Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational and Applied Mathematics 191 (2006) 166–178
JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS
www.elsevier.com/locate/cam

Improved sign-based learning algorithm derived by the composite nonlinear Jacobi process

Aristoklis D. Anastasiadis[a,*], George D. Magoulas[b], Michael N. Vrahatis[c]

[a] *School of Computer Science and Information Systems, Birkbeck College, University of London, London Knowledge Lab, 23-29 Emerald Street, WC1N 3QS, London, UK*
[b] *School of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London WC1E 7HX, UK*
[c] *Computational Intelligence Laboratory (CI Lab), Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece*

Received 28 February 2005

Abstract

In this paper a globally convergent first-order training algorithm is proposed that uses sign-based information of the batch error measure in the framework of the nonlinear Jacobi process. This approach allows us to equip the recently proposed Jacobi–Rprop method with the global convergence property, i.e. convergence to a local minimizer from any initial starting point. We also propose a strategy that ensures the search direction of the globally convergent Jacobi–Rprop is a descent one. The behaviour of the algorithm is empirically investigated in eight benchmark problems. Simulation results verify that there are indeed improvements on the convergence success of the algorithm.

Keywords: Supervised learning; Nonlinear iterative methods; Nonlinear Jacobi; Pattern classification; Feedforward neural networks; Convergence analysis; Global convergence

1. Introduction

Nowadays, artificial neural networks are vital components of many systems and are considered a powerful tool for pattern classification [5,11]. The vast majority of artificial neural network solutions are trained with supervision.

In this context, the training phase is one of the most important stages for the neural network to function properly and achieve good performance. In supervised learning the desired outputs are supplied by a "teacher" and the network is being forced into producing the correct outputs by adjusting the weights iteratively, in order to globally minimize a measure of the difference between its actual output and the desired output for all examples in a training set [10]. Finding the global minimum is a difficult task in neural networks due to their complex objective function, the so-called error function [10,27].

* Corresponding author.
E-mail address: aris@dcs.bbk.ac.uk (A.D. Anastasiadis).

0377-0427/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2005.06.034

Back-Propagation (BP) [27] is a popular training algorithm, which minimizes the error function by updating the weights w using the steepest descent method [4]:

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

where E is the *batch error measure* defined as the sum-of-squared-differences error function (SSE) over the entire training set, while ∇E denotes the gradient of E. The parameter η is a heuristic, called *step-size*.

Choosing the right value for the step-size is very important as it has an impact on the training speed, the success of the learning process and the quality of the results produced by the network. First-order algorithms with individual adaptive step-sizes provide dynamic tuning of the step-size using adaptation techniques that are able to handle the trade off between maximizing the length of the step-size and reducing oscillations [15,16]. A variety of approaches that use second derivative related information to accelerate the learning process have been proposed for small to medium size networks [4,17,18,31].

An inherent difficulty with first-order and second-order learning schemes is convergence to local minima. While some local minima can provide acceptable solutions, they often result in poor network performance. This problem can be overcome through the use of global optimization at the expense of an increase in the computational cost, particularly for large networks [7,21,22,30].

In this paper we focus on sign-based training schemes. Among these, the Resilient propagation (Rprop) algorithm proposed by Riedmiller and Braun [24–26], is widely used and performs very well in pattern classification tasks. Rprop takes into account only the sign of the derivative to indicate the direction of the weight update. The effectiveness of Rprop in practical applications has motivated the development of several variants with the aim to improve the convergence behaviour and effectiveness of the original method. Recently a modification of the Rprop, the so-called Jacobi–Rprop (JRprop) method has been proposed [2,3]. Empirical evaluations of JRprop gave good results, showing that JRprop outperforms in several cases the Rprop and Conjugate Gradient algorithms [3]. This paper proposes a globally convergent JRprop-based learning scheme and derives a theoretical justification for its development.

The paper is organized as follows. First, we give a brief outline of the theoretical background behind the Jacobi–Rprop algorithm. Next, the new globally convergent algorithm is presented and a theoretical result that justifies its convergence is derived. Then we conduct an empirical evaluation of the new algorithm by comparing it with the classic Rprop, and the recently proposed JRprop [2,3]. Finally our results are discussed and conclusions are drawn.

2. The composite Jacobi-Bisection algorithm

In order to provide a complete view of the proposed approach we briefly describe in this section the composite Jacobi-Bisection method [2,3] that provides the basis for the development of the globally convergent JRprop. The idea is to combine "individual" information about the error surface, described by the sign of the partial derivative of the error function with respect to a weight, with more "global" information from the magnitude of the network learning error, in order to decide for each weight individually whether or not to reduce, or even revert, a step.

Following the nonlinear Jacobi prescription, one-dimensional subminimization is applied along each weight direction in order to compute a minimizer of an objective function $f : \mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}$ [33]. More specifically, starting from an arbitrary initial vector $x^0 \in \mathscr{D}$, one can subminimize at the kth iteration the function $f(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_n^k)$, along the ith direction and obtain the corresponding subminimizer $\hat{x}_i$. Obviously for the subminimizer $\hat{x}_i$ holds that

$$\partial_i f(x_1^k, \ldots, x_{i-1}^k, \hat{x}_i, x_{i+1}^k, \ldots, x_n^k) = 0, \tag{2}$$

where $\partial_i f(x_1, \ldots, x_i, \ldots, x_n)$ denotes the partial derivative of f with respect to the ith coordinate. This is a one-dimensional subminimization because all the components of the vector x^k, except for the ith component, are kept constant. Then the ith component is updated according to

$$x_i^{k+1} = x_i^k + \tau_k(\hat{x}_i - x_i^k), \tag{3}$$

for some relaxation factor τ_k. The objective function f is subminimized in parallel for all i.

In neural network training we have to minimize the batch error function E with respect to each one of the weights w_{ij}. Let us assume that along a weight's direction an interval is known which brackets a local minimum $\hat{w}_{ij}$. When

the gradient of the error function is available at the endpoints of the interval of uncertainty along this weight direction, it is necessary to evaluate function information at an interior point in order to reduce this interval. This is because it is possible to decide if between two successive iterations (k) and $(k-1)$ the corresponding interval brackets a local minimum simply by looking at the function values $E(k-1)$, $E(k)$ and gradient values $\partial E(k-1)/\partial w_{ij}$, $\partial E(k)/\partial w_{ij}$ at the endpoints of the considered interval (see [28] for a general discussion on this issue). The conditions that have to be satisfied are [28, pp. 34–35]

$$\begin{aligned}
&\frac{\partial E(\mathscr{S}_1)}{\partial w_{ij}} < 0 \quad \text{and} \quad \frac{\partial E(\mathscr{S}_2)}{\partial w_{ij}} > 0,\\
&\frac{\partial E(\mathscr{S}_1)}{\partial w_{ij}} < 0 \quad \text{and} \quad E(\mathscr{S}_1) < E(\mathscr{S}_2),\\
&\frac{\partial E(\mathscr{S}_1)}{\partial w_{ij}} > 0 \quad \text{and} \quad \frac{\partial E(\mathscr{S}_2)}{\partial w_{ij}} > 0 \quad \text{and} \quad E(\mathscr{S}_1) > E(\mathscr{S}_2),
\end{aligned} \tag{4}$$

where $\mathscr{S}_1$ and $\mathscr{S}_2$ determine the sets of weights for which the coordinate that corresponds to the weight w_{ij} is replaced by $a_i = \min\{w_{ij}(k-1), w_{ij}(k)\}$, and $b_i = \max\{w_{ij}(k-1),\ w_{ij}(k)\}$ correspondingly. Notice that, at this instance, between two successive iterations $(k-1)$ and (k) all the other coordinate values remain the same. The above three conditions lead to the conclusion that the interval $[a_i, b_i]$ includes a local subminimizer along the direction of weight w_{ij}. A robust method of interval reduction called bisection can now be used. We will consider here the bisection method which has been modified to the following version described in [32]:

$$w_i^{p+1} = w_i^p + h_i \operatorname{sign}(\partial_i E(w^p))/2^{p+1}, \tag{5}$$

where $p = 0, 1, \ldots$ is the number of subminimization steps, $\partial_i E$ denotes the partial derivative of E with respect to the ith coordinate and $w_i^0 = a_i$; $h_i = \operatorname{sign}(\partial_i E(w^0))\ (b_i - a_i)$; w_i^0 determines the weight at the $(k-1)$ iteration while w^p is obtained by replacing the coordinate of w^0 that corresponds to the weight w_{ij} by w_i^p and sign defines the well-known triple-valued sign function. Of course, iterations (5) converge to $\hat{w}_i \in (a_i, b_i)$ if for some w_i^p, $p = 1, 2, \ldots$, the first one of conditions (4) holds. In this case, the bisection method always converges with certainty within the given interval (a_i, b_i).

The reason for choosing the bisection method is that it always converges within the given interval (a_i, b_i), as mentioned above, and it is a globally convergent method. Also, the number of steps of the bisection method that are required for the attainment of an approximate minimizer $\hat{w}_i$ of Eq. (2) within the interval (a_i, b_i) to a predetermined accuracy ε is known beforehand and is given by

$$v = \lceil \log_2[(b_i - a_i)\varepsilon^{-1}] \rceil. \tag{6}$$

Moreover, it has a great advantage since it is the worst-case optimal, i.e. it possesses asymptotically the best possible rate of convergence in the worst-case [29]. This means that it is guaranteed to converge within the predefined number of iterations and moreover, no other method has this property. Therefore, using the value of v of relation (6) it is easy to know in advance the number of iterations necessary to approximate a minimizer $\hat{w}_i$ to a specified degree of accuracy. Finally, it requires only the algebraic signs of the values of the gradient to be computed.

A theoretical result that ensures local convergence of the Jacobi-Bisection algorithm is presented in [3]. Below, we will focus on a composite one-step Jacobi-Bisection method which exhibited very good performance in our tests reported in [3].

3. The globally convergent JRprop

The term global convergence is used in our context in a similar way as in Dennis and Schnabel [8, p. 5] "to denote a method that is designed to converge to a *local* minimizer of a nonlinear function, *from almost any starting point*". Dennis and Schnabel also note that "it might be appropriate to call such methods *local* or *locally convergent*, but these descriptions are already reserved by tradition for another usage". Moreover, Nocedal [20, p. 200] defines a globally convergent algorithm as an algorithm with iterates that converge from a remote starting point. Thus, the notion of global convergence is totally different from global optimization [30]. To this end, equipping JRprop with the global

convergence property will ensure the algorithm will globally converge to a local minimum starting from any initial condition.

First, let us recall some concepts from the theory of unconstrained minimization. Suppose that (i) $f : \mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}$ is the function to be minimized and f is bounded below in $\mathbb{R}^n$; (ii) f is continuously differentiable in a neighbourhood $\mathscr{N}$ of the level set $\mathscr{L} = \{x : f(x) \leqslant f(x^0)\}$, and (iii) the gradient of f, ∇f, is Lipschitz continuous on $\mathbb{R}^n$, that is, there exists a Lipschitz constant $L > 0$ such that $\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|$, $\forall x, y, \in \mathscr{N}$, and x^0 is the starting point of the following iterative scheme:

$$x^{k+1} = x^k + \tau^k d^k, \tag{7}$$

where d^k is the search direction and $\tau^k > 0$ is a step-length obtained by means of a one-dimensional search.

Convergence of the general iterative (7) requires that the search direction d^k satisfies the condition $\nabla f(w^k)^\top d^k < 0$, which guarantees that d^k is a descent direction of $f(x)$ at x^k. The step-length τ^k in (7) can be determined by means of a number of rules, such as Armijo's rule [8], Goldstein's rule [8], or Wolfe's rule [34], and guarantees the convergence in certain cases. For example, when the step-length is obtained through Wolfe's rule [34]

$$f(x^k + \tau^k d^k) - f(x^k) \leqslant \sigma_1 \tau^k \nabla f(x^k)^\top d^k, \tag{8}$$

$$\nabla f(x^k + \tau^k d^k)^\top d^k \geqslant \sigma_2 \nabla f(x^k)^\top d^k, \tag{9}$$

where $0 < \sigma_1 < \sigma_2 < 1$, then a theorem by Wolfe [34] is used to obtain convergence results. Moreover, Wolfe's Theorem suggests that if the cosine of the angle between the search direction d^k and $-\nabla f(x^k)$ is positive then

$$\lim_{k\to\infty} \|\nabla f(x^k)\| = 0, \tag{10}$$

which means that the sequence of gradients converges to zero [8,20]. For an iterative scheme (7), limit (10) is the best type of global convergence result that can be obtained (see [20] for a detailed discussion). Evidently, no guarantee is provided that (7) will converge to a global minimizer, x^*, but only that it possesses the global convergence property [8,20] to a local minimizer.

In batch training, when the batch error measure is defined as the sum-of-squared-differences error function E over the entire training set, the error function E is bounded from below, since $E(w) \geqslant 0$. For a given training set and network architecture, if a w^* exists such that $E(w^*) = 0$, then w^* is a global minimizer; otherwise, w with the smallest $E(w)$ value is considered a global minimizer. Also, when using *smooth enough* activations (the derivatives of at least order p are available and continuous), such as the well-known hyperbolic tangent, the logistic activation function, etc., the error E is also smooth enough.

Based on the above we proceed with the following convergence result for JRprop's scheme.

Theorem 1. *Suppose that for the error function E conditions* (i)–(iii) *are fulfilled. Then, for any* $w^0 \in \mathbb{R}^n$ *and any sequence* $\{w^k\}_{k=0}^{\infty}$ *generated by JRprop's scheme*

$$w^{k+1} = w^k - \tau^k \operatorname{diag}\{\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k\} \operatorname{sign}(\nabla E(w^k)), \tag{11}$$

where $\operatorname{sign}(\nabla E(w^k))$ *denotes the column vector of the signs of the components of* $\nabla E(w^k) \equiv (\partial_1 E(w^k), \partial_2 E(w^k), \ldots, \partial_n E(w^k))$, $\tau^k > 0$, *satisfies Wolfe's conditions* (8)–(9), η_m^k $(m = 1, 2, \ldots, i-1, i+1, \ldots, n)$ *are small positive real numbers generated by the JRprop learning rates' schedule*:

$$\begin{aligned}
&\textit{if} \quad E(w^k) \leqslant E(w^{k-1}) \,\{ \\
&\textit{if} \quad (\partial_m E(w^{k-1}) \cdot \partial_m E(w^k) > 0) \quad \textit{then} \quad \eta_m^k = \min\{\eta_m^{k-1} \cdot \eta^+, \Delta_{\max}\} && (12)\\
&\textit{if} \quad (\partial_m E(w^{k-1}) \cdot \partial_m E(w^k) < 0) \quad \textit{then} \quad \eta_m^k = \max\{\eta_m^{k-1} \cdot \eta^-, \Delta_{\min}\} && (13)\\
&\textit{if} \quad (\partial_m E(w^{k-1}) \cdot \partial_m E(w^k) = 0) \quad \textit{then} \quad \eta_m^k = \eta_m^{k-1} && (14)\\
&\},
\end{aligned}$$

where $0 < \eta^- < 1 < \eta^+$, $\Delta_{\max}$ *is the learning rate upper bound*, $\Delta_{\min}$ *is the learning rate lower bound, and*

$$\eta_i^k = -\frac{\sum_{\substack{j=1 \\ j\neq i}}^{n} \eta_j^k \partial_j E(w^k) + \delta}{\partial_i E(w^k)}, \quad 0 < \delta \ll \infty, \quad \partial_i E(w^k) \neq 0, \tag{15}$$

holds that $\lim_{k\to\infty} \|\nabla E(w^k)\| = 0$.

Proof. Evidently, E is bounded below on $\mathbb{R}^n$. The sequence $\{w^k\}_{k=0}^{\infty}$ generated by the iterative (11) follows the direction

$$d^k = -\text{diag}\{\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k\} \,\text{sign}(\nabla E(w^k)),$$

which is a descent direction if η_m^k, where $m = 1, 2, \ldots, i-1, i+1, \ldots, n$, are positive real numbers derived from relations (12)–(14), and η_i^k is given by relation (15), since $\nabla E(w^k)^\top d^k < 0$. Following the proof of [33, Theorem 6], since d^k is a descent direction and E is continuously differentiable and bounded below along the radius $\{w^k + \tau d^k \mid \tau > 0\}$, then there always exist τ^k satisfying relations (8)–(9) [8,20]. Moreover, Wolfe's Theorem [8,20] suggests that if the cosine of the angle between the descent direction d^k and the $-\nabla E(w^k)$ is positive then $\lim_{k\to\infty} \|\nabla E(w^k)\| = 0$. In our case, indeed $\cos\theta_k = -\nabla E(w^k)^\top d^k / \|\nabla E(w^k)\| \, \|d^k\| > 0$. □

The Globally convergent modification of the JRprop, named GJRprop, is implemented through relations (11)–(15). It is also important to mention that in case of an error increase, the corresponding weight update procedure of JRprop, described in [3], is adopted. The role of δ is to alleviate problems with limited precision that may occur in simulations, and should take a small value proportional to the square root of the relative machine precision. In our tests we set $\delta = 10^{-6}$ in an attempt to test the convergence accuracy of the proposed strategy. Also $\tau^k = 1$ for all k allows the minimization step along the resultant search direction to be explicitly defined by the values of the local learning rates $(\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k)$. The length of the minimization step can be regulated through τ^k tuning to satisfy conditions (8)–(9). Checking condition (9) at each iteration requires additional gradient evaluations; thus, in practice condition (9) can be enforced simply by placing the lower bound on the acceptable values of the learning rates [16, p. 1772], i.e. $\Delta_{\min}$.

4. Empirical study

In this section, we evaluate the performance of the GJRprop, and compare it with the JRprop and the Rprop algorithms. We have used well-studied problems from the UCI Repository of Machine Learning Databases of the University of California [19], as well as problems studied extensively by other researchers, such as the parity-*N* problems that possess strong local minima and stationary points. Literature suggests standard neural architectures for these problems, so it helps us to reduce as much as possible biases introduced by the size of the weight space. In all cases we have used networks with classic logistic activations. Below, we report results from 150 independent trials. These 150 random weight initializations are the same for all the learning algorithms. In all cases we have used networks with sigmoid hidden and output nodes, and adopted the notation I-H-O to denote a network architecture with I inputs, H hidden layer nodes and O outputs nodes.

For the UCI problems, cancer1, diabetes1, thyroid1, and *Escherichia coli* (*E. coli*), we have used the data sets as supplied on the PROBEN1 website [23]. PROBEN1 provides explicit instructions for generating training and test sets, and choosing network architectures [23]. The data set for the *E. coli* problem was used as supplied on the UCI repository and the sets for training and testing were generated following guidelines published by Horton [12].

The results reported below present the average number of iterations (epochs), the average training time to reach the error goal ± the corresponding value of standard deviation, the average generalization (generalization is measured as the percentage of correctly classified test patterns), and the percentage of convergence success (this percentage is calculated out of 150 runs).

In all experiments the parameters have been set as follows: $\eta^+ = 1.2$; $\eta^- = 0.5$; $\Delta_{ij}^0 = \eta^0 = 0.1$; $\Delta_{\max} = 50$ [26]. Finally, we have set $\delta = 10^{-6}$ in an attempt to test the convergence accuracy of the proposed strategy and also $\tau^k = 1$ for all k.

Table 1
Comparison of algorithm performance in the cancer problem for the converged runs

Cancer				
Algorithm	Iterations	Time (s)	Generalization (%)	Convergence (%)
Rprop	234	1.6 ± 0.60	97.4	95
JRprop	150	1.2 ± 0.50	97.2	96
GJRprop	137	1.0 ± 0.38	97.5	99

Table 2
Number of times, out of 150 runs, each algorithm performs better than the other methods in the cancer problem with respect to training speed and generalization

Cancer	Times faster algorithm			Times better generalization		
Algorithm	Rprop	JRprop	GJRprop	Rprop	JRprop	GJRprop
Rprop	—	55	25	—	20	13
JRprop	93	—	57	42	—	36
GJRprop	125	102	—	53	50	—

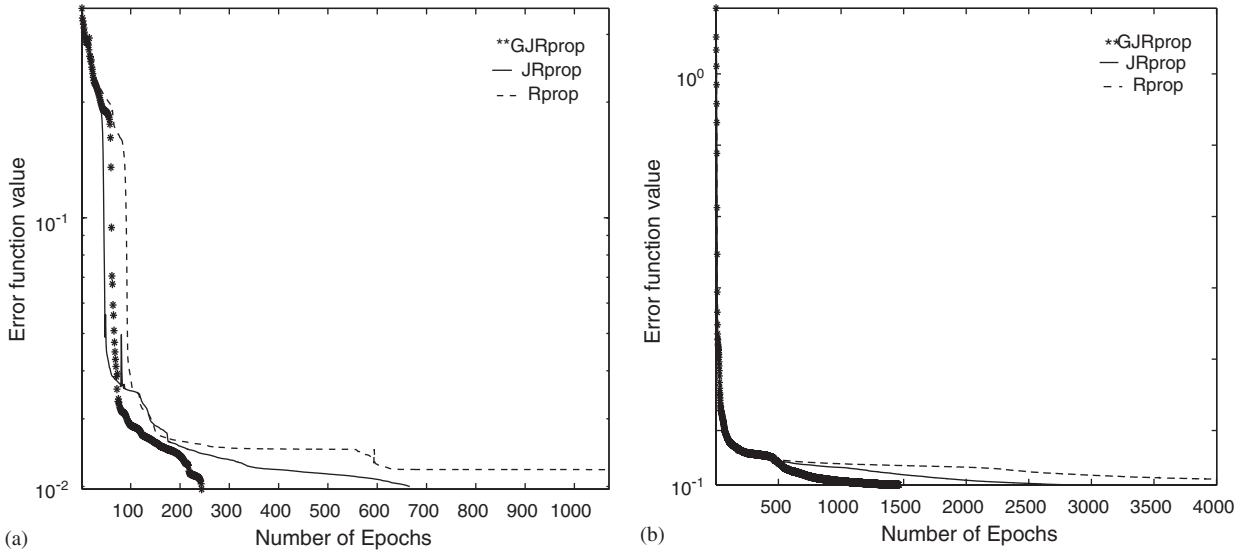

Fig. 1. GJRprop, JRprop and Rprop learning curves for (a) the cancer problem and (b) the diabetes problem.

4.1. The Cancer1 problem

The breast cancer diagnosis problem is based on nine inputs describing a tumour as benign or malignant. The data set consists of 350 patterns. We used a feed-forward neural network with 9-4-2-2 nodes as suggested in the PROBEN1 benchmark collection and in [3]. The error goal in training was $E < 0.02$ to harmonize with the training errors obtained in [3,13]. The results for this pattern classification problem are summarized in Table 1. The new algorithm performs significantly better than the other two methods. Table 2 presents the number of times each algorithm outperforms the other methods in terms of training speed and generalization within 150 independent runs. It yields that the new learning scheme is frequently faster and achieves better generalization than the other two members of the Rprop family. Fig. 1(a) presents an example of convergence behaviour starting from the same initial conditions: the Rprop converges to a local

Table 3
Comparison of algorithm performance in the diabetes problem for the converged runs

Diabetes				
Algorithm	Iterations	Time (s)	Generalization (%)	Convergence (%)
Rprop	380	2.3 ± 2.0	75.5	90
JRprop	310	1.9 ± 1.5	75.4	93
GJRprop	290	1.7 ± 0.8	75.8	98

Table 4
Number of times, out of 150 runs, each algorithm performs better than the other methods in the diabetes problem with respect to training speed and generalization

Diabetes	Times faster algorithm			Times better generalization		
Algorithm	Rprop	JRprop	GJRprop	Rprop	JRprop	GJRprop
Rprop	—	45	38	—	18	20
JRprop	68	—	51	30	—	20
GJRprop	101	70	—	52	50	—

minimizer, whilst both JRprop and GJRprop converge to a feasible solution ($E \leqslant 10^{-2}$) with GJRprop outperforming all other methods.

4.2. The Diabetes1 problem

The aim of this real-world classification task is to decide when a Pima Indian individual is diabetes positive or not. We have eight inputs representing personal data and results from a medical examination. The data set consists of 384 patterns. The PROBEN1 collection proposes several architectures for this problem, including one with 8-2-2-2 nodes. We decided to use this architecture as it was also suggested by others [3,13]. The error goal in this case was set at 0.14 to conform to the training error obtained in [3,13].

Table 3 summarizes the performance of the tested algorithms. The increased training speed does not affect the generalization performance of the new method. It is worth noting the standard deviation value of the GRprop is significantly less than the corresponding Rprop and JRprop values, which means GJRprop performance is closer to the average value. Table 4 gives an analytic view of the comparative results in the 150 trials. Fig. 1(b) illustrates a training instance where all the methods start under the same initial conditions: the Rprop converges to a local minimizer, whilst both JRprop and GJRprop converge to a solution with $E \leqslant 10^{-1}$.

4.3. The Escherichia coli problem

This problem concerns the classification of the *E. coli* protein localization patterns into eight localization sites. *E. coli*, being a prokaryotic gram-negative bacterium, is an important component of the biosphere. Three major and distinctive types of proteins are characterized in *E. coli*: enzymes, transporters and regulators. The largest number of genes encodes enzymes (34%) (this should include all the cytoplasm proteins) followed by the genes for transport functions and the genes for regulatory process (11.5%) [14].

In these experiments the neural networks were tested using four-fold cross-validation, as this approach has been used before in the literature for training probabilistic and nearest-neighbor classifiers in this problem [12]. The best available architecture that was suggested is a 7-16-8 FNN [1]. Rprop-trained FNNs of this architecture achieved better generalization than the best results reported in the literature [12], when the training error goal was $E < 0.02$ [1].

Results from 150 runs for three algorithms using the same architecture are given in Table 5. A detailed account of the algorithms' performance is exhibited in Table 6. Fig. 2(a) illustrates the behaviour of the training algorithms in a

Table 5
Comparison of algorithm performance in the *E.coli* problem for the converged runs

E. coli				
Algorithm	Iterations	Time (s)	Generalization (%)	Convergence (%)
Rprop	140	1.25 ± 0.31	90.0	99
JRprop	130	1.15 ± 0.25	90.0	99
GJRprop	125	1.10 ± 0.20	90.1	100

Table 6
Number of times, out of 150 runs, each algorithm performs better than the other methods in the *E. coli* problem with respect to training speed and generalization

E. coli	Times faster algorithm			Times better generalization		
Algorithm	Rprop	JRprop	GJRprop	Rprop	JRprop	GJRprop
Rprop	—	62	61	—	55	49
JRprop	87	—	70	70	—	67
GJRprop	86	73	—	87	70	—

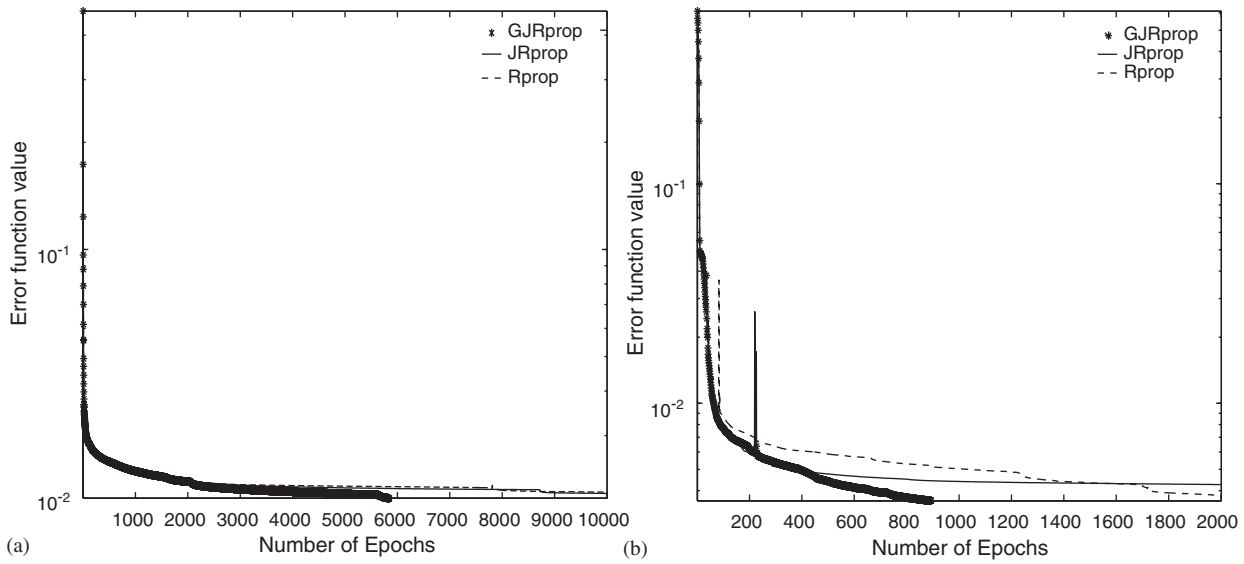

Fig. 2. GJRprop, JRprop and Rprop learning curves for (a) the *E. coli* problem and (b) the thyroid problem.

case where $E \leqslant 0.01$. Convergence to a feasible solution is achieved by GJRprop within 6000 iterations while the other schemes require more than 10000 iterations.

4.4. The thyroid problem

In this problem, the aim is to find whether the patient's thyroid has over-function, normal function, or under-function. We used the *thyroid*1 dataset (3600 patterns), a network with 21-4-3 nodes, and the error goal was set at 0.0036, as suggested in [3,30].

Comparative results are given in Table 7. GJRprop outperforms the other algorithms. Moreover, the value of the deviation of the new algorithm is significantly lower than the standard deviation of the other two methods (see

Table 7
Comparison of algorithm performance in the thyroid problem for the converged runs

Thyroid				
Algorithm	Iterations	Time (secs)	Generalization (%)	Convergence (%)
Rprop	710	23.90 ± 12.5	98.12	87
JRprop	640	21.40 ± 10.1	98.12	89
GJRprop	620	19.90 ± 7.5	98.23	95

Table 8
Number of times, out of 150 runs, each algorithm performs better than the other methods in the thyroid problem with respect to training speed and generalization

Thyroid	Times faster algorithm			Times better generalization		
Algorithm	Rprop	JRprop	GJRprop	Rprop	JRprop	GRprop
Rprop	—	71	64	—	66	57
JRprop	79	—	66	69	—	60
GJRprop	86	84	—	78	76	—

Table 7). Finally, it is worth mentioning that the GJRprop exhibits significantly improved convergence success compared to the other tested algorithms. This can be attributed to the ability of new globally convergent algorithm to follow descent directions.

A detailed account of the algorithms' performance is exhibited in Table 8. The new learning scheme is faster than Rprop and JRprop 86 and 84 times, respectively. In terms of generalization success, GJRprop outperforms Rprop and JRprop 78 and 76 times, respectively.

Fig. 2(b) illustrates a case where GJRprop converges to a minimizer while Rprop and JRprop get stuck at local minimizers with higher error values. As shown in Fig. 2(b) Rprop's and JRprop's learning curves exhibit nonmonotone behaviour denoted by two hard peaks: one around time point 100 for the Rprop, and the other around point 200 for the JRprop. The GJRprop decreases monotonically the error function as it always follows a descent direction.

4.5. Boolean function approximation problems

Another set of experiments has been conducted to empirically evaluate the performance of the globally convergent method in a well-studied class of boolean function approximation problems that exhibit strong local minima and stationary points [6,9]. These problems include the XOR problem (whose local minima and saddle points have been analysed in detail) and the various parity-N problems, which are considered as classic benchmarks [15,21,31]. The adopted architectures were 2-2-1 for the XOR, 3-3-1 for the parity-3, 4-4-1 for the parity-4, 5-5-1 for the parity-5.

For the XOR problem the error target was set to $E \leqslant 10^{-5}$ within 2000 iterations and for the parity-5 problem was set to $E \leqslant 10^{-6}$, while for the other remaining boolean function approximation problems the acceptable solution was set at $E \leqslant 5 \times 10^{-5}$. All these target values are considered low enough to guarantee convergence to a "global" solution.

4.5.1. XOR problem

Table 9 shows the performance of each algorithm. GJRprop exhibits better convergence success than other methods: GJRprop achieved 79% average convergence success, while Rprop and JRprop achieved on average 62% and 68%. The GJRprop outperforms significantly over the Rprop in terms of convergence speed and has relatively similar speed with the JRprop.

Fig. 3(a) gives an example of algorithms' convergence. Starting from the same initial conditions, the Rprop and the JRprop converge to a local minimizer, whilst GJRprop reaches a lower value.

Table 9
Comparison of algorithm performance in the XOR problem for the converged runs

XOR			
Algorithm	Iterations	Time (s)	Convergence (%)
Rprop	160	1.57 ± 1.0	62
JRprop	105	1.22 ± 0.6	68
GJRprop	112	1.26 ± 0.5	79

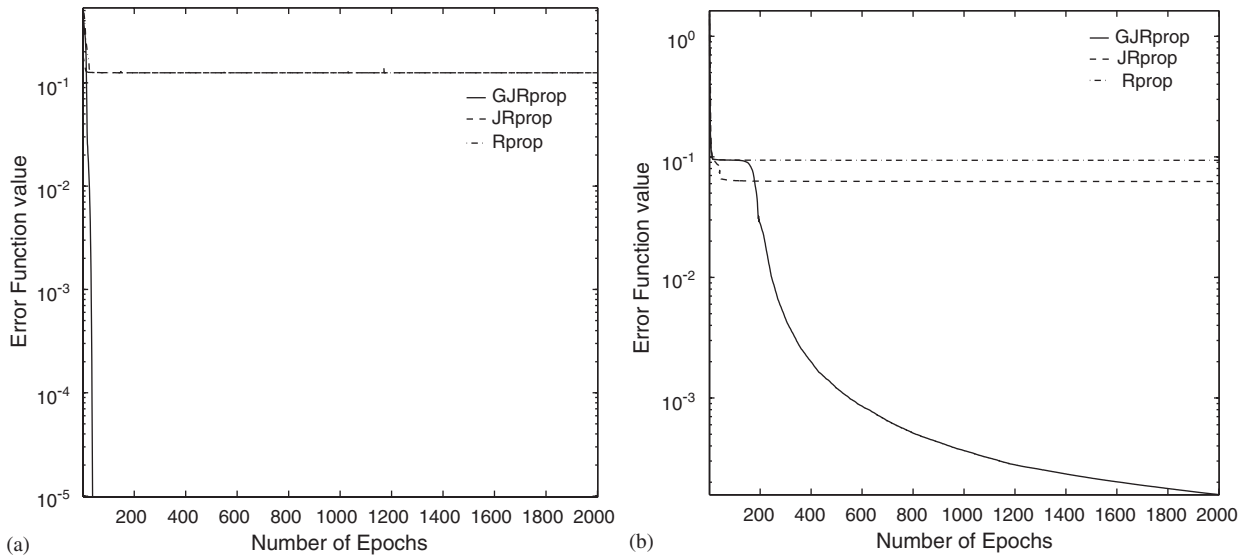

Fig. 3. Learning error curves for (a) the XOR problem and (b) the parity-3.

Table 10
Comparison of algorithm performance in the parity-3 problem for the converged runs

Parity-3			
Algorithm	Iterations	Time (s)	Convergence (%)
Rprop	885	3.8 ± 1.9	79
JRprop	850	3.5 ± 1.6	77
GJRprop	840	3.3 ± 1.3	88

4.5.2. Parity-3 problem

Table 10 presents comparative results in terms of training speed (in s) and convergence success for the 150 runs. It presents the average training time and the corresponding standard deviation for each algorithm calculated over the converged runs. GJRprop shows an increase in the percentage of convergence success. The globally convergent scheme manages to escape from some local minima and finds acceptable solutions with higher possibility than the other two tested methods do. Finally, Fig. 3(b) shows a case where Rprop and JRprop converge to local minima while GJRprop reaches a minimizer with lower function value for the parity-3 problem.

4.5.3. Parity-4 problem

Comparative results for the parity-4 problem are given in Table 11. GJRprop outperforms the other algorithms particularly in convergence success. It achieves to meet the error goal with 91% success whilst Rprop and JRprop

Table 11
Comparison of algorithm performance in the parity-4 problem for the converged runs

Parity-4			
Algorithm	Iterations	Time (s)	Convergence (%)
Rprop	810	5.7 ± 3.4	77
JRprop	720	4.8 ± 3.0	81
GJRprop	615	4.2 ± 2.2	91

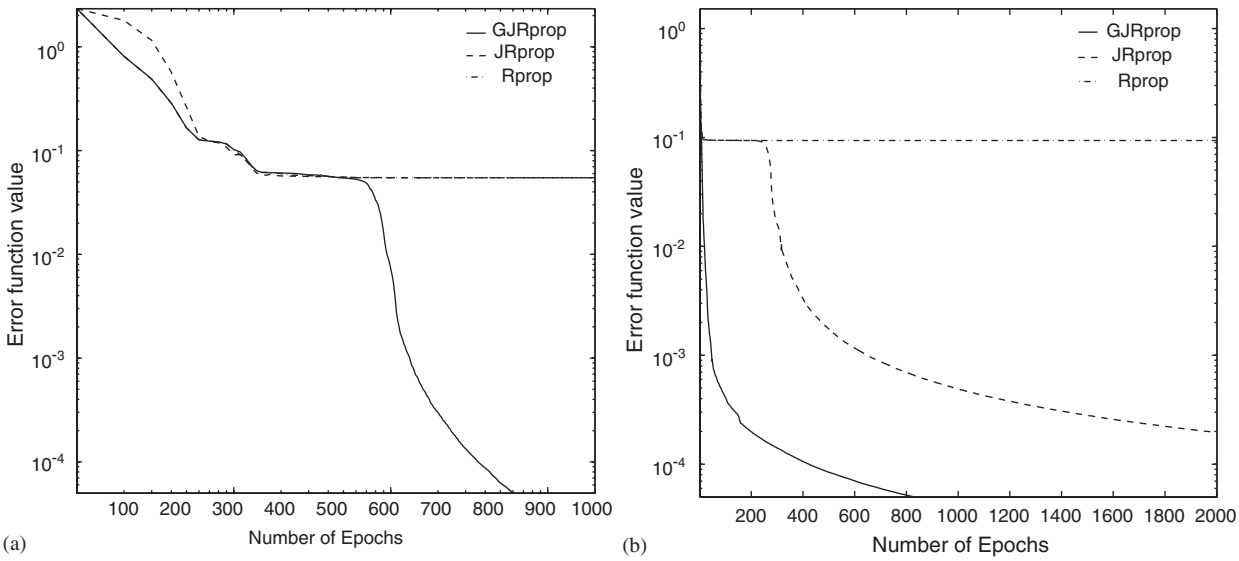

Fig. 4. Typical learning error curves for (a) the parity-4 problem and (b) the parity-5 problem.

Table 12
Comparison of algorithm performance in the parity-5 problem for the converged runs

Parity-5			
Algorithm	Iterations	Time (s)	Convergence (%)
Rprop	950	6.7 ± 3.6	61
JRprop	760	4.1 ± 2.0	64
GJRprop	795	4.5 ± 2.1	82

have significantly less convergence success. Fig. 4(a) illustrates a case where GJRprop converges to a minimizer while Rprop and JRprop get stuck at a local minimizer with higher error value.

4.5.4. Parity-5 problem

Comparative results for the parity-5 problem are presented in Table 12. The JRprop algorithm achieves the best training speed, while GJRprop exhibits comparable performance. Both of them outperform the Rprop algorithm. Furthermore, the GJRprop is more stable and shows an important convergence improvement over the other tested methods. Fig. 4(b) illustrates a case where GJRprop converges to an acceptable minimizer while the other methods converge to local minima with higher function values.

5. Conclusions

It is widely accepted that the Rprop algorithm is one of the best performing sign-based learning algorithms for neural networks with arbitrary topology. In this paper we built on the nonlinear Jacobi process to develop a globally convergent composite Jacobi–Rprop. In our experiments, the GJRprop exhibited better training speed than Rprop and JRprop in six out of the eight benchmarks, and demonstrated stability in locating minimizers with a high percentage of success in all cases. The comparative study reported in the paper also showed that GJRprop exhibits more consistent behaviour than the other algorithms. Nevertheless, we acknowledge that this is a small scale study. Further research into the performance of the method is needed to fully explore its advantages and identify possible limitations in pattern recognition problems.

References

[1] A.D. Anastasiadis, G.D. Magoulas, X. Liu, Classification of protein localisation patterns via supervised neural network learning, in: Proceedings of the Fifth Symposium on Intelligent Data Analysis (IDA-03), Berlin, Germany, August 2003, Lecture Notes in Computer Science, vol. 2810, Springer, Berlin, 2003, pp. 430–439.
[2] A.D. Anastasiadis, G.D. Magoulas, M.N. Vrahatis, An efficient improvement of the Rprop algorithm, in: Proceedings of the First International Workshop on Artificial Neural Networks in Pattern Recognition (IAPR 2003), University of Florence, Italy, 2003, pp.197–201.
[3] A.D. Anastasiadis, G.D. Magoulas, M.N. Vrahatis, Sign-based learning schemes for pattern classification, Pattern Recognition Lett. 26 (2005) 1926–1936.
[4] R. Battiti, First-and second-order methods for learning: between steepest descent and Newton's method, Neural Computation 4 (1992) 141–166.
[5] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford, 1995.
[6] E.K. Blum, Approximation of Boolean functions by sigmoidal networks: Part I: XOR and other two variable functions, Neural Computation 1 (1989) 532–540.
[7] R.M. Burton, G.J. Mpitsos, Event dependent control of noise enhances learning in neural networks, Neural Networks 5 (1992) 627–637.
[8] J.E. Dennis, R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, SIAM, Philadelphia, 1996.
[9] M. Gori, A. Tesi, On the problem of local minima in backpropagation, IEEE Trans. Pattern Analysis and Machine Intelligence 14 (1992) 76–85.
[10] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Company, 1994.
[11] J. Hertz, A. Krogh, R. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, MA, 1991.
[12] P. Horton, K. Nakai, Better prediction of protein cellular localization sites with the *k* nearest neighbors classifier, in: Proceedings of the Intelligent Systems in Molecular Biology, 1997, pp. 368–383.
[13] C. Igel, M. Husken, Empirical evaluation of the improved Rprop learning algorithms, Neurocomputing 50 (2003) 105–123.
[14] P. Liang, B. Labedan, M. Riley, Physiological genomics of *Escherichia coli* protein families, Physiological Genomics 9 (1) (2002) 15–26.
[15] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Effective back-propagation training with variable stepsize, Neural Networks 10 (1997) 69–82.
[16] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Improving the convergence of the backpropagation algorithm using learning rate adaptation methods, Neural Computation 11 (1999) 1769–1796.
[17] G.D. Magoulas, M.N. Vrahatis, T.N. Grapsa, G.S. Androulakis, Neural network supervised training based on a dimension reducing method, in: S.W. Ellacot, J.C. Mason, I.J. Anderson (Eds.), Mathematics of Neural Networks: Models, Algorithms and Applications, Kluwer Academic Publishers, Dordrecht, 1997, pp. 245–249.
[18] M.F. Möller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks 6 (1993) 525–533.
[19] P.M. Murphy, D.W. Aha, UCI Repository of machine learning databases, Irvine, Department of Information and Computer Science, University of California, CA, 1994 ⟨http://www.ics.uci.edu/mlearn/MLRepository.html⟩.
[20] J. Nocedal, Theory of algorithms for unconstrained optimization, Acta Numerica (1992) 199–242.
[21] V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Learning in multilayer perceptrons using global optimization strategies, Nonlinear Anal.: Theory, Methods and Applications 47 (2001) 3431–3436.
[22] V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Supervised training using global search methods, in: N. Hadjisavvas, P. Pardalos (Eds.), Advances in Convex Analysis and Global Optimization, vol. 54, Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001, pp. 421–432.
[23] L. Prechelt, PROBEN1—A set of benchmarks and benchmarking rules for neural network training algorithms, T.R. 21/94, Fakultät für Informatik, Universität, Karlsruhe, 1994.
[24] M. Riedmiller, Rprop—Description and Implementation Details Technical Report, University of Karlsruhe, January 1994.
[25] M. Riedmiller, Advanced supervised learning in multilayer perceptrons—from backpropagation to adaptive learning techniques, Internat. J. Comput. Standards and Interfaces 16 (1994) 265–278 (Special Issue on Neural Networks).
[26] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the Rprop algorithm, in: Proceedings of the International Conference on Neural Networks, San Francisco, CA, 1993, pp. 586–591.

[27] D.E. Rumelhart, J.L. McClellend, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge, 1986, pp. 318–362.
[28] L.E. Scales, Introduction to Non-linear Optimization, MacMillan Publishers Ltd., 1985, pp. 34–35.
[29] K. Sikorski, Optimal Solution of Nonlinear Equations, Oxford University Press, New York, 2001.
[30] N.K. Treadgold, T.D. Gedeon, Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm, IEEE Trans. Neural Networks 9 (4) (1998) 662–668.
[31] P.P. Van der Smagt, Minimization methods for training feedforward neural networks, Neural Networks 7 (1994) 1–11.
[32] M.N. Vrahatis, Solving systems of nonlinear equations using the nonzero value of the topological degree, ACM Trans. Math. Software 14 (1988) 312–329;
M.N. Vrahatis, Solving systems of nonlinear equations using the nonzero value of the topological degree, ACM Trans. Math. Software 14 (1988) 330–336.
[33] M.N. Vrahatis, G.D. Magoulas, V.P. Plagianakos, From linear to nonlinear iterative methods, Appl. Numer. Math. 45 (1) (2003) 59–77.
[34] P. Wolfe, Convergence conditions for ascent methods, SIAM Rev. 11 (1969) 226–235;
P. Wolfe, Convergence conditions for ascent methods, SIAM Rev. 13 (1971) 185–188