International Journal of Bifurcation and Chaos, Vol. 16, No. 7 (2006) 1929–1950
© World Scientific Publishing Company

ADAPTIVE ALGORITHMS FOR NEURAL NETWORK SUPERVISED LEARNING: A DETERMINISTIC OPTIMIZATION APPROACH

GEORGE D. MAGOULAS
School of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London WC1E 7HX, UK

MICHAEL N. VRAHATIS
Computational Intelligence Laboratory, Department of Mathematics, University of Patras Artificial Intelligence Research Center, University of Patras, GR-26110 Patras, Greece

Received January 5, 2005; Revised March 15, 2005

Networks of neurons can perform computations that even modern computers find very difficult to simulate. Most of the existing artificial neurons and artificial neural networks are considered biologically unrealistic, nevertheless the practical success of the backpropagation algorithm and the powerful capabilities of feedforward neural networks have made neural computing very popular in several application areas. A challenging issue in this context is learning internal representations by adjusting the weights of the network connections. To this end, several first-order and second-order algorithms have been proposed in the literature. This paper provides an overview of approaches to backpropagation training, emphazing on first-order adaptive learning algorithms that build on the theory of nonlinear optimization, and proposes a framework for their analysis in the context of deterministic optimization.

Keywords: Adaptive learning; backpropagation training; feedforward neural networks; supervised training; unconstrained optimization.

1. Introduction

Neural computing is inspired by information processing and computation in the brain. The brain is a highly complex, nonlinear, and parallel information-processing system [Haykin, 1994]. The feature that has made the brain such an appealing area of interest is its ability to learn from experience. Learning is a continuous process, where each day our brain re-structures itself to become accustomed to new surroundings and builds its own rules through past experiences. The human brain is composed of billions of cells called neurons. The cell body–soma contains the nucleus; networks of nerve fibre called dendrites are connected to the soma. Each cell consists of a single long fibre called an axon which branches out into many strands and substrands which are eventually connected to the dendrites of other neurons through synaptic junctions called synapses.

Information processing takes place by feeding the neuron with input through its dendrites; this information could either be raw data from the outside world or processed data from another neuron (see Fig. 1). The neuron processes information it receives and sends out electrical activity through the axon, when this activity reaches the synapses it is converted into electrical effects that either excite or inhibit activity in connected neurones. When a

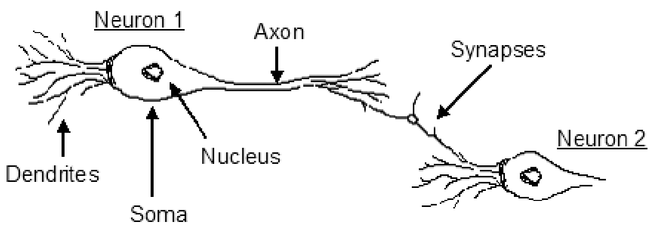


Fig. 1. Biological neurons.

neuron receives excitatory input it sends electrical activity down its axon.

The theoretical basis of neural networks was developed in 1943 by the neurophysiologist Warren McCulloch of the University of Illinois and the mathematician Walter Pitts of the University of Chicago. In 1954 Belmont Farley and Wesley Clark of the Massachusetts Institute of Technology succeeded in running the first simple neural network.

Highly simplified models of the human brain have been proposed in the area of computational intelligence [Haykin, 1994]. Nodes, or artificial neurons, in these models are usually considered as simplified models of biological neurons (see Fig. 2). The neuron consists of a number of inputs x_n and a single output y. Connections between the inputs and the neuron represent dendrites each with a corresponding weight w_n that is applied to the input. Weights have the same effect as the synapses in a biological neuron.

In Fig. 2, each input of the artificial neuron corresponds to a single attribute or an output of another neuron. Inputs must be numerical. However in some circumstances, e.g. neural computing applied to mortgage assessment, the input may be qualitative such as the occupation of the applicant. In these cases the attribute must be transformed into numerical values during a pre-processing stage. Each input has an associated weight that represents the relative strength or the importance of the input. An initial value is assigned to each weight, but these values are not static. As the neuron is trained, as will be explained below, the weights are continuously adjusted to obtain a more accurate result and it is through these continuous adjustments that the neuron learns. A bias is simply added to the weighted sum of each node; it has a constant input of 1 and is updated during training like weights. The role of the summation function is to calculate the weighted average or activation level of all input elements, found by multiplying each input value by its weight and then totaling them for a weighted sum. The transfer or activation function of each processing neuron is a mathematical formula that determines the output y of the neuron. Its purpose is to prevent outputs from reaching very large values which can inhibit training. A number of transfer functions are used in practice, such as those shown in Fig. 3, i.e. the linear, nonlinear (sigmoid) and binary threshold functions. Lastly, the output y corresponds to the solution of a problem, and usually needs to be transformed into a format suitable as input to another neuron or as a piece of information that is understandable to the user.

A popular way to training an artificial neuron is by *error correction*: the difference between neuron's actual output y and the correct output t, as defined by the user, is calculated. This difference is also known as *learning error*. If the response at the output is incorrect then the neuron weights should be

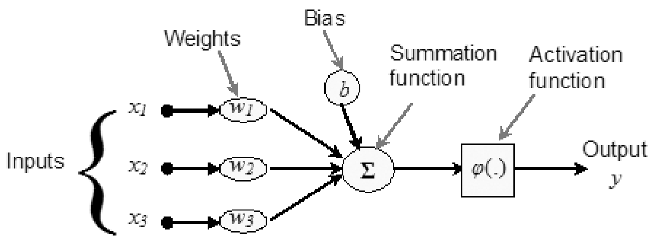
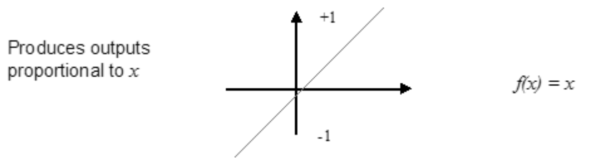
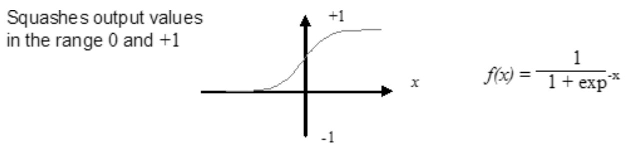


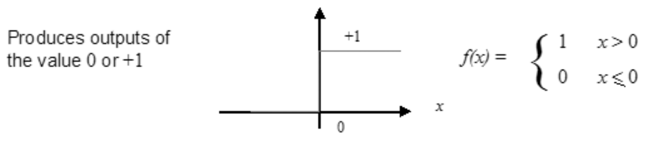
Fig. 2. Model of a single artificial neuron.



(a) Linear Transfer Function



(b) Sigmoid Transfer Function



(c) Threshold Transfer Function

Fig. 3. Popular transfer functions used in artificial neurons.

changed so that it is more likely to produce the correct response the next time that the input stimulus is presented. For example, this is achieved by changing the nth connection weight:

$$w_n^{\text{new}} = w_n^{\text{old}} + \Delta w_n, \tag{1}$$

where w_n^{new} is the updated value, w_n^{old} is the current value and Δw_n the estimated change needed to produce the correct response. This process of change is applied until the learning error reaches a suitable value.

This form of training is called *supervised learning* because it assumes an external teacher supplies the correct output t. As soon as the neuron is able to perform sufficiently well on additional test cases, it can be used to classify new inputs. Another form of training, which is adopted in practice is called *unsupervised learning*. In this approach there is no teacher that provides the correct output for each input data but inputs are grouped together by measuring the similarity among them rather than the error [Haykin, 1994].

At this point it is useful to provide a simple example in order to visualize the learning error during supervised training. Let us consider the case of a single neuron with two weights, w_1 and w_2 and sigmoid activation function $f(x) = (1+e^{-x})^{-1}$. This minimal architecture is trained to associate a set of eight inputs with a set of eight outputs that have been defined by the user. Training occurs by feeding the artificial neuron with each one of the inputs; producing the output and calculating the corresponding error. The overall error produced depends on the current values of the weights, W_1 and W_2, and is defined by the following objective function

$$E(W_1, W_2) = \sum_{p=1}^{8} (y_p - t_p)^2, \tag{2}$$

where p denotes the input stimulus, y_p denotes the actual output for input p and t_p denotes the correct output. Figure 4 illustrates the error function produced when $W_1 \in [-3, +3]$ and $W_2 \in [-5, +5]$. The objective function of Eq. (2) defines the sum-of-squared-error and is commonly used in neural networks learning.

The error function of Eq. (2) is not the only possible choice for the objective function. A variety of distance functions are available in the literature, such as the Minkowsky, Mahalanobis, Camberra, Chebychev, Quadratic, Correlation, Kendall's Rank Correlation and Chi-square distance metrics; the Context-Similarity measure; the Contrast Model; hyperectangle distance functions and others [Wilson & Martinez, 1997].

In general, the success of a learning system depends upon the adopted distance function [Wilson & Martinez, 1997]. For example, a function for measuring the network performance is the piecewise linear perceptron criterion function [Duda & Hart, 1973]

$$E(w) = -\sum_{z \in Z(w)} z^\top w, \tag{3}$$

where $Z(w)$ is the subset of samples misclassified by w (for these samples $z^\top w \leq 0$). Obviously if $Z(w)$ is empty then $E(w) = 0$; otherwise $E(w) > 0$. From a geometrical point of view this objective function tries to minimize the sum of the distances from the misclassified samples to the decision boundary

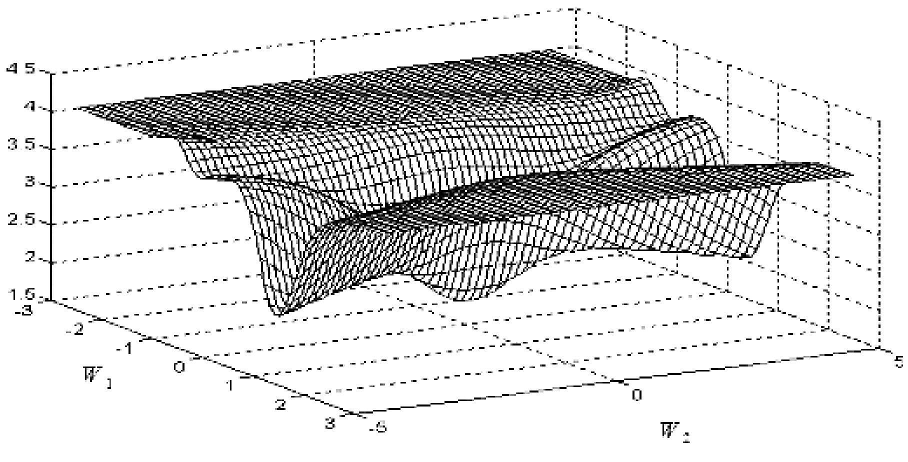


Fig. 4. Error surface of a single neuron.

but its gradient is not continuous. Another function of the same family that uses a continuous gradient is the quadratic function [Mays, 1964; Hassoun, 1995]:

$$E(w) = \sum_{z^\top w \leq b} \frac{(z^\top w - b)^2}{\|z\|^2}, \tag{4}$$

where b is a positive constant bias. In general, any differentiable function that minimizes the errors $e_p = y_p - t_p$, for all p, can be used.

The sum-of-squared-error function of Eq. (2) can be generalized to the well known *Minkowski-r* objective function [Hanson & Burr, 1988]:

$$E(w) = \sum_{p=1}^{P} |y_p - t_p|^r. \tag{5}$$

Note that Eq. (5) for $r = 1$ is also known as the *Manhattan distance*; for $r = \infty$ is the *Chebychev distance* and for $r = 2$ it reduces to the sum-of-squared-error function. One weakness of the sum-of-squared-error function is that it is influenced by the relative magnitude of the input attributes. Therefore, a process of normalization, the exact way of which depends on the nature of the problem and on domain knowledge, is often necessary. Nevertheless, the choice $r = 2$ has become very popular because it can be shown that under some assumptions it minimizes both the sum-of-squared-error and the probability of prediction error (maximum likelihood) [Hassoun, 1995].

During training, weights are repeatedly updated using Eq. (1) until the learning error is small enough. As will be explained in the next sections the way Δw_n is calculated influences the accuracy and quality of the solution, i.e. of the weight combination that makes the learning error small enough.

Artificial neurons can be connected together forming networks of artificial neurons that are called *Artificial Neural Networks* (ANNs). ANNs offer a powerful and distributed computing architecture equipped with significant abilities such as learning of highly nonlinear and multivariable relationships, adaptation to changing environments and learning of new data/concepts, self-organization, real-time (online) operation, and they can be implemented in parallel and in Very Large Scale Integrated Systems (VLSI) [Magoulas *et al.*, 2001; Plagianakos & Vrahatis, 2002; Magoulas *et al.*, 2004]. ANNs have already been successfully used in many real life applications, such as medical imaging, control, and decision making. They demonstrate considerable ability in handling

(a) *incompleteness*: missing parameter values;
(b) *incorrectness*: systematic or random noise in the data;
(c) *sparseness*: few and/or non-representable records available;
(d) *inexactness*: inappropriate selection of parameters for the given task.

Furthermore, ANNs perform *pattern matching* and exhibit *human-like characteristics* such as generalization, and robustness to noise.

The rest of the paper is organized as follows. The next section formulates the supervised training problem from an optimization perspective. Then, in Sec. 3, adaptive algorithms with a common learning rate for each weight are presented and their convergence is discussed in the framework of Armijo–Goldstein–Price conditions. The paper proceeds with a detailed review of first-order adaptive algorithms with a different adaptive learning for each weight. A framework for their analysis is presented in the context of the nonlinear Jacobi process. In Sec. 5, sources of imprecision that influence the success of the supervised training procedure are discussed and an approach based on the nonlinear successive overrelaxation is presented to handle imprecise information. Lastly, complexity issues relevant to supervised training are briefly discussed. Section 7 ends the paper with conclusions.

2. Formulation of the Supervised Training Problem

The training algorithm is the method used for updating the weights of Eq. (1). This paper focuses on a particular class of training algorithms that perform supervised training. In this type of learning, the desired output for each set of inputs is known and fed into the network. The difference between the desired and the actual output is used to calculate modifications to the weights that globally minimize this difference (see Fig. 5). The rapid computation of such a global minimum is a rather difficult task since, in general, the number of network weights is high and the corresponding nonconvex multimodal error function possesses multitudes of local minima and has broad flat regions adjoined with narrow steep ones.

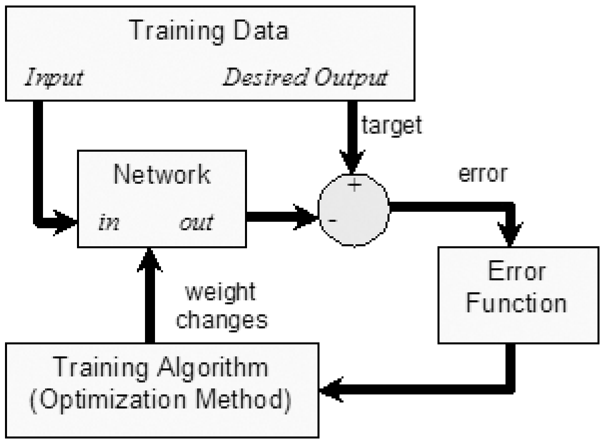


Fig. 5. Generic supervised training scheme.

Feedforward Neural Networks (FNNs) form the most popular category of ANNs. In FNNs neurons are organized in layers, where neurons of one layer are connected to each one of the neurons of the next layer, and inputs are propagated through the various layers of the network moving from the input layer to the output layer (for the architecture of a feedforward neural network see Fig. 6).

The operation of an FNN is usually based on the following equations:

$$\mathrm{net}_j^l = \sum_{i=1}^{n_{l-1}} w_{ij}^{l-1,l} y_i^{l-1}, \quad y_j^l = f(\mathrm{net}_j^l), \tag{6}$$

where net_j^l is for the jth neuron in the lth layer $(j = 2, \ldots, n_l)$, the sum of its weighted inputs. The weights from the ith node at the $(l-1)$ layer to the jth neuron at the lth layer are denoted by $w_{ij}^{l-1,l}$, y_j^l is the output of the jth neuron that belongs to the lth layer, and $f(\mathrm{net}_j^l)$ is the jth's node activation function.

If there is a fixed, finite set of input–output samples, the squared error over the training dataset, which contains P representative samples, is:

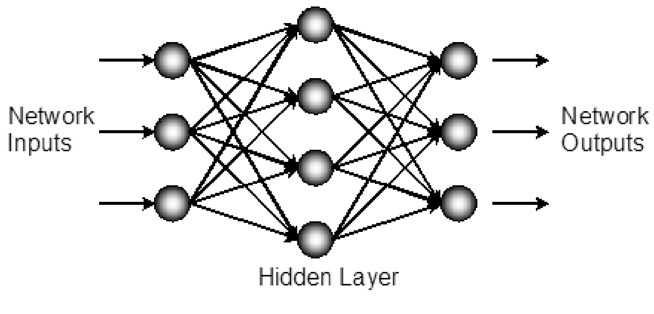


Fig. 6. Architecture of a feedforward neural network.

$$\begin{aligned} E(w) &= \sum_{p=1}^{P}\sum_{j=1}^{n_L}(y_{jp}^L - t_{jp})^2 \\ &= \sum_{p=1}^{P}\sum_{j=1}^{n_L}[f^L(\mathrm{net}_j^L + \theta_j^L) - t_{jp}]^2. \end{aligned} \tag{7}$$

This equation gives the general form of the FNN *error function* to be minimized, in which $t_{j,p}$ specifies the desired response at the jth output neuron for the stimulus p and $y_{j,p}^L$ is the output of the jth neuron at layer L that depends on the weights of the network, and f is a nonlinear activation function, such as the well known logistic function $f(x) = (1 + e^{-x})^{-1}$. The network weights can be expressed using vector notation $w \in \mathbb{R}^n$, as:

$$\begin{aligned} w = (\ldots, w_{ij}^{l-1,l}, w_{i+1\,j}^{l-1,l}, \ldots, w_{N_{l-1}\,j}^{l-1,l}, \\ \theta_j^l, w_{i\,j+1}^{l-1,l}, w_{i+1\,j+1}^{l-1,l}, \ldots)^\top, \end{aligned} \tag{8}$$

where θ_j^l denotes the bias of the jth neuron ($j = 1, \ldots, N_l$) at the lth layer ($l = 2, \ldots, L$), and n denotes the total number of weights and biases in the network.

A variety of approaches adapted from unconstrained optimization theory have been applied to solve this problem. The Back-Propagation (BP) algorithm [Rumelhart *et al.*, 1986] is widely recognized as a powerful tool for training FNNs using information from the first derivatives of the error function. In an attempt to use not only the gradient of the error function but also the second derivative to accelerate the learning process training algorithms that apply nonlinear conjugate gradient methods, such as the Fletcher–Reeves or the Polak–Ribiere methods [Møller, 1993; Van der Smagt, 1994], or variable metric methods, such as the Broyden–Fletcher–Goldfarb-Shanno method [Watrous, 1987; Battiti, 1992], or even Newton's method [Parker, 1987] have been proposed. However, these approaches 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. Moreover, approximations of the Hessian matrix made during training may be close to singular, or badly scaled, and as a consequence they might produce inaccurate results. 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 [Dennis & Moré, 1977; Nocedal, 1992; Battiti, 1992].

The popular BP algorithm [Rumelhart *et al.*, 1986] minimizes the error function using the following Steepest Descent (SD) method [Gill *et al.*, 1981] with constant learning rate η:

$$w^{k+1} = w^k - \eta\, g(w^k), \quad k = 0, 1, 2, \ldots \tag{9}$$

where the gradient vector $g(w^k) \equiv \nabla E(w^k)$ is usually computed by back–propagation of the error through the layers of the FNN (see [Rumelhart *et al.*, 1986]). In theory, the SD method requires the assumption that E is twice continuously differentiable on an open neighborhood $\mathcal{S}(w^0)$, where $\mathcal{S}(w^0) = \{w\colon E(w) \leq E(w^0)\}$ 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 $\mathcal{S}(w^0)$ where H denotes the Hessian matrix of E [Cauchy, 1847; Goldstein, 1962, 1965].

In practice, checking the validity of the above condition requires calculating the Hessian for each point in the level set. However, the manipulation of the full Hessian matrix, H, is too expensive in computation and storage for FNNs with several hundred weights [Becker & Le Cun, 1988]. To alleviate this situation, Le Cun [Le Cun *et al.*, 1993] proposed a technique that employs weight perturbations to estimate the principle eigenvalues and eigenvectors of the Hessian without calculating the full matrix H on line. In [Le Cun *et al.*, 1993], the largest eigenvalue of the Hessian is mainly determined by the FNN architecture, the initial weights and by short-term low-order statistics of the training data. Le Cun's technique could be used to determine an initial η at an additional cost in the number of presentations of the training set during early training. Another approach proposed in the literature is to consider a learning rate that is proportional to the inverse of the Lipschitz constant which, unfortunately, is not easily available [Armijo, 1966; Magoulas *et al.*, 1997a, 1997b]. So despite these efforts, obtaining convergence of BP training algorithms utilizing a constant learning rate is still considered very difficult [Kuan & Hornik, 1991; Liu *et al.*, 1995] and practitioners usually chose $0 < \eta < 1$ to ensure that successive weight updates will not lead to missing a minimum of the error surface.

A practical way to ensure global convergence using an arbitrary learning rate, i.e. convergence to a local minimizer of the error function from any starting point, is to use the Armijo–Goldstein–Price conditions as explained below.

To this end, the following assumptions are needed [Dennis & Schnabel, 1983; Kelley, 1995]:

(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 $v \in \mathbb{R}^n$, the gradient $g(x)$ of E satisfies the Lipschitz condition $\|g(w) - g(v)\| \leq L\|w - v\|$, 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, via the curvature of E, and to ensure that the first derivatives are continuous [Dennis & Schnabel, 1983; Kelley, 1995].

If these assumptions are fulfilled the BP algorithm can be made globally convergent by determining the learning rate in such a way that the error function is exactly subminimized along the direction of the negative of the gradient in each iteration. 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 rate so that the error function is sufficiently decreased on each iteration, accompanied by a significant change in the value of w.

The following conditions, associated with the names of Armijo, Goldstein and Price [Ortega & Rheinboldt, 2000] are used to formulate the above ideas and to define a criterion of acceptance of any weight iterate produced by Eq.(9):

$$E(w^k - \eta_k\, g(w^k)) - E(w^k) \leq -\sigma_1 \eta_k \|g(w^k)\|^2, \tag{10}$$

$$g(w^k - \eta_k\, g(w^k))g(w^k) \geq \sigma_2 \|g(w^k)\|^2, \tag{11}$$

where $0 < \sigma_1 < \sigma_2 < 1$. Thus, we seek to satisfy the conditions (10)–(11) by selecting an appropriate value for the learning rate: the first condition ensures that using η_k the error function is reduced at each iteration and the second condition prevents η_k from becoming too small.

3. Adaptive Learning Rate Algorithms in an Optimization Context

Adaptive learning rate algorithms 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 [Vogl *et al.*, 1988; Battiti, 1989];

(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 [Chan & Fallside, 1987];

(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 [Jacobs, 1988; Minai & Williams, 1990] and Silva and Almeida's method [Silva & Almeida, 1990] follow the last approach. Another method of this group, named *quickprop*, has been presented in [Fahlman, 1989]. Quickprop is based on independent secant steps in the direction of each weight [Vrahatis *et al.*, 2000b]. Riedmiller and Braun in 1993 proposed the *Rprop* algorithm [Riedmiller & Braun, 1993]. 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. This approach accelerates training, mainly, in the flat regions of the error function [Pfister & Rojas, 1993; Rojas, 1996].

Note that all the above mentioned learning rate adaptation methods employ heuristic coefficients in an attempt to secure converge of the BP algorithm to a minimizer of E and to avoid oscillations. A different approach is to exploit the local shape of the error surface as described by the direction cosines. In this case the learning rate is a weighted average of the direction cosines of weight changes at the current and several previous successive iterations [Hsin *et al.*, 1995]. Lastly, the so-called *search-then-converge* schedules combine the desirable features of the standard Least-Mean-Square and traditional stochastic approximation algorithms [Darken *et al.*, 1992] to gradually decrease the learning rate.

Next we examine approaches to dynamically adapt the rate of learning that are based on optimization methods [Yu *et al.*, 1995; Magoulas *et al.*, 1997a, 1997b; Vrahatis *et al.*, 2000a; Anastasiadis *et al.*, 2005a; Anastasiadis *et al.*, 2005b]. In the context of unconstrained optimisation, Armijo's modified SD algorithm automatically adapts the rate of convergence [Armijo, 1966]. In order to incorporate Armijo's search method for the adaptation of the learning rate in the BP algorithm, the following assumptions are needed:

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

(b) For $w^0 \in \mathbb{R}^n$ define $\mathcal{S}(w^0) = \{w : E(w) \leq E(w^0)\}$, then $E \in C^1$ on $\mathcal{S}(w^0)$ and $g(w)$ is Lipschitz continuous on $\mathcal{S}(w^0)$, i.e. there exists a Lipschitz constant $L > 0$, such that

$$\|g(w) - g(v)\| \leq L\|w - v\|, \tag{12}$$

for every pair $w, v \in S(w^0)$.

(c) $r > 0$ implies that $m(r) > 0$, where $m(r) = \inf_{w \in \mathcal{S}_r(w^0)} \|g(w)\|$, $\mathcal{S}_r(w^0) = \mathcal{S}_r \cap \mathcal{S}(w^0)$, $\mathcal{S}_r = \{w : \|w - w^*\| \geq r\}$, and w^* is any point for which $E(w^*) = \inf_{w \in \mathbb{R}^n} E(w)$, (if $\mathcal{S}_r(w^0)$ is void, we define $m(r) = \infty$).

If the above assumptions are fulfilled and $\eta_m = \eta_0/q^{m-1}$, $m = 1, 2, \ldots$, with η_0 an arbitrary initial learning rate, then the sequence (9) can be written as:

$$w^{k+1} = w^k - \eta_{m_k} g(w^k), \quad k = 0, 1, 2, \ldots \tag{13}$$

where m_k is the smallest positive integer for which

$$E(w^k - \eta_{m_k}\, g(w^k)) - E(w^k) \leq -\frac{1}{2}\eta_{m_k}\|g(w^k)\|^2, \tag{14}$$

and it converges to the weight vector w^* which minimizes the function E [Armijo, 1966; Ortega & Rheinboldt, 2000]. Of course, this adaptation method does not guarantee to find the optimal learning rate but only an acceptable one, so that convergence is obtained and oscillations are avoided. This is achieved using the inequality (14) which ensures that the error function is reduced sufficiently with every iteration. This approach is able to handle arbitrary learning rates and in this way learning by neural networks on a first-time basis for a given problem becomes feasible.

3.1. *Adapting a self-determined learning rate*

The work of Cauchy [1847] and Booth [1949] suggests determining the learning rate η_k by a Newton step for the equation $E(w^k - \eta\, d^k) = 0$, for the case that $E : \mathbb{R}^n \to \mathbb{R}$ satisfies $E(w) \geq 0\ \forall\, w \in \mathbb{R}^n$. Thus

$$\eta_k = \frac{E(w^k)}{g(w^k)^\top d^k},$$

where d^k denotes the search direction. When $d^k = g(w^k)$, the iterative scheme (9) is reformulated as:

$$w^{k+1} = w^k - \left[\frac{E(w^k)}{\|g(w^k)\|^2}\right] g(w^k), \quad k = 0, 1, 2, \ldots \tag{15}$$

The iterations (15) constitute a gradient method that has been studied by Altman [1961]. The implementation of this method for minimizing the nonquadratic error function is based on the assumption that E is replaced at the kth stage of the iteration by a quadratic function E^k which approximates E in a neighborhood of the kth iterate w^k. The minimizer of E^k is then taken to be the next iterate w^{k+1}.

Obviously, the iterative scheme (15) takes into consideration information from both the error function and the gradient magnitude. When the gradient magnitude is small it considers the local shape of E is flat, otherwise it is steep. The value of the error function indicates how close to the global minimizer this local shape is. Taking into consideration the above pieces of information, the iterative scheme (15) is able to escape from local minima located far from the global minimizer.

In practical applications, the error function has broad flat regions adjoined with narrow steep ones. This causes the iterative scheme (15) to create very large learning rates, due to the small values of the denominator, pushing the neurons into saturation and thus it exhibits pathological convergence behavior. In order to alleviate this situation and eliminate the possibility of using an unsuitable self-determined learning rate, denoted by η_0, a proper learning rate "tuning" is suggested. Therefore we decide whether the obtained weight vector is acceptable or not by considering if the condition (14) is satisfied or not. Unacceptable vectors are redefined using learning rates defined by the relation

$$\eta_k = \frac{\eta_0}{q^{m_k - 1}}, \quad \text{for} \ \ m_k = 1, 2, \ldots$$

Moreover, this strategy allows to use one-dimensional minimization of the error function without losing global convergence (see [Magoulas *et al.*, 1999] for implementations of this method).

3.2. *Lipschitz constant estimation for learning rate adaptation*

It is well known that the optimal value of the learning rate η in (9) depends on the morphology of the error surface and as Armijo pointed out in [Armijo, 1966] it is related to the value of the Lipschitz constant L. For example, when the error surface has steep regions L is large and a small value for the learning rate is appropriate in order to guarantee the BP convergence. On the other hand when the error surface has flat regions, L is small and a large learning rate is appropriate to accelerate the convergence speed. Despite these simple intuitive rules, the fundamental algorithmic issue is to find the proper learning rate that compensates for the small magnitude of the gradient in the flat regions and damps down the large weight changes in highly curved regions that lead the weight vector to oscillate instead of approaching the minimum.

Based on Armijo's approach the sequence of weight vectors $\{w^k\}_{k=0}^{\infty}$ defined by (9) can be redefined as:

$$w^{k+1} = w^k - 0.5\, L^{-1}\, g(w^k), \quad k = 0, 1, 2, \ldots. \tag{16}$$

It can be shown, in a deterministic framework of analysis, that the weight update equation (16) converges to the point w^* (see [Magoulas *et al.*, 1997a, 1997b]). However, in the neural network implementation, neither the morphology of the error surface nor the value of L are known *a priori*. Thus, a local estimation of the Lipschitz constant can be used. This is defined for the kth iteration as:

$$\Lambda^k = \frac{\|g(w^k) - g(w^{k-1})\|}{\|w^k - w^{k-1}\|}, \tag{17}$$

where w^k and w^{k-1} are a pair of subsequent weight updates. Relation (17) exploits all the local information regarding the direction without any additional function and gradient evaluations. In this way, the learning rate $0.5/\Lambda^k$ is sensitive to the local shape of the error function.

In order to eliminate the possibility of using an unsuitable local estimation of the Lipschitz constant, which can lead the iterative procedure (16) to oscillate temporarily, a proper learning rate "tuning" is needed. To this end, any point of the sequence $\{w^k\}_{k=0}^{\infty}$, that does not satisfy the condition (14), has to be redefined using Armijo's search method. To be more specific, Armijo's search method gradually reduces inappropriate learning rate values to acceptable ones, according to the relation

$$\eta_k = \frac{1}{2^{m_k} \Lambda^k}, \quad \text{for} \ \ m_k = 1, 2, \ldots$$

to satisfy the condition (14). Thus, it provides a damping effect in order to avoid oscillations and keeps the descent direction. This strategy also allows to guarantee the convergence of the algorithm due to Armijo [Magoulas *et al.*, 1997a, 1997b].

4. Approaches to Adapting a Different Learning Rate for Each Weight Through Optimization

As mentioned in the previous section, the SD method requires the assumption that E is twice continuously differentiable and that η is chosen to satisfy the relation $\sup \|H(w)\| \leq \eta^{-1} < \infty$ in the level set $\mathcal{S}(w^0)$ [Cauchy, 1847; Goldstein, 1962, 1965].

The eigensystem of the Hessian matrix H can be used to determine the shape of the error function E in the neighborhood of a local minimizer [Jacobs, 1988; Androulakis *et al.*, 1997]. Thus, studying the sensitivity of the minimizer to small changes by approximating the error function by 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. Thus, a value for the learning rate that yields a large variation along the eigenvector corresponding to the maximum eigenvalue may result in oscillations. On the other hand, a value of the learning rate that yields a small variation along the eigenvector corresponding to the minimum eigenvalue may result in small steps along this direction and thus, in slight reduction of the error function. Thus, in general, a learning rate appropriate for any one weight direction is not necessarily appropriate for the others.

To alleviate this situation, a different adaptive learning rate for each weight coordinate has been suggested by several researchers [Jacobs, 1988; Fahlman, 1989; Minai & Williams, 1990; Silva & Almeida, 1990; Pfister & Rojas, 1993; Riedmiller & Braun, 1993]. This approach allows to find the proper learning rate that compensates the small magnitude of the gradient in a flat direction, in order to avoid slow convergence, and damps down a large weight change in a steep one, in order to avoid oscillations.

In this case, the weight updates are described by the following iterative scheme:

$$w^{k+1} = w^k - \operatorname{diag}\{\eta_1^k, \ldots, \eta_n^k\} g(w^k), \quad k = 0, 1, 2, \ldots. \tag{18}$$

Clearly, in (18) the weight vector 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, defined by the term $-\eta_i \partial_i E(w)$, along the direction of each weight coordinate (where $\partial_i E(w)$ denotes the partial derivative of E with respect to the ith coordinate). Note that training algorithms based on this approach employ heuristic procedures for the determination of the adaptive learning rates and they do not guarantee that the weight updates will converge to a minimizer of E [Jacobs, 1988; Fahlman, 1989; Minai & Williams, 1990; Silva & Almeida, 1990; Pfister & Rojas, 1993; Riedmiller & Braun, 1993].

4.1. *Adaptive learning rate algorithms as nonlinear Jacobi processes*

In the sequel, we show that the class of training algorithms with a different learning rate for each weight is equivalent to the class of *nonlinear Jacobi* methods for the numerical solution of a system of nonlinear equations:

$$F(x) = \Theta^n \equiv (0, 0, \ldots, 0), \tag{19}$$

where $F = (f_1, \ldots, f_n) : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}^n$ is a continuously differentiable mapping on an open neighborhood $\mathcal{D}^* \subset \mathcal{D}$ of a solution $x^* \in \mathcal{D}$ of (19).

This particular class of methods is based on the *nonlinear Jacobi process* [Ortega & Rheinboldt, 2000; Vrahatis *et al.*, 2003] and when applied to the system

$$g(w) \equiv \nabla E(w) = \Theta^n, \tag{20}$$

or equivalently to the following system of equations:

$$\begin{aligned} \partial_1 E(w_1, w_2, \ldots, w_n) &= 0, \\ \partial_2 E(w_1, w_2, \ldots, w_n) &= 0, \\ &\vdots \\ \partial_n E(w_1, w_2, \ldots, w_n) &= 0, \end{aligned} \tag{21}$$

allows to handle the n-dimensional minimization problem of the error function E, at the kth epoch, using reduction to the following one-dimensional nonlinear equations for the components of the gradient $\partial_1 E, \partial_2 E, \ldots, \partial_n E$:

$$\partial_i E(w_1^k, \ldots, w_{i-1}^k, w_i, w_{i+1}^k, \ldots, w_n^k) = 0, \qquad (22)$$

for w_i, $i = 1, \ldots, n$, and then set:

$$w_i^{k+1} = w_i^k + \gamma_k (w_i - w_i^k), \qquad (23)$$

for some relaxation factor γ_k.

Thus, the training of an FNN by applying a back-propagation based algorithm that uses a different adaptive learning rate for each weight coordinate, coincides with the minimization of the error function E in the direction of each coordinate. To be more specific, these training algorithms may be considered as a composition of the nonlinear Jacobi process with any one-dimensional iterative method. The Jacobi method is applied directly to an equation of the form (20) and then, to obtain the $(k+1)$st update, w_i^{k+1} is taken as the result of one-dimensional methods applied to Eq. (22) by considering all but the ith weight as constant.

According to the proposed approach, the well known algorithms of *delta-bar-delta* [Jacobs, 1988], *extented delta-bar-delta* [Minai & Williams, 1990] and *Silva-Almeida* [Silva & Almeida, 1990] belong to this general class of training methods. This is so because they approximate a minimizer of E in the direction of each coordinate using one-step of a descent method, which is an approximation of the solution of Eq. (22). Their approximation of the minimizer of E in each direction depends on the way the learning rate is chosen; a heuristically chosen learning rate for each direction is used and convergence is not guaranteed. Specifically, the learning rate is evaluated by employing heuristic procedures that exploit information regarding the history of the partial derivative of $E(w)$ with respect to the ith weight and/or the history of the corresponding learning rate. Then, the algorithms immediately define the new weight coordinate. Of course, their approximations coincide with those obtained by this general class of methods by taking in Eq. (23) a proper value of γ_k. The heuristics used by these algorithms can also be considered as estimations of the optimal value of γ_k.

4.2. *One-dimensional iterative methods in training algorithms with adaptive learning rates*

As mentioned is the previous subsection, the minimizer approximation, obtained by the subminimization techniques, is relaxed by the factor γ_k. Then, in practice, it is not always useful to spend a lot of computational effort in finding very accurate approximations of the subminimizer in each coordinate direction. Furthermore, the computational effort is increased when the dimension of the problem is very high. Thus, a single iteration of the subminimization technique in each direction is sufficient. This is also, in practice, the case of the heuristic procedures [Jacobs, 1988; Fahlman, 1989; Minai & Williams, 1990; Silva & Almeida, 1990; Pfister & Rojas, 1993; Riedmiller & Braun, 1993] which utilize one iteration of the subminimization process too.

Thus, if one-step of the one-dimensional method is applied to Eq. (22) we are able to obtain various training algorithms with adaptive rates of convergence. In the sequel we propose several iterative schemes which are based on traditional one-dimensional iterative methods [Ralston & Rabinowitz, 1978; Ortega & Rheinboldt, 2000; Gill *et al.*, 1981; Press *et al.*, 1992]. Hence, Eq. (18) is reformulated according to the iterative method used.

(1) The Jacobi–modified bisection method: It is known that the bisection is a global convergence method, it always converges within the given interval, it is worst-case optimal, i.e. it possesses asymptotically the best possible rate of convergence in the worst-case [Sikorski, 2001] and it has a known behavior concerning the number of iterations required, to obtain a root with a predetermined accuracy. This method has been proposed for neural network training in [Magoulas *et al.*, 1996].

In order to compute a root w_i of (22) in the interval $[a_i, b_i]$ we use the following iterative formula [Vrahatis, 1988a, 1988b]:

$$w_i^{p+1} = w_i^p + \operatorname{sgn} \partial_i E(w^0) \operatorname{sgn} \partial_i E(w^p) \frac{h_i}{2^{p+1}},$$
$$p = 0, 1, 2, \ldots, \qquad (24)$$

with $w_i^0 = a_i$, $h_i = b_i - a_i$ and where sgn defines the well known triple valued sign function. Thus the method systematically reduces the specified interval by derivative comparison. Of course, the iterations (24) converge to a root $w_i \in (a_i, b_i)$ if for some w_i^p, $p = 1, 2, \ldots$, the following holds:

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

The number of iterations p of the sequence (24) which are required to obtain an approximate root

w_i^*, such that $|w_i - w_i^*| \leq \delta$ for some $\delta \in (0,1)$, is given by $p = \lceil \log_2(h_i \delta^{-1}) \rceil$, where the notation $\lceil\cdot\rceil$ is the ceiling function of the real number quoted.

To ensure that the solution point is a minimizer along the ith coordinate direction we choose the endpoints a_i and b_i in such a way that, at the left endpoint a_i the ith component of the gradient vector has negative value, or, at the right endpoint b_i the ith component of the gradient vector has positive value. In order that this condition is fulfilled we choose these endpoints by the following relation:

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

Thus the weight update equation can be defined as:

$$w_i^{k+1} = w_i^k + \gamma_k (w_i - w_i^k), \tag{25}$$

where w_i is the result of (24).

Variants of this method have been recently proposed in [Anastasiadis *et al.*, 2003] and [Anastasiadis *et al.*, 2004] and tested in several neural network benchmarks outperforming other well known adaptive training algorithms. Other training schemes that proceed solely with the information of the sign of the gradient components have been proposed in [Pfister & Rojas, 1993; Riedmiller & Braun, 1993]. These schemes employ heuristic parameters in the form of learning rate lower and upper bounds to define the interval of search.

(2) The one-step Jacobi–Newton method: The Newton's method is considered extremely powerful, since it converges quadratically. In our case one-step Jacobi–Newton iteration can be defined by:

$$w_i^{k+1} = w_i^k - \gamma_k \frac{\partial_i E(w^k)}{\partial_{ii}^2 E(w^k)}. \tag{26}$$

In [Magoulas & Vrahatis, 2000] a variant of this method was tested in real–world applications exhibiting good performance. A relative training method of this type, which applies (26) to obtain learning rates, can also be found in [Yu *et al.*, 1995].

(3) The one-step Jacobi-secant method: This method replaces the second derivative needed in Newton's method by its finite difference approximation. Thus, the weight update can be defined as:

$$w_i^{k+1} = w_i^k - \gamma_k \left\{ \frac{\partial_i E(w^k) - \partial_i E(w^{k-1})}{w_i^k - w_i^{k-1}} \right\}^{-1} \partial_i E(w^k), \tag{27}$$

which looks like the basic formula of the *quickprop* method [Fahlman, 1989]. One difficulty with the secant method is that the iterates may diverge when $\partial_i E(w^k)$ and $\partial_i E(w^{k-1})$ have the same sign [Gill *et al.*, 1981]. To overcome this difficulty a modification of the secant method is suggested, named *regula falsi*, in which a "better" approximation of the solution, i.e. the latest point w_i^{k+1}, replaces either w_i^k or w_i^{k-1} depending on which corresponding partial derivative value agrees in sign with $\partial_i E(w^{k+1})$ [Gill *et al.*, 1981]. Training algorithms that employ this method have been introduced in [Magoulas & Vrahatis, 2000; Vrahatis *et al.*, 2000b; Vrahatis *et al.*, 2003] for neural network training.

(4) The one-step Jacobi–Steffensen method: This method is of practical interest because, under suitable conditions, it exhibits the same quadratic convergence as Newton's method while it does not require second derivatives [Ortega & Rheinboldt, 2000]. In this case the weight update equation takes the following form:

$$w_i^{k+1} = w_i^k - \gamma_k \left\{ \frac{\partial_i E(w^k) - \partial_i E\left[w^k - \partial_i E(w^k)\, e_i\right]}{\partial_i E(w^k)} \right\}^{-1} \partial_i E(w^k), \tag{28}$$

where e_i denotes the ith column of the identity matrix.

(5) The Jacobi–Brent method: This method combines bisection and inverse quadratic interpolation [Press *et al.*, 1992] to converge from the neighborhood of a root which is defined as an interval whose endpoints have opposite sign values. Of course, provision should be taken for the case of a root outside the interval. In [Brent, 1973] several techniques are suggested to avoid this situation. Thus, in our case, if three previous points are available, then the weight update is given by the following formula:

$$\begin{aligned} w_i^{k+1} = w_i^k + \gamma_k \Bigg\{ & \frac{\partial_i E(w^{k-2}) \partial_i E(w^{k-1})}{[\partial_i E(w^k) - \partial_i E(w^{k-2})]\,[\partial_i E(w^k) - \partial_i E(w^{k-1})]}\, w_i^k \\ & + \frac{\partial_i E(w^{k-1}) w_i^{k-2}}{[\partial_i E(w^{k-2}) - \partial_i E(w^{k-1})]\,[\partial_i E(w^{k-2}) - \partial_i E(w^k)]}\, \partial_i E(w^k) \\ & + \frac{\partial_i E(w^{k-2}) w_i^{k-1}}{[\partial_i E(w^{k-1}) - \partial_i E(w^k)]\,[\partial_i E(w^{k-1}) - \partial_i E(w^{k-2})]}\, \partial_i E(w^k) - w_i^k \Bigg\}. \end{aligned} \tag{29}$$

(6) The Jacobi–Illinois and the Jacobi–Pegasus methods: These methods are modifications of the secant method. They differ from the regula falsi method in the way they discard previous approximations of the solution. The point which is the most recent of the prior approximations of the solution is retained. These methods modify the length of the step by a factor α which is set equal to 0.5 for the Illinois algorithm and $\partial_i E(w^{k-1})/(\partial_i E(w^{k-1}) + \partial_i E(w^k))$ for the Pegasus algorithm, [Ralston & Rabinowitz, 1978]. The weight update equation has the following general form:

$$\begin{aligned} w_i^{k+1} = w_i^k + \gamma_k \Bigg[& \frac{\alpha\, \partial_i E(w^{k-2})\, w_i^k}{[\alpha\, \partial_i E(w^{k-2}) - \partial_i E(w^k)]} \\ & - \frac{w_i^{k-2}\, \partial_i E(w^k)}{[\alpha\, \partial_i E(w^{k-2}) - \partial_i E(w^k)]} - w_i^k \Bigg]. \end{aligned} \tag{30}$$

Relative training algorithms can be easily constructed by utilizing any one-dimensional iterative method. To facilitate the reader we recall two concepts which will be used in the following convergence theorem.

(1) The Property A^π: Young [1954] discovered a class of matrices that can be partitioned into block–tridiagonal form, possibly after a suitable permutation and they are described as having *property A*.

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 induces also 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 4.1 [Axelsson, 1996]. 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 equations (25)–(30), denoted in general by $\mathcal{P}$, converge to w^*. A measure of the rate of their convergence is obtained by taking appropriate roots of successive errors. To this end we use the following definition.

Definition 4.2 [Ortega & Rheinboldt, 2000]. Let $\{w^k\}_{k=0}^{\infty}$ be any sequence that converges to w^*. Then the number

$$R\{w^k\} = \lim_{k \to \infty} \sup \|w^k - w^*\|^{1/k}, \tag{31}$$

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

$$R(\mathcal{P}, w^*) = \sup\{R\{w^k\}; \{w^k\} \in C(\mathcal{P}, w^*)\}, \tag{32}$$

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

For completeness, we present below a convergence result, originally introduced in [Magoulas & Vrahatis, 2000].

Theorem 4.1. *Let E: $\mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable on 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 centered at w^* with radius r, $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$ such that the sequence $\{w^k\}_{k=0}^{\infty}$ generated by the nonlinear Jacobi (22)–(23) iterative procedure $\mathcal{P}$ converges to w^* which minimizes E and $R(\mathcal{P}, w^*) = \varrho$.*

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 [Ortega & Rheinboldt, 2000]). Finding such a point is equivalent to obtaining the corresponding solution $w^* \in \mathcal{D}$ of Eq. (20) or equivalently to solving the system of Eqs. (21) by applying the nonlinear Jacobi iterations (22)–(23) to this system employing any one-dimensional iterative method [Ralston & Rabinowitz, 1978; Ortega & Rheinboldt, 2000; Gill *et al.*, 1981; Press *et al.*, 1992].

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

$$H(w^*) = D(w^*) - L(w^*) - L^\top(w^*).$$

Since, $H(w^*)$ is symmetric and positive definite, then $D(w^*)$ is positive definite [Varga, 2000]. Moreover, since $H(w^*)$ has the property (A^π), the eigenvalues of the matrix:

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

are real and for the spectral radius $\rho(\Phi(w^*))$ of $\Phi(w^*)$ holds that [Axelsson, 1996]:

$$\rho(\Phi(w^*)) = \varrho < 1,$$

then there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$, such that, for any $w^0 \in \mathcal{S}$, there is a unique sequence $\{w^k\}_{k=0}^{\infty} \subset \mathcal{S}$ which satisfies (22)–(23) such that $\lim_{k\to\infty} w^k = w^*$ and $R(\mathcal{P}, w^*) = \varrho$ [Ortega & Rheinboldt, 2000; Voigt, 1971; Reinboldt, 1974]. Thus the Theorem is proved. ■

Remark 4.1. By the proof of Theorem 4.1, it is shown that the asymptotic rate of convergence of the composite algorithms is independent of the precise rate of the one-dimensional method. This fact has two interesting consequences. First, the asymptotic rate of convergence is not enhanced if one takes more than one step of the one-dimensional method. Second, the extra computational cost required when higher order one-dimensional methods are used to form training algorithms with adaptive learning rate for each weight does not speed up the learning process. Thus, in this context a method that approximates the second derivative, such as the quickprop method, would be superior to any other higher order method that requires second derivatives.

4.3. *Estimating the Lipschitz constant along each weight direction*

Below, we derive a method that exploits the local information regarding the direction and the morphology of the error surface at the current point in the weight space in order to dynamically adapt a different learning rate for each weight. This learning rate adaptation is based on estimation of the local Lipschitz constant along each weight direction [Magoulas *et al.*, 1999].

As mentioned in the previous section, the inverse of the Lipschitz constant L can be used to obtain the optimal learning rate which is $0.5\,L^{-1}$ [Armijo, 1966]. The value L in Rel. (12) can be estimated by the *modulus of continuity of g on $\mathcal{S}(w^0)$* [Ortega & Rheinboldt, 2000]:

$$\mu(g, \delta) = \sup\{\|g(w) - g(v)\|,\ \text{for } w, v \in \mathcal{S}(w^0),\ \text{and } \|w - v\| \le \delta\}.$$

Note that if g is Lipschitz for some real number L and for all w, $v \in \mathcal{S}(w^0)$ then we have immediately that:

$$\mu(g, \delta) \le L\,\delta.$$

Furthermore, by taking the infinity norm (Chebychev distance) the Lipschitz constant L using Rel. (12) can be obtained by the following relation:

$$L = \max_{w \ne v} \frac{\|g(w) - g(v)\|_\infty}{\|w - v\|_\infty}.$$

Thus, in order to obtain a local estimation Λ^k of L we use the following relation:

$$\Lambda^k = \frac{\max\limits_{1\le j\le n} |\partial_j E(w^k) - \partial_j E(w^{k-1})|}{\max\limits_{1\le j\le n} |w_j^k - w_j^{k-1}|}, \tag{33}$$

where w^k and w^{k-1} is a pair of consecutive weight updates at the kth iteration.

Now, in order to take into consideration the shape of the error surface to dynamically adapt a different learning rate for each weight, we estimate

Λ^k along the ith direction, $i = 1, \ldots, n$, at the kth iteration by:

$$\Lambda_i^k = \frac{|\partial_i E(w^k) - \partial_i E(w^{k-1})|}{|w_i^k - w_i^{k-1}|}, \tag{34}$$

and we use the inverse of Λ_i^k to estimate the learning rate of the ith coordinate direction.

The reason for choosing Λ_i^k instead of Λ^k is that when large changes of the ith weight occur and the error surface along the ith direction is flat, we have to take a larger learning rate along this direction. This can be done by taking (34) instead of (33), since in this case (34) underestimates Λ^k. On the other hand, when small changes of the ith weight occur and the error surface along the ith direction is steep, (34) overestimates Λ^k and thus the learning rate to this direction is dynamically reduced in order to avoid oscillations. Therefore, the larger the value of Λ_i^k is, the smaller learning rate is used and vice versa. As a consequence, the iterative scheme (9) is reformulated as:

$$w^{k+1} = w^k - \gamma_k \text{diag}\left\{\frac{1}{\Lambda_1^k}, \ldots, \frac{1}{\Lambda_n^k}\right\} g(w^k), \quad k = 0, 1, 2, \ldots \tag{35}$$

where γ_k is a relaxation coefficient. By properly tuning γ_k, we are able to avoid temporary oscillations and/or to enhance the rate of convergence when we are far from a minimum.

A search technique for γ_k consists of finding the weight vectors of the sequence $\{w^k\}_{k=0}^{\infty}$, that satisfy the following condition:

$$E(w^{k+1}) - E(w^k) \leq \frac{1}{2}\gamma_{m_k}\left\langle -\text{diag}\left\{\frac{1}{\Lambda_1^k}, \ldots, \frac{1}{\Lambda_n^k}\right\} g(w^k), g(w^k)\right\rangle. \tag{36}$$

If a weight vector w^{k+1} does not satisfy the above condition, it has to be evaluated again using the Armijo–Goldstein–Price conditions. In this case, the search method gradually reduces inappropriate γ_k values to acceptable ones by finding the smallest positive integer $m_k = 1, 2, \ldots$ such that $\gamma_{m_k} = \gamma_0 / q^{m_k - 1}$ satisfies the condition (36).

A common characteristic of all the methods that adapt a different learning rate for each weight coordinate is that they require the global information obtained by taking into consideration all the coordinates at each iteration. To this end, learning rate lower and upper bounds are usually used [Pfister & Rojas, 1993; Riedmiller & Braun, 1993] to avoid the usage of an extremely small or large learning rate component, which misguides the resultant search direction. The learning rate lower bound (η_{lb}) is related to the desired accuracy in obtaining the final weights and helps to avoid unsatisfactory convergence rate. The learning rate upper bound (η_{ub}) helps limiting the influence of a large learning rate component on the resultant descent direction and depends on the shape of the error function; in the case η_{ub} is exceeded for a particular weight, its learning rate in the kth iteration is set equal to the previous one of the same direction. It is worth noticing that the values of both η_{lb} and η_{ub} do not affect the stability of the algorithm.

5. Imprecision Incurred in Supervised Training

Training algorithms described so far require precise error function and gradient values. However, the fact that neural networks are simulated on computers with finite accuracy bounds implies that it may be difficult or impossible to obtain very precise values for the error function and the gradient [Wray & Green, 1995]. Thus, software-based FNNs while exploiting the advantage of training by examples, are directly affected by numerical imprecision; a common problem encountered in numerical simulations. The arithmetic operations required in the numerical simulations of the BP affect the accuracy of the result. All of these operations can be severely impacted by imprecision, especially for problems that are ill-conditioned even when a high precision is used [Holt & Hwang, 1993]. Moreover, using minimization methods for training FNNs derivative calculations as well as one-dimensional subminimization (for the case of nonlinear conjugate gradient methods) and approximations of the inverse Hessian (for the case of quasi-Newton and variable metric methods) are required. A detailed analysis on the sources of imprecision involved to this kind of computations is presented in [Gill *et al.*, 1981; Battiti, 1992].

A crucial factor of imprecision, as pointed in [Wray & Green, 1995], is the evaluation of the activation function $f(\text{net}_j)$. This function can be calculated using a polynomial approximation which implies that numerical accuracy constraints are introduced in the calculation of the error value.

In the special case of FNN applications with a very large number of patterns, the errors involved because of the imprecision in the computation of the gradient of the batch error measure may be comparable to the gradient itself. In general, the rounding off error is one serious source of imprecision in the error function value E and its gradient. When this type of error is generated by overflow, mostly occurred during the calculation of the term net_j^l, is not crucial due to the characteristics of the sigmoid neuron model. However the underflow error that occurs during the calculation of the backpropagating error signal [Rumelhart *et al.*, 1986] denoted by δ_j is very crucial and can lead to nonconvergence and saturation [Rigler *et al.*, 1991]. Despite the fact that the error associated with neuron j can be significant, δ_j may become negligible and rounded to zero if the derivative of the activation function $f'(\text{net}_j^l)$ is very small. In this case weight adaptation is not possible although there is a large value of error.

A similar to saturation phenomenon occurs when second derivative based training methods are used. In this case, problematic situation occurs when the Hessian is not positive definite, as well as when it is ill-conditioned or singular (see [Magoulas *et al.*, 1997a, 1997b; Battiti, 1992] for simulations on these kind of problems).

Moreover, in various small and large scale FNN applications the error surface has flat regions. This results in the evaluation of imprecise gradient values which affects all training methods that use first derivatives in case we are far from the minimum.

Imprecision is also encountered when the partial derivative of E with respect to the ith weight is evaluated using the forward-difference approximation:

$$\partial_i E(w) \approx \frac{E(w + \text{pert} \cdot e_i) - E(w)}{\text{pert}}, \tag{37}$$

where pert is a small quantity proportional to the square root of the relative machine precision and e_i denotes the ith column of the identity matrix [Dennis & Schnabel, 1983]. This approach has been used by several researchers as an alternative to the generation of derivatives using the backpropagation chain rule [Rumelhart *et al.*, 1986] because only forward operations of the FNN can give the weight updates. As reported in [Gill *et al.*, 1981], truncation error as a consequence of the neglected terms in the Taylor series, condition or cancellation error due to imprecise values of E and rounding off errors are introduced in this case.

The above mentioned problematic situations can be handled, at least in part, by developing training algorithms that can take into consideration that the error function and gradient values are known only imprecisely. In the next subsection, a training method eminently suitable to work under imprecise conditions is presented. If a method is capable of converging when imprecise values are used then computational effort can be saved by avoiding the extra work required to compute precise function and gradient values.

5.1. *The training with imprecision algorithm*

The algorithm is derived from a recently proposed method for unconstrained optimization [Vrahatis *et al.*, 1996]. This optimization method has an improved performance when compared with nonlinear conjugate gradient methods and BFGS on many well known test cases (see [Vrahatis *et al.*, 1996] for comparative results).

In order to find a proper weight vector w^* so that the FNN exhibits a desired behavior we want a sequence of weight vectors $\{w^k\}_0^\infty$ that converges to a minimizer of the error function E. An element of this sequence of weight vectors can be obtained by solving an one-dimensional equation for each component of the weight vector as the following one:

$$\begin{aligned} &E(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w, w_{i+1}^k, \ldots, w_n^k) \\ &\quad - E(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i^k, w_{i+1}^k, \ldots, w_n^k) = 0. \end{aligned} \tag{38}$$

Solving the above equation for w and keeping all the other components of the weight vector in their constant values we get $\hat{w}_n$ as a solution. Then, the weight vector $(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, \hat{w}_i, w_{i+1}^k, \ldots, w_n^k)$ possesses the same error function value with the vector $(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i^k, w_{i+1}^k, \ldots, w_n^k)$, i.e. these points belong to the same contour line of the function E.

Assuming that the Hessian of the error function E at w^* is positive definite, which means that E curves up from w^* in all directions, any point which belongs to the line that connects the points $(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, \hat{w}_i, w_{i+1}^k, \ldots, w_n^k)$ and $(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i^k, w_{i+1}^k, \ldots, w_n^k)$ possesses smaller error value than these points. To find such a point we update at the $(k+1)$st iteration the ith component w_i^k using the following relation:

$$w_i^{k+1} = w_i^k + \zeta(\hat{w}_i - w_i^k), \quad \zeta \in (0, 1). \tag{39}$$

Obviously the above procedure handles n-dimensional problems using reduction to simpler one-dimensional equations like Eq. (38). Of course, any one of the well known one-dimensional rootfinding methods [Ralston & Rabinowitz, 1978; Ortega & Rheinboldt, 2000; Gill *et al.*, 1981; Press *et al.*, 1992] can be employed to solve Eq. (38). Here we use the bisection method because it can be applied to imprecise problems. Specifically, in order to compute a root $\hat{w}_i$ of Eq. (38) in the interval (a_i, b_i), we use the simplified version of the bisection method [Vrahatis, 1988a, 1988b] which can be modified to the following one:

$$\hat{w}_i^{p+1} = \hat{w}_i^p + \frac{c\,\mathrm{sgn}(E(z^p) - E(z))}{2^{p+1}}, \tag{40}$$

where $p = 0, 1, \ldots$ indicates iterations, sgn defines the well known sign function, $z^p = (w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i^p, w_{i+1}^k, \ldots, w_n^k)$, $z = (w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i^k, w_{i+1}^k, \ldots, w_n^k)$, $c = \mathrm{sgn}(E(z^0) - E(z))h_i$ with $h_i = b_i - a_i$, and $z^0 = (w_1^{k+1}, \ldots, w_{i-1}^{k+1}, a_i, w_{i+1}^k, \ldots, w_n^k)$.

To minimize the function E by applying the above sequence we choose the endpoints a_i and b_i in such a way that, at the left endpoint a_i the ith component of the gradient has negative value, or, at the right endpoint b_i the ith component of the gradient has positive value. In order that this condition is fulfilled, we choose these endpoints by the following relation:

$$a_i = w_i^k - \frac{1}{2}\{1 + \mathrm{sgn}\,\partial_i E(z)\}h_i, \quad b_i = a_i + h_i. \tag{41}$$

The number of iterations p of the sequence (40) which are required to obtain an approximate root $\hat{w}_i$ such that $|\hat{w}_i - w_i^*| \le \delta$, for some $\delta \in (0, 1)$, is given by:

$$p = \lceil \log_2(h_i \delta^{-1}) \rceil. \tag{42}$$

It is evident from the sequence (40) that the only computable information required by this method is related to algebraic signs and thus it can be proceeded solely by comparing the relative size of the function values. Also, the gradient values in Rel. (41) can be calculated by using Rel. (37). In this case the sign can also be accurately obtained by comparing the relative size of the function value. So, rounding and quantization errors, causing, in simulations, imprecise function values, cannot affect its convergence as long as the signs are preserved. In addition, storage requirements regarding gradient are minimized. Furthermore, Sequence (40) is a global convergence method, it always converges within the given interval, it is worst-case optimal, i.e. it possesses asymptotically the best possible rate of convergence in the worst-case [Sikorski, 2001] and it has a known behavior concerning the number of iterations required to obtain a root with a predetermined accuracy (see Rel. (42)).

A high level description of the batch version of this algorithm is outlined below.

Initialization: Randomly initialize the weights; define stepsizes h_i in each weight direction; the relaxation coefficients ζ, γ; the maximum number of epochs (*ME*); the predetermined desired accuracies δ and ε.

Recursion: For $k = 1, \ldots, ME$ (epochs); and for $i = 1, 2, \ldots, n$ (components of the weight vector w).

(1) Present all patterns to the network and compute the output of all nodes.
  (a) Calculate E using Eq. (7).
  (b) Define the interval (a_i, b_i) where the bisection seeks the root that minimizes E using Eq. (41).
  (c) Find the parameter $\hat{w}$ that satisfies Eq. (38) by applying Eq. (40) in (a_i, b_i) within accuracy δ.
  (d) If $\hat{w} \ge b_i - \delta$, set $y^0 = (w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i^k, w_{i+1}^k, \ldots, w_n^k)$ and go to Step (5); otherwise continue.
  (e) Set $w_i^{k+1} = w_i^k + \zeta(\hat{w} - w_i^k)$.
  (f) If $i < n$, increment i and begin recursion.

(2) Check the convergence criterion $\|w^{k+1} - w^k\| \le \varepsilon$. In case it is true, the weights remain unmodified so *Terminate*; otherwise go to next step.
(3) Set $w^{k+1} = w^k + \gamma\,(w^{k+1} - w^k)$.
(4) If $\mathrm{sgn}\left(E(w^{k+1}) - E(w^k)\right) \le 0$, increment k and reset i; otherwise set $y^0 = w^k$ and continue.
(5) Apply using Rel. (37) the scheme (13)–(14) by utilizing the starting value y^0 and take its output value y^{AR}.
(6) Set $w^{k+1} = y^{\mathrm{AR}}$, increment k and reset i.

Termination: Get the final weights and the corresponding error value.

Remark 5.1. When the values of E or gradient values can be accurately obtained we may also use the following convergence criteria at Step (2) of our algorithm:

$$\left|\frac{E(w^{k+1}) - E(w^k)}{E(w^k)}\right| \le \varepsilon, \quad \text{or} \quad \|g(w^{k+1})\| \le \varepsilon.$$

Step (2) of the above algorithm detects if the bisection does not converge. This can happen only when

Eq. (40) converges to the right endpoint b_i or when Bolzano's criterion is not fulfilled. Furthermore, Steps (5) and (6) apply the Armijo's condition.

It is evident that the above procedure can be considered as an *nonlinear Successive Over-Relaxation* (nonlinear SOR) method [Ortega & Rheinboldt, 2000] since it allows to handle the n-dimensional minimization problem of the function E, at the kth epoch, by solving the following one-dimensional nonlinear equations:

$$\partial_i E(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i, w_{i+1}^k, \ldots, w_n^k) = 0, \quad (43)$$

for w_i, $i = 1, \ldots, n$, and then set:

$$w_i^{k+1} = w_i^k + \gamma_k (w_i - w_i^k), \quad (44)$$

for some relaxation factor γ_k. It is obvious that any one of the well known one-dimensional rootfinding methods [Ralston & Rabinowitz, 1978; Ortega & Rheinboldt, 2000; Gill *et al.*, 1981; Press *et al.*, 1992] can be employed to solve Eq. (43).

For all these methods including the proposed one we present below a convergence result, originally introduced in [Vrahatis *et al.*, 1996].

Theorem 5.1. *Let $E : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable on 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. Then there exists an open ball centered at w^* with radius r, $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$ such that the sequence $\{w^k\}_{k=0}^{\infty}$ generated by the nonlinear SOR (43)-(44) iterative procedure $\mathcal{P}$ converges to w^* which minimizes E and $R(\mathcal{P}, w^*) = \varrho$.*

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 [Ortega & Rheinboldt, 2000]). Finding such a point is equivalent to obtaining the corresponding solution $w^* \in \mathcal{D}$ of Eq. (20) or equivalently to solving the system of Eqs. (21) by applying the nonlinear SOR (43)–(44) iterative procedure to this system employing any one–dimensional iterative method.

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

$$H(w^*) = D(w^*) - L(w^*) - L^\top(w^*).$$

Since $H(w^*)$ is symmetric and positive definite then, $D(w^*)$ is nonsingular [Varga, 2000]. Suppose that:

$$\Phi_\gamma(w^*) = [D(w^*) - \gamma L(w^*)]^{-1} \times [(1-\gamma) D(w^*) + \gamma L^\top(w^*)],$$

for $\gamma \in (0, 2)$. Now, by virtue of Ostrowski Theorem [Varga, 2000] holds that:

$$\rho(\Phi_\gamma(w^*)) = \varrho < 1,$$

for any $\gamma \in (0, 2)$ and therefore, by the nonlinear SOR theorem [Ortega & Rheinboldt, 2000], there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$, such that, for any $w^0 \in \mathcal{S}$, there is a unique sequence $\{w^k\} \subset \mathcal{S}$ that satisfies the nonlinear SOR prescription such that $\lim_{k\to\infty} w^k = w^*$ and $R(\mathcal{P}, w^*) = \varrho$. Thus the Theorem is proved. ■

6. Complexity Considerations in Artificial Neural Network Training

The effectiveness of an ANN-based approach for solving a particular problem can be examined from a computational complexity point of view [Abu-Mostafa, 1986]. For example, in a neural network solution the number of computations is a measure of time complexity, the number of neurons in an ANN is a measure of space complexity (memory requirements), and the number of weights and biases in the ANN is a measure of Kolmogorov complexity [Hassoun, 1995]. Hence, the goal is to minimize simultaneously the time, space and Kolmogorov complexities of the network.

With regards to the number of neurons in an ANN, the *universal approximation theorem*, proved in [White, 1990], states the following for the general problem of function approximation:

Theorem 6.1. *Standard Feedforward Neural Networks with only a single hidden layer can approximate any continuous function uniformly on any compact set and any measurable function to any desired degree of accuracy.*

A direct implication of the above theorem is that any lack of success in applications must arise from inadequate learning and/or an insufficient number of hidden units and/or the lack of a deterministic relationship between the input patterns and the desired response (target).

The selection of the optimal network architecture for a specific task remains up to date an open problem. An upper bound on the architecture of an FNN designed to approximate a continuous function defined on the unit cube in $\mathbb{R}^n$ is given by the following Theorem [Pinkus, 1999]:

Theorem 6.2. *On the unit cube in* $\mathbb{R}^n$ *any continuous function can be uniformly approximated, to within any error by using a two hidden layer network having* $2n+1$ *units in the first layer and* $4n+3$ *units in the second layer.*

Although single linear threshold units have very limited computational abilities, a network of linear threshold units with one hidden layer can represent any Boolean function as the following Theorem [Anthony, 2003] states.

Theorem 6.3. *There is a Feedforward linear threshold Neural Network with one hidden layer capable of computing any Boolean function.*

A universal network for Boolean functions on $\{0,1\}^n$ is a linear threshold network which is capable of computing every Boolean function of n variables [Anthony, 2003]. In particular [Anthony, 2003] has shown that a one hidden layer linear threshold network with n inputs, 2^n units in the hidden layer, and one output unit, is universal. A question naturally arises as to whether there is a universal network with fewer threshold units. By an easy counting argument, one can obtain a lower bound on the size of any universal network, regardless of its structure. In particular (see [Nechiporuk, 1964; Siu *et al.*, 1995]), any universal network (regardless of how many layers it has) must have at least $\Omega(2^{n/2}/\sqrt{n})$ threshold units. Moreover, any one hidden layer universal network for Boolean functions must have at least $\Omega(2^n/n^2)$ threshold units.

In practice, convergence success, cost of calculations for each step of the training algorithm, total number of iterations required to obtain an acceptable solution are all influencing the complexity of obtaining a neural network–based solution. Various Levenberg–Marquardt, quasi–Newton and trust–region algorithms have been proposed for small to medium size neural nets with sigmoid transfer functions [Hagan & Menhaj, 1994; Kollias & Anastassiou, 1989] in order to reduce the time for training. Variants of these methods, such as the limited-memory quasi-Newton and double dogleg, have been also proposed in an attempt to reduce the memory requirements of these methods [Ampazis & Perantonis, 2002; Bertsekas, 1995]. Nevertheless, first-order methods, such as variants of gradient descent discussed in this paper and conjugate-gradient algorithms [Møller, 1993] appear to be more efficient in training large size neural nets.

Another important consideration for the convergence success of an iterative scheme is its susceptibility to ill-conditioning: the minimization of the network's learning error is often ill-conditioned, especially when there are many hidden neurons [Saarinen *et al.*, 1993]. Second-order methods are considered better for handling ill-conditioned problems [Battiti, 1992; Magoulas *et al.* 1997a, 1997b]; nevertheless, it is not certain that the computational cost for each step and the memory requirements of these methods provide significant advantage when the algorithm starts from a remote starting point (training starts with random weight values) [Battiti, 1992], particularly when the networks use a large number of weights [Magoulas *et al.*, 2002; Plagianakos *et al.*, 2002].

An additional factor that impacts on the complexity of the training process is the existence of a multitude of suboptimal solutions [Gori & Tesi, 1992] which affect the convergence success of the training algorithms. Convergence to a local minimum prevents the ANN from learning the entire training set and results in inferior network performance, or possibly in premature convergence. Intuitively, the existence of local minima is due to the fact that the error function is the superposition of nonlinear activation functions that may have minima at different points, which sometimes results in a nonconvex error function [Gori & Tesi, 1992]. The insufficient number of hidden neurons, as well as improper initial weight values can cause convergence to a local minimum. Several researchers have presented conditions on the network architecture, the training set and the initial weight vector that allow BP-like procedures to reach the optimal solution [Gori & Tesi, 1992; Yu & Chen, 1995]. However, conditions such as the linear separability of the patterns and the pyramidal structure of the ANN [Gori & Tesi, 1992] as well as the need for a great number of hidden neurons (as many neurons as patterns to learn) make these interesting results not easily interpretable in practical situations even for simple problems. Thus, in practice the problem of local minima is treated with a variety of techniques, such as generating a new starting point and starting training all over again after convergence to a local minimum [Magdon-Ismail & Atiya, 2000]; training with noise [Burton & Mpitsos, 1992; Rögnvaldsson, 1994; Anastasiadis & Magoulas, 2004]; variants of simulated annealing [Treadgold & Gedeon, 1998].

7. Conclusions

Backpropagation provides an efficient method to calculate how much changing the weight of a connection would influence the difference between the actual behavior of an artificial neural network in response to a particular input and the desired bahavior as defined by a teacher. This approach is widely known as supervised training and although it is not a plausible model of how learning takes place in the human brain it has demonstrated impressive success in adjusting the weights to optimize an objective function that represents performance.

This paper has focused on adaptive backpropagation algorithms which use either a common adaptive learning rate for all weights or an individual adaptive learning rate for each weight coordinate. A number of techniques for adaptation of learning rate was reviewed, and it has been shown that the training algorithms with adaptive learning rate for each weight can be analyzed as nonlinear Jacobi methods applied to the gradient of the error function. This approach has helped us to synthesize various adaptive algorithms that build on the theory of nonlinear optimization. The way simulations of neural networks are affected by the limited precision was discussed and an algorithm based on the nonlinear successive overrelaxation proces was presented. This approach is eminently useful when training is affected by technology imperfections and environmental changes that cause unpredictable deviations of parameter values from the designed configuration. Lastly, various factors that affect the complexity of neural networks training have been discussed.

Acknowledgments

George D. Magoulas gratefully acknowledges support by the UK Engineering and Physical Sciences Research Council (GR/R92554/01-02). Michael N. Vrahatis gratefully acknowledges the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the Program PYTHAGORAS.

References

Abu-Mostafa, Y. S. [1986] "Complexity of random problems," *Complexity in Information Theory*, ed. Abu-Mostafa, Y. S. (Springer-Verlag, Berlin), pp. 115–131.

Altman, M. [1961] "Connection between gradient methods and Newton's method for functionals," *Bull. Acad. Polon. Sci. Ser. Sci. Math. Astron. Phys.* **9**, 877–880.

Ampazis, N. & Perantonis, S. J. [2002] "Two highly efficient second order algorithms for training feedforward networks," *IEEE Trans. Neural Networks* **13**, 1064–1074.

Anastasiadis, A. D., Magoulas, G. D. & Vrahatis M. N. [2003] "An efficient improvement of the Rprop algorithm," *Artificial Neural Networks in Pattern Recognition, Proc. 1st Int. Association of Pattern Recognition-TC3 Workshop*, eds. Gori, M. & Marinai, S. (Stampa Digitale, Firenze), pp. 197–201.

Anastasiadis, A. D. & Magoulas, G. D. [2004] "Nonextensive statistical mechanics for hybrid learning of neural networks," *Physica A* **344**, 372–382.

Anastasiadis, A. D., Magoulas, G. D. & Vrahatis, M. N. [2004] "A globally convergent Jacobi-bisection method for neural network training," *Proc. Int. Conf. Computational Methods in Sciences and Engineering 2004*, Lecture Series on Computer and Computational Sciences, Vol. 1 (VSP International Science Publishers, Zeist, The Netherlands), pp. 843–848.

Anastasiadis, A. D., Magoulas, G. D. & Vrahatis, M. N. [2005a] "New globally convergent training scheme based on the resilient propagation algorithm," *Neurocomputing* **64**, 253–270.

Anastasiadis, A. D., Magoulas, G. D. & Vrahatis, M. N. [2005b] "Sign-based learning schemes for pattern classification," *Patt. Recogn. Lett.*, in press.

Androulakis, G. S., Magoulas, G. D. & Vrahatis, M. N. [1997] "Geometry of learning: Visualizing the performance of neural network supervised training methods," *Nonlin. Anal. Th. Meth. Appl.* **30**, 4539–4544.

Anthony, M. [2003] "Boolean functions and artificial neural networks," *CDAM Research Report LSE-CDAM-2003-01*, Department of Mathematics and Centre for Discrete and Applicable Mathematics, The London School of Economics and Political Science, London, UK.

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

Axelsson, O. [1996] *Iterative Solution Methods* (Cambridge University Press, NY).

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

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

Becker, S. & Le Cun, Y. [1988] "Improving the convergence of the back–propagation learning with second order methods," *Proc. 1988 Connectionist Models Summer School*, eds. Touretzky, D. S., Hinton, G. E.

& Sejnowski, T. J. (Morgan Koufmann, San Mateo, CA), pp. 29–37.

Bertsekas, D. P. [1995] *Nonlinear Programming* (Athena Scientific, Belmont, MA).

Booth, A. [1949] "An application of the method of steepest descent to the solution of systems of nonlinear simultaneous equations," *Quart. J. Mech. Appl. Math.* **2**, 460–468.

Brent, R. P. [1973] *Algorithms for Minimization Without Derivatives* (Prentice-Hall, Inc., Englewood Cliffs, NJ).

Burton, R. M. & Mpitsos, G. J. [1992] "Event dependent control of noise enhances learning in neural networks," *Neural Networks* **5**, 627–637.

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

Chan, L. W. & Fallside, F. [1987] "An adaptive training algorithm for back–propagation networks," *Comput. Speech Lang.* **2**, 205–218.

Darken, C., Chiang, J. & Moody, J. [1992] "Learning rate schedules for faster stochastic gradient search," *Proc. IEEE 2nd Workshop on Neural Networks for Signal Processing*, pp. 3–12.

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

Dennis, J. E. & Schnabel R. B. [1983] *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice-Hall, Englewood Cliffs, NJ).

Duda, R. O. & Hart, P. E. [1973] *Pattern Recognition and Scene Analysis* (Wiley, NY).

Fahlman, S. E. [1989] "Faster-learning variations on back–propagation: An empirical study," *Proc. 1988 Connectionist Models Summer School* (Morgan Kaufmann, San Mateo, CA), pp. 38–51.

Gill, P. E., Murray, W. & Wright, M. H. [1981] *Practical Optimization* (Academic Press, NY).

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

Goldstein, A. A. [1965] "On steepest descent," *SIAM J. Contr.* **3**, 147–151.

Gori, M. & Tesi, A. [1992] "On the problem of local minima in backpropagation," *IEEE Trans. Patt. Anal. Mach. Intell.* **14**, 76–85.

Hagan, M. T. & Menhaj, M. [1994] "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Networks* **5**, 989–993.

Hanson, S. J. & Burr, D. J. [1988] "Minkowski–r backpropagation: Learning in connectionist models with non–Euclidean error signals," *Neural Information Processing Systems*, ed. Anderson, D. Z. (American Institute of Physics, NY), pp. 348–357.

Hassoun, M. H. [1995] *Fundamentals of Artificial Neural Networks* (MIT Press, Cambridge, MA).

Haykin, S. [1994] *Neural Networks: A Comprehensive Foundation* (Macmillan College Publishing Company).

Holt, J. L. & Hwang, J. [1993] "Finite precision error analysis of neural network hardware implementations," *IEEE Trans. Comp.* **42**, 281–290.

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

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

Kelley, C. T. [1995] *Iterative Methods for Linear and Nonlinear Equations* (SIAM Publications, Philadelphia).

Kollias, S. & Anastassiou, D. [1989] "An adaptive least squares algorithm for the efficient training of multilayered networks," *IEEE Trans. Circuits Syst.* **36**, 1092–1101.

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

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

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

Magdon–Ismail, M. & Atiya, A. F. [2000] "The early restart algorithm," *Neural Comput.* **12**, 1303–1312.

Magoulas, G. D., Vrahatis, M. N. & Androulakis, G. S. [1996] "A new method in neural network supervised training with imprecision," *Proc. Third IEEE Int. Conf. Electronics, Circuits and Systems*, Rodos, Greece, pp. 287–290.

Magoulas, G. D., Vrahatis, M. N. & Androulakis, G. S. [1997a] "Effective backpropagation with variable stepsize," *Neural Networks* **10**, 69–82.

Magoulas, G. D., Vrahatis, M. N., Grapsa, T. N. & Androulakis, G. S. [1997b] "Neural network supervised training based on a dimension reducing method," *Mathematics of Neural Networks: Models, Algorithms & Applications*, eds. Ellacot, S. W., Mason, J. C. & Anderson, I. J., Chap. 42 (Kluwer Academic Publishers), pp. 245–249.

Magoulas, G. D., Vrahatis, M. N. & Androulakis, G. S. [1999] "Improving the convergence of the backpropagation algorithm using learning rate adaptation methods," *Neural Comput.* **11**, 1769–1796.

Magoulas, G. D. & Vrahatis, M. N. [2000] "A class of adaptive learning rate algorithms derived by one-dimensional subminimization methods," *Neural Paral. Sci. Comput.* **8**, 147–168.

Magoulas, G. D., Plagianakos, V. P. & Vrahatis, M. N. [2001] "Adaptive stepsize algorithms for on–line training of neural networks," *Nonlin. Anal. Th. Meth. Appl.* **47**, 3425–3430.

Magoulas, G. D., Plagianakos, V. P. & Vrahatis, M. N. [2002] "Globally convergent algorithms with local learning rates," *IEEE Trans. Neural Networks* **13**, 774–779.

Magoulas, G. D., Plagianakos, V. P. & Vrahatis, M. N. [2004] "Neural network-based colonoscopic diagnosis using on-line learning and differential evolution," *Appl. Soft Comput.* **4**, 369–379.

Mays, C. H. [1964] "Effects of adaptation parameters on convergence time and tolerance for adaptive threshold elements," *IEEE Trans. Electron. Comput.* **13**, 465–468.

Minai, A. A. & Williams, R. D. [1990] "Back-propagation heuristics: A study of the extended delta-bar-delta algorithm," *Proc. IEEE Int. Joint Conf. Neural Networks*, San Diego, USA, pp. 595–600.

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

Nechiporuk, E. I. [1964] "The synthesis of networks from threshold elements," *Problemy Kibernetiki* **11**, 49–62.

Nocedal, J. [1992] "Theory of algorithms for unconstrained optimization," *Acta Numerica* **1**, 199–242.

Ortega, J. M. & Rheinboldt, W. C. [2000] *Iterative Solution of Nonlinear Equations in Several Variables*, Classics in Applied Mathematics, Vol. 30 (SIAM, Philadelphia).

Parker, D. B. [1987] "Optimal algorithms for adaptive networks: Second order back–propagation, second order direct propagation, and second order Hebbian learning," *Proc. IEEE Int. Conf. Neural Networks* Vol. 2, pp. 593–600.

Pfister, M. & Rojas, R. [1993] "Speeding-up backpropagation — A comparison of orthogonal techniques," *Proc. Joint Conf. Neural Networks*, Nagoya, Japan, pp. 517–523.

Pinkus, A. [1999] "Approximation theory of the MLP model in neural networks," *Acta Numerica* **8**, 143–195.

Plagianakos, V. P., Magoulas, G. D. & Vrahatis, M. N. [2002] "Deterministic nonmonotone strategies for effective training of multi–layer perceptrons," *IEEE Trans. Neural Networks* **13**, 1268–1284.

Plagianakos, V. P. & Vrahatis, M. N. [2002] "Parallel evolutionary trained algorithms for 'hardware-friendly' neural networks," *Natural Comput.* **1**, 307–322.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. F. [1992] *Numerical Recipes in C* (Cambridge University Press, NY).

Ralston, A. & Rabinowitz, P. [1978] *A First Course in Numerical Analysis* (McGraw-Hill, NY).

Reinboldt, W. C. [1974] *Methods for Solving Systems of Nonlinear Equations* (SIAM Publications, Philadelphia, PA).

Riedmiller, M. & Braun, H. [1993] "A direct adaptive method for faster backpropagation learning: The Rprop algorithm," *Proc. IEEE Int. Conf. Neural Networks*, San Francisco, CA, pp. 586–591.

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

Rögnvaldsson, T. [1994] "On Langevin updating in multilayer perceptrons," *Neural Comput.* **6**, 916–926.

Rojas, R. [1996] *Neural Networks*: *A Systematic Introduction* (Springer-Verlag, Berlin).

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. [1986] "Learning internal representations by error propagation," *Parallel Distributed Processing*: *Explorations in the Microstructure of Cognition*, eds. Rumelhart, D. E. & McClelland, J. L., Vol. 1, pp. 318–362.

Saarinen, S., Bramley, R. & Cybenko, G. [1993] "Ill-conditioning in neural network training problems," *Siam J. Sci. Comput.* **14**, 693–714.

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

Sikorski, K. [2001] *Optimal Solution of Nonlinear Equations* (Oxford University Press, NY).

Silva, F. & Almeida, L. [1990] "Acceleration techniques for the back–propagation algorithm," *Lecture Notes in Computer Science* **412**, 110–119.

Siu, K.-Y., Roychowdhury, V. & Kailath, T. [1995] *Discrete Neural Computation*: *A Theoretical Foundation*, Prentice Hall Information and System Sciences Series (Prentice Hall, Englewood Cliffs, NJ).

Treadgold, N. K. & Gedeon, T. D. [1998] "Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm," *IEEE Trans. Neural Networks* **9**, 662–668.

Van der Smagt, P. P. [1994] "Minimization methods for training feedforward neural networks," *Neural Networks* **7**, 1–11.

Varga, R. [2000] *Matrix Iterative Analysis*, 2nd edition (Springer, Berlin).

Vogl, T. P., Mangis, J. K., Rigler, J. K., Zink, W. T. & Alkon, D. L. [1988] "Accelerating the convergence of the back-propagation method," *Biol. Cybern.* **59**, 257–263.

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

Vrahatis, M. N. [1988a] "Solving systems of nonlinear equations using the nonzero value of the topological degree," *ACM Trans. Math. Softw.* **14**, 312–329.

Vrahatis, M. N. [1988b] "CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations," *ACM Trans. Math. Softw.* **14**, 330–336.

Vrahatis, M. N., Androulakis, G. S. & Manoussakis, G. E. [1996] "A new unconstrained optimization method for imprecise function values," *J. Math. Anal. Appl.* **197**, 586–607.

Vrahatis, M. N., Androulakis, G. S., Lambrinos, J. N. & Magoulas, G. D. [2000a] "A class of gradient unconstrained minimization algorithms with adaptive stepsize, *J. Comput. Appl. Math.* **114**, 367–386.

Vrahatis, M. N., Magoulas, G. D. & Plagianakos, V. P. [2000b] "Globally convergent modification of the Quickprop method," *Neural Process. Lett.* **12**, 159–169.

Vrahatis, M. N., Magoulas, G. D. & Plagianakos, V. P. [2003] "From linear to nonlinear iterative methods," *Appl. Numer. Math.* **45**, 59–77.

Watrous, R. L. [1987] "Learning algorithms for connectionist networks: Applied gradient of nonlinear optimization," *Proc. IEEE Int. Conf. Neural Networks*, Vol. 2, pp. 619–627.

White, H. [1990] "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings," *Neural Networks* **3**, 535–549.

Wilson, D. R. & Martinez, T. R. [1997] "Improved heterogeneous distance functions," *J. Artif. Intell. Res.* **6**, 1–34.

Wolfe, P. [1969] "Convergence conditions for ascent methods," *SIAM Rev.* **11**, 226–235.

Wolfe, P. [1971] "Convergence conditions for ascent methods II: Some corrections," *SIAM Rev.* **13**, 185–188.

Wray, J. & Green, G. G. R. [1995] "Neural networks, approximation theory and finite precision computation," *Neural Networks* **8**, 31–37.

Young, D. [1954] "Iterative methods for solving partial difference equations of elliptic type," *Trans. Amer. Math. Soc.* **76**, 92–111.

Yu, X.-H. & Chen, G.-A. [1995] "On the local minima free condition of backpropagation learning," *IEEE Trans. Neural Networks* **6**, 1300–1303.

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