, $i = 1, \ldots, r$ are nonsingular.

2) The Root–convergence factor: It is useful for any iterative procedure to have a measure of the rate of its convergence. In our case, we are interested in how fast the weight update equation (10), denoted $\mathcal{P}$, converge to w^*. A measure of the rate of its convergence is obtained by taking appropriate roots of successive errors. To this end we use the following definition.

Definition 2 [37]: Let $\{w^k\}_{k=0}^{\infty}$ be any sequence that converges to w^*. Then the number

$$R\{w^k\} = \lim_{k \to \infty} \sup \|w^k - w^*\|^{1/k}, \tag{11}$$

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

$$R(\mathcal{P}, w^*) = \sup\{R\{w^k\}; \{w^k\} \in C(\mathcal{P}, w^*)\}, \tag{12}$$

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

Our convergence analysis consists of two parts. In first part results regarding the local convergence properties of the algorithm are presented. In the second part appropriate conditions are proposed to guarantee the global convergence of the algorithm, i.e. the convergence to a minimizer from any starting point.

4.1 Local Convergence

In the first part, which concerns the local convergence of the algorithm 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. Furthermore, it is interesting to evaluate the asymptotic rate of convergence of the algorithm, i.e. the speed of the algorithm as it convergence to a minimizer. Of course this is not necessarily related to its convergence speed when it is away from the minimizer.

Theorem: Let $E : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ 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 $\mathcal{P}$ converges to w^* which minimizes E and $R(\mathcal{P}, w^*) < 1$.

Proof: Clearly, 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 [37]). Finding such a point is equivalent to obtaining the corresponding solution $w^* \in \mathcal{D}$ of the following system:

$$\nabla E(w) = \Theta^n, \tag{13}$$

by applying the nonlinear Jacobi process [37] and employing any one–dimensional method for the subminimization process [15, 37, 42, 43].

Now, consider the decomposition of $H(w^*)$ into its diagonal, strictly lower–triangular and strictly upper–triangular parts

$$H(w^*) = D(w^*) - L(w^*) - L^\top(w^*). \tag{14}$$

Since, $H(w^*)$ is symmetric and positive definite, then $D(w^*)$ is positive definite [55]. Moreover, since $H(w^*)$ has the property A^π, the eigenvalues of

$$\Phi(w^*) = D(w^*)^{-1} \left[L(w^*) + L^\top(w^*) \right], \tag{15}$$

are real and $\rho(\Phi(w^*)) = \varrho < 1$ [3]; 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 method (10) such that $\lim_{k\to\infty} w^k = w^*$ and $R(\mathcal{P}, w^*) = \varrho < 1$ [37, 44, 57]. Thus the Theorem is proved.

The proof of Theorem 1 shows that the asymptotic rate of convergence of the algorithm is not enhanced if one takes more than one iteration of the one–dimensional minimization method. Instead, the asymptotic rate of convergence only depends on the spectral radius ρ. Note also that this theorem is applicable to any training algorithm that adapts a different learning rate for each weight using one–dimensional subminimization methods and does not employ heuristics.

The local convergence analysis of the new algorithm was developed under appropriate assumptions and provides useful insight into the new method. However, in practice, neural network users want a guarantee that the training algorithm will reduce the error at each epoch and that the error will not fluctuate. Particularly, neural network practitioners are interested in techniques will satisfy the above mentioned requirements when the inital weights are far from the neighborhood of a minimizer. Unfortunately, it is well known that far from the neighborhood of a minimizer, the error function has broad flat regions adjoined with narrow steep ones. It is possible that this kind of morphology of the error function will cause the iterative scheme (10) to create very large learning rates, due to the small values of the denominator, pushing the neurons into saturation and thus the algorithm will exhibit pathological convergence behavior.

Therefore we want that the iterative scheme (10) will generate weight iterates that achieve a sufficient reduction in the error function at each epoch. Only these weight iterates will be

considered to 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 way 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 global convergence of the algorithm.

4.2 Globally Convergent Algorithms

In order to ensure global convergence of the adaptive algorithm, i.e. convergence to a local minimizer of the error function from any starting point, the following assumptions are needed [10, 24]:

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

$$\|\nabla E(w) - \nabla E(v)\| \leq L\|w - v\|, \tag{16}$$

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 the algorithm 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 and Price [37], are used to formulate the above ideas and to define a criterion of acceptance of any weight iterate:

$$E(w^{k+1}) - E(w^k) \leq \sigma_1 \nabla E(w^k)^\top \eta^k, \tag{17}$$

$$\nabla E(w^{k+1})^\top \nabla E(w^k) \geq \sigma_2 \nabla E(w^k)^\top \eta^k, \tag{18}$$

where $0 < \sigma_1 < \sigma_2 < 1$ and η^k is the column vector $(\eta_1^k, \ldots, \eta_n^k)$. Thus, by using appropriate values for the learning rates we seek to satisfy Conditions (17)–(18): the first condition ensures that using η_i^k, $i = 1, \ldots, n$, the error function is reduced with every epoch of the algorithm and the second condition prevents η_i^k, $i = 1, \ldots, n$, from becoming too small. Furthermore, these conditions can be used to enhance the new training algorithm with tuning techniques that are able to handle arbitrary large learning rates. Note that the value $\sigma_1 = 0.5$ is usually suggested in the literature [2, 35].

A simple technique to tune the length of the minimization step, so that it satisfies Conditions (17)–(18) at each epoch, is to decrease the learning rates by a reduction factor $1/q$, where $q > 1$ [37]. This has the effect that each η_i is decreased by the largest number in the sequence $\{q^{-m}\}_{m=1}^{\infty}$, so that the Condition (17) is satisfied. We remark here that the selection of q is not critical for successful learning, however it has an influence on the number of error function evaluations required to obtain an acceptable weight vector. Thus, some training problems respond well to one or two reductions in the learning rates by modest amounts (such as $1/2$) and others require many such

reductions, but might respond well to a more aggressive learning rate reduction (for example by factors of 1/10, or even 1/20). On the other hand, reducing η_i too much can be costly since the total number of epochs will be increased. Consequently, when seeking to satisfy the Condition (17) it is important to ensure that the learning rates are not reduced unnecessarily so that the Condition (18) is not satisfied. Since, in the algorithm, the gradient vector is known only at the beginning of the iterative search for an acceptable weight vector, the Condition (18) cannot be checked directly (this task requires additional gradient evaluations at each epoch of the training algorithm), but is enforced simply by placing a lower bound on the acceptable values of each η_i. This bound on the learning rates has the same theoretical effect as Condition (18) and ensures global convergence [10, 52].

Another approach to perform learning rates reduction is to estimate the appropriate reduction factor at each iteration. 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 are suggested that exploit the available information about the error function. Relative techniques have been proposed by Dennis and Schnabel [10] and Battiti [4].

5 Applications

In this section we give comparative results for five batch training algorithms in 1000 simulation runs: Back-propagation with constant learning rate (BP); Back-propagation with constant learning rate and constant Momentum [48], named BPM ; Adaptive Back-propagation with adaptive momentum (ABP) proposed by Vogl [56]; Back-propagation with an adaptive learning rate for each weight proposed by Silva and Almeida [49], named SA; Back-propagation with a self–determined learning rate for each weight (BPS).

The selection of initial weights is very important in FNN training [59]. A well known initialization heuristic for FNNs is to select the weights with uniform probability from an interval (w_{min}, w_{max}), where usually $w_{min} = -w_{max}$. However, if the initial weights are very small the backpropagated error is so small that practically no change takes place for some weights and more iterations are necessary to decrease the error [47, 48]. In the worst case the error remains constant and the learning stops in an undesired local minimum [27]. On the other hand, very large values of weights speed up learning but they can lead to saturation and to flat regions of the error surface where training is considerably slow [28, 30, 47].

A common choice for the range of the initial weights is the interval $(-1, +1)$ (see [21, 22, 40, 46]). This interval has been used to randomly choose initial weights for our experiments. The values of the heuristic learning parameters used in each problem are shown in Table 1. The initial learning rate has been carefully chosen so that the BP algorithm indicates rapid convergence without oscillating towards a global minimum and has been kept the same for all the algorithms tested. Then all other heuristics have been tuned. For the momentum term m we have tried 9 different values ranging from 0.9 to 0.1. Much effort has been made to properly tune the learning rate increment and decrement factors, η_{inc}, u and η_{dec}, d, respectively. To be more specific, various different values, up to 2, have been tested for the learning rate increment factor, while different values between 0.1 and 0.9 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 [56] and indeed it has been found to work better than others tested. All the combinations of these parameter values have been tested on 3 simulation runs, starting from the same initial weights, to find the best available in terms of accelarated training.

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 SA one gradient evaluation and one error function evaluation are necessary at each epoch; for BPS there is a number of additional

Table 1: Learning parameters used in the experiments

Algorithm	Texture classification	Vowel spotting
BP	$\eta_0 = 0.001$	$\eta_0 = 0.0034$
BPM	$\eta_0 = 0.001$	$\eta_0 = 0.0034$
	$m = 0.9$	$m = 0.7$
ABP	$\eta_0 = 0.001$	$\eta_0 = 0.0034$
	$m = 0.9$	$m = 0.1$
	$\eta_{inc} = 1.05$	$\eta_{inc} = 1.07$
	$\eta_{dec} = 0.5$	$\eta_{dec} = 0.8$
	$ratio = 1.04$	$ratio = 1.04$
SA	$\eta_0 = 0.001$	$\eta_0 = 0.0034$
	$u = 1.05$	$u = 1.3$
	$d = 0.6$	$d = 0.7$
BPS	$*$	$*$

$*$ no heuristics required

error function evaluations when the Goldstein/Armijo condition (17) 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 [34] suggests to count a gradient evaluation three times more than an error function evaluation).

5.1 Texture Classification Problem

The first experiment is a texture classification problem. A total of 12 Brodatz texture images [7]: 3, 5, 9, 12, 15, 20, 51, 68, 77, 78, 79, 93 (see Figure 1 in [31]) of size 512×512 is acquired by a scanner at 150dpi. From each texture image 10 subimages of size 128×128 are randomly selected, and the co–occurence method, introduced by Haralick [20] is applied. In the co–occurence method, the relative frequencies of gray–level pairs of pixels at certain relative displacements are computed and stored in a matrix. As suggested by other researchers [36, 54], the combination of the nearest neighbor pairs at orientations 0^o, 45^o, 90^o and 135^o are used in the experiment. 10 sixteenth-dimensional training patterns are created from each image. A 16–8–12 FNN (224 weights, 20 biases) with sigmoid activations is trained to classify the patterns to 12 texture types. The termination condition is a classification error $CE < 3\%$.

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 simulations that converge to a global minimum. Obviously, the number of gradient evaluations is equal to the number of error function evaluations for the BP, the BPM, the ABP and the SA.

The results of Tables 2 suggest that BPS 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 the best performance of all methods tested, however it requires fine tuning five heuristic parameters. SA is also faster than BPS needing only tuning three heuristic parameters, but has smaller percentage of success than BPS.

The successfully trained FNNs are tested for their generalization capability using patterns from 20 subimages of the same size randomly selected from each image. To evaluate the generalization

Table 2: 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
SA	704	2112.6	82/18750	704	2112.6	82/18750	88
BPS	791	512.3	417/5318	2185	1405.9	1116/14006	100

performance of the FNN the *max* rule is used, i.e. a test pattern is considered to be correctly classified if the corresponding output neuron has the greatest value among the output neurons. The average success rate classification for each algorithm was: BP=90%, BPM=90%, ABP=93.5%, SA=93.6%, BPS=93%. These results confirm that increased training speed achieved by ABP, SA and BPS affect by no means their generalization capability.

5.2 Vowel Spotting Problem

In the second experiment a 15–15–1 FNN (240 weights and 16 biases), based on neurons of hyperbolic tangent activations, is used for vowel spotting. Vowel spotting provides a preliminary acoustic labeling of speech, which can be very important for both speech and speaker recognition procedures. The speech signal, coming from a high quality microphone in a very quiet environment, is recorded, sampled at 16KHz and digitized at 16–bit precision. The sampled speech data is then 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 a comparative study of several speech parameters for speaker independent speech recognition and speaker recognition purposes [50]. The PLP analysis includes: spectral analysis, critical–band spectral resolution, equal–loudness pre–emphasis, intensity–loudness power law, autoregressive modeling. It results in a 15th dimensional feature vector for each frame.

The FNN is trained as speaker independent using labelled training data from a large number of speakers from the TIMIT database [14] and classifies the feature vectors into $\{-1, +1\}$ for the non–vowel/vowel model. The network is part of a text-independent speaker identification and verification system which is based on using only the vowel part of the signal [13].

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 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 [12, 13].

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. In this examination, a small set of rules is used. These rules are based on the principle that the cost of rejecting a vowel is much less than the cost of mistakenly accepting a non-vowel and considering it as a vowel and concern the distance, the duration and the amplitude of the responses of the FNN, [12, 13, 51]. The results of the training phase are shown in Table 4, where the abbreviations are as in Table 2.

The resuts of Table 3 show that BPS compares favourably on the number of gradient evaluations

Table 3: 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
SA	250	157.5	118/951	250	157.5	118/951	36
BPS	362	358.5	96/1580	1756	1692.1	459/7396	64

to BP, BPM and ABP. ABP exhibits the slowest convergence but has the most reliable performance regading the number of successful simulations. Note that SA provides the fastest training, but also exhibits the worst percentage of success. On the other hand, it has the best performance regarding the generalization performance (see Table 5). BPS also improves the error rate achieved by the BP by 1%. Thus, the average performance of the BPS is satisfactory.

With regards to generalization, the adaptive methods provide comparable performance which on the the average improves of the error rate percentage achieved by BP–tranined FNNs by 2%. However, BPM trained FNNs did not provide any improvement.

6 Conclusions

The papers analyzed adaptive learning with a different learning rate for each weight as a nonlinear Jacobi process. Along this line an algorithm with a self-determined learning rate for each weight has been presented. A model for the analysis of the local and global convergence of the algorithm has been proposed. It was shown that the asymptotic rate of convergence of the training algorithm does not depend on the multi–step subminimization methods. However this result is not necessarily related with the convergence speed when the algorithm is away from the minimizer. With regards to global convergence, the Goldstein/Armijo conditions have been appropriately adapted to secure the convergence of the algorithm.

The algorithm compares satisfactory with other popular training algorithms without using highly problem–dependent heuristic parameters. Its performance in the two reported experiments is promising.

In a future contribution we will focus on the effects of property A^{π} in FNN training as well as on the issue of global efficiency, or global rate of convergence, of the training algorithm, which is related to the estimation of the error function reduction at every epoch.

References

[1] G. S. Androulakis, G. D. Magoulas and M. N. Vrahatis, Geometry of learning: visualizing the performance of neural network supervised training methods, *Nonlinear Analysis,Theory, Methods and Applications*, 30 (1997), 4539–4544.

[2] L. Armijo, Minimization of functions having Lipschitz continuous first partial derivatives, *Pacific Journal of Mathematics*, 16 (1966), 1–3.

[3] O. Axelsson, *Iterative Solution Methods.* Cambridge University Press, New York, 1996.

[4] R. Battiti, Accelerated backpropagation learning: two optimization methods, *Complex Systems*, 3 (1989), 331–342.

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

[6] S. Becker and Y. Le Cun, Improving the convergence of the back–propagation learning with second order methods. In *Proceedings of the 1988 Connectionist Models Summer School*, D.S. Touretzky, G.E. Hinton, and T.J. Sejnowski eds., 29-37, Morgan Koufmann, San Mateo, CA, 1988.

[7] P. Brodatz, *Textures - a photographic album for artists abd designers*, Dover, NY, 1966.

[8] L. W. Chan and F. Fallside, An adaptive training algorithm for back–propagation networks, *Computers Speech and Language*, 2 (1987), 205–218.

[9] J. E. Dennis and J. J. Moré, Quasi-Newton methods, motivation and theory, *SIAM Review*, 19 (1977), 46–89.

[10] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and nonlinear equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[11] S. E. Fahlman, Faster-learning variations on back–propagation: an empirical study, *Proceedings of the 1988 Connectionist Models Summer School*, D.S. Touretzky, G.E. Hinton, and T.J. Sejnowski, eds., 38–51, Morgan Kaufmann, San Mateo, CA, 1989.

[12] N. Fakotakis and J. Sirigos, A high-performance text-independent speaker recognition system based on vowel spotting and neural nets, *Proceedings of the IEEE International Conference on Acoustic Speech and Signal Processing*, 2, 661–664, 1996.

[13] N. Fakotakis and J. Sirigos, A high-performance text-independent speaker identification and verification system based on vowel spotting and neural nets. To appear in IEEE Trans. Speech and Audio processing.

[14] W. Fisher, V. Zue, J. Bernstein and D. Pallet, An acoustic-phonetic data base, *Journal of Acoustical Society of America*, Suppl. A, 81 (1987), 581–592.

[15] P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization*, Academic Press, NY., 1981.

[16] A. A. Goldstein, Cauchy's method of minimization, *Numerische Mathematik*, 4 (1962), 146–150.

[17] A.A. Goldstein, On steepest descent, *SIAM J. Control*, 3 (1965), 147–151.

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

[19] E. M. Johansson, F. U. Dowala, and D. M. Goodman, Back–propagation learning for multilayer feed forward neural networks using the conjugate gradient method, *Int. J. of Neural Systems*, 2 (1991), 291–302.

[20] R. Haralick, K. Shanmugan and I. Dinstein, Textural features for image classification, *IEEE Trans. System, Man and Cybernetics*, 3 (1973), 610-621.

[21] Y. Hirose, K. Yamashita and S. Hijiya, Back–propagation algorithm which varies the number of hidden units, *Neural Networks*, 4 (1991), 61–66.

[22] M. Hoehfeld and S. E. Fahlman, Learning with limited numerical precision using the cascade-correlation algorithm, *IEEE Trans. on Neural Networks*, 3 (1992), 602–611.

[23] H.-C. Hsin, C.-C. Li, M. Sun and R. J. Sclabassi, An adaptive training algorithm for back–propagation neural networks, *IEEE Transactions on System, Man and Cybernetics*, 25 (1995), 512–514.

[24] C.T. Kelley, *Iterative methods for linear and nonlinear equations*, SIAM publications, Philadelphia, 1995.

[25] C.M. Kuan and K. Hornik, Convergence of learning algorithms with constant learning rates, *IEEE Trans. Neural Networks*, 2 (1991), 484–488.

[26] S. Y. Kung, K. Diamantaras, W. D. Mao, J. S. Taur, Generalized perceptron networks with nonlinear discriminant functions, *Neural Networks Theory and Applications*, Mammone R.J. and Zeevi Y.Y., eds., 245–279, Academic Press, Boston, 1991.

[27] Y. Lee, S.-H. Oh and M. W. Kim, An analysis of premature saturation in backpropagation learning, *Neural Networks*, 6 (1993), 719–728.

[28] P.J.G. Lisboa and S. J. Perantonis, Complete solution of the local minima in the XOR problem, *Network*, 2 (1991), 119–124.

[29] R. Liu, G. Dong and X. Ling, A convergence analysis for neural networks with constant learning rates and non–stationary inputs, *Proceedings of the 34th Conf. on Decision and Control*, New Orleans, 1278-1283, 1995.

[30] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis, A new method in neural network supervised training with imprecision, *Proceedings of the IEEE 3rd International Conference on Electronics, Circuits and Systems*, 287–290, 1996.

[31] G. D. Magoulas, M. N. Vrahatis and G. S. Androulakis, Effective back–propagation with variable stepsize, *Neural Networks*, 10 (1997), 69–82.

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

[33] A.A. Minai and R. D. Williams, Back-propagation heuristics: a study of the extended delta-bar-delta algorithm, *Proceedings of the IEEE Int. Joint Conf. on Neural Networks*, San Diego, 1990, 595-600.

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

[35] J. Nocedal, Theory of algorithms for unconstrained optimization, *Acta Numerica*, 199–242, 1991.

[36] P. P. Ohanian and R. C. Dubes, Performance evaluation for four classes of textural features, *Pattern Recognition*, 25, 8 (1992), 819–833.

[37] J.M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, NY, 1970.

[38] A.G. Parlos, B. Fernandez, A. F. Atiya, J. Muthusami and W. K. Tsai, An accelerated learning algorithm for multilayer perceptron networks, *IEEE Trans. on Neural Networks*, 5 (1994), 493–497.

[39] D. B. Parker, Optimal Algorithms for Adaptive Networks: Second Order Back–Propagation, Second Order Direct Propagation, and Second Order Hebbian Learning, *Proceedings of the IEEE International Conference on Neural Networks*, 2, 593–600, 1987.

[40] B. Pearlmutter, Gradient descent: second–order momentum and saturating error, *Advances in Neural Information Processing Systems 4*, J.E. Moody, S.J. Hanson, and R.P. Lippmann, eds., 887–894, Morgan Kaufmann, San Mateo, CA, 1992.

[41] M. Pfister and R. Rojas, Speeding-up backpropagation -A comparison of orthogonal techniques, *Proceedings of the Joint Conference on Neural Networks*, Nagoya, Japan, 517–523, 1993.

[42] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. F. Flannery, *Numerical recipes in C*, Cambridge University Press, New York, 1992.

[43] A. Ralston and P. Rabinowitz, *A first course in numerical analysis*, McGraw-Hill, New York, 1978.

[44] W.C. Reinboldt, *Methods for solving systems of nonlinear equations*, *SIAM Publications*, Philadelphia, Pennsylvania, 1974.

[45] M. Riedmiller and H. Braun, A direct adaptive method for faster backpropagation learning: the Rprop algorithm, *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, 586–591, 1993.

[46] M. Riedmiller, Advanced supervised learning in multi-layer perceptrons - From backpropagation to adaptive learning algorithms, *International Journal of Computer Standards and Interfaces*, Special issue on Neural Networks, 5, 1994.

[47] A. K. Rigler, J. M. Irvine and T. P. Vogl, Rescaling of variables in backpropagation learning, *Neural Networks*, 4 (1991), 225–229.

[48] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning Internal Representations by Error Propagation, D. E. Rumelhart, J. L., and McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1, pp. 318–362. MIT Press, Cambridge, Massachusetts, 1986.

[49] F. Silva and L. Almeida, Acceleration techniques for the back–propagation algorithm, *Lecture Notes in Computer Science*, 412, 110–119. Springer-Verlag,Berlin, 1990.

[50] J. Sirigos, N. Fakotakis and G. Kokkinakis, A comparison of several speech parameters for speaker independent speech recognition and speaker recognition, *Proceedings of the 4th European Conference of Speech Communications and Technology*, 1995.

[51] J. Sirigos, V. Darsinos, N. Fakotakis and G. Kokkinakis, Vowel/non-vowel decision using neural networks and rules, *Proceedings of the 3rd IEEE International Conference on Electronics, Circuits, and Systems*, 510–513, 1996.

[52] G. A. Shultz, R. B. Schnabel and R. H. Byrd, A family of trust region based algorithms for unconstrained minimization with strong global convergence properties, University of Colorando, Computer Science TR CU-CS-216-82, 1982.

[53] P. P. Van der Smagt, Minimization Methods for training feedforward neural networks, *Neural Networks*, 7 (1994), 1–11.

[54] J. Strang and T. Taxt, Local frequency features for texture classification, *Pattern Recognition*, 27, 10 (1994), 1397–1406.

[55] R. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1962.

[56] T. P. Vogl, J. K. Mangis, J. K. Rigler, W. T. Zink and D. L. Alkon, Accelerating the convergence of the back-propagation method, *Biological Cybernetics*, 59 (1988), 257–263.

[57] R. G. Voigt, Rates of convergence for a class of iterative procedures, *SIAM J. Numer. Anal.*, 8 (1971), 127–134.

[58] R. L. Watrous, Learning Algorithms for Connectionist Networks: Applied Gradient of Nonlinear Optimization, *Proceedings of the IEEE International Conference on Neural Networks*, 2, 619–627, 1987.

[59] L. F. Wessel and E. Barnard, Avoiding false local minima by proper initialization of connections, *IEEE Trans. Neural Networks*, 3 (1992), 899–905.

[60] D. Young, Iterative methods for solving partial difference equations of elliptic type, *Trans. American Mathematical Society*, 76 (1954), 92–111.

[61] X.-H. Yu, G.-A. Chen and S.-X. Cheng, Dynamic learning rate optimization of the backpropagation algorithm, *IEEE Trans. Neural Networks*, 6 (1995), 669–677.