Available online at www.sciencedirect.com

SCIENCE DIRECT®

Neurocomputing 64 (2005) 253–270

NEUROCOMPUTING

www.elsevier.com/locate/neucom

New globally convergent training scheme based on the resilient propagation algorithm

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

[a]*School of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London WC1E 7HX, UK*

[b]*Computational Intelligence Laboratory, Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece*

Available online 22 January 2005

Abstract

In this paper, a new globally convergent modification of the Resilient Propagation-Rprop algorithm is presented. This new addition to the Rprop family of methods builds on a mathematical framework for the convergence analysis that ensures that the adaptive local learning rates of the Rprop's schedule generate a descent search direction at each iteration. Simulation results in six problems of the PROBEN1 benchmark collection show that the globally convergent modification of the Rprop algorithm exhibits improved learning speed, and compares favorably against the original Rprop and the Improved Rprop, a recently proposed Rrpop modification.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Supervised learning; Batch learning; First-order training algorithms; Convergence analysis; Global convergence property; Rprop; IRprop

*Corresponding author.
E-mail addresses: aris@dcs.bbk.ac.uk (A.D. Anastasiadis), gmagoulas@dcs.bbk.ac.uk (G.D. Magoulas), vrahatis@math.upatras.gr (M.N. Vrahatis).

0925-2312/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2004.11.016

1. Introduction

A number of advanced algorithms have been proposed so far in neural networks learning. Methods such as conjugate gradients [13], the Levenberg–Marquardt algorithm [6] and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [5] are considered popular choices for training feedforward neural networks. These algorithms attempt to use second derivative-related information to accelerate the learning process [2]. Although, the capacity of modern computers has been improved considerably the last few years, there are still problems that hamper the use of these powerful second-order algorithms in some problems. For example, a large number of weights often makes the direct application of second-order methods impractical. Moreover, these methods use approximations of the Hessian matrix which some time during training may become close to singular, or badly scaled, and as a consequence they might produce inaccurate results. Lastly, it is not certain that the extra computational cost required by the second-order methods speeds up the minimization process for nonconvex functions when far from the minimizer [17].

As a result, gradient descent methods are still the most widely used class of algorithms for supervised learning of neural networks. The most popular training algorithm of this category is the batch back-propagation (BP) [22]. It is a first-order method that minimizes the error function by updating the weights using the steepest descent method [2]

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

where E is the batch error measure defined as the sum of squared differences error function (SSE) over the entire training set and $\nabla E(w^k) = g(w^k)$ is the gradient vector of E. The parameter η is a heuristic, called learning rate. Proper learning rate values help to avoid convergence to a saddle point or a maximum. In order to secure the convergence of the BP training algorithm and avoid oscillations in a steep direction of the error surface a small learning rate is chosen ($0<\eta<1$). However, it is well known that this approach tends to be inefficient.

Adaptive gradient-based algorithms with individual step sizes try to overcome the inherent difficulty of choosing the right learning rates for each region of the search space depending on the application [11,12]. This is done by controlling the weight update of each weight in order to minimize oscillations and at the same time maximize the length of the step size. One of the best algorithms of this class, in terms of convergence speed, accuracy and robustness with respect to its learning parameters, is the Resilient backpropagation (Rprop) algorithm introduced by Riedmiller and Braun [21].

The effectiveness of Rprop in practical applications has motivated the development of several variants aiming at improving the convergence behavior and effectiveness of the original method. These variants can be roughly categorized into (i) hybrid learning schemes that equip Rprop with second derivative related information, such as the *QRprop* algorithm, which approximates the second derivative by one-dimensional secant steps, and the *Diagonal Estimation Rprop—DERprop* [19], which directly computes the diagonal elements of the Hessian matrix; (ii) approaches inspired from global optimization theory that combine Rprop and annealing strategies, such as the *simulated annealing Rprop—SARprop* and the *restart mode simulated annealing Rprop—ReSARprop* [23]. Recently, the *Improved Rprop—IRprop* algorithm [8] has shown improved convergence speed when compared against existing Rprop variants, as well as the conjugate gradients and the BFGS training methods.

Relevant literature shows that Rprop-based learning schemes exhibit fast convergence in empirical evaluations, but usually require introducing or even fine tuning additional heuristics. For example, annealing schedules require heuristics for the acceptance probability and the visiting distribution, while second derivative methods employ heuristics in the various approximations of the second derivatives. Moreover, literature shows a lack of theoretical results underpinning the development of Rprop modifications, particularly in the case of hybrid schemes. This is not surprising as heuristics make it difficult to guarantee convergence to a local minimizer of the error function when adaptive learning rates for each weight are used in the calculation weight updates [8,12,19,21].

This paper proposes a new Rprop-based learning scheme and presents a theoretical justification for its development. In the next section, the Rprop algorithm is described and the basic ideas behind the recently proposed IRprop are reviewed. Next, the new algorithm and the corresponding theoretical result are presented. Then, results on the experimental evaluation of the algorithm as well as comparisons with the original Rprop and the IRprop are reported. The paper ends with short discussion and concluding remarks.

2. The Rprop and the improved-Rprop algorithms

The Rprop algorithm employs a sign-based scheme to update the weights in order to eliminate harmful influences of the derivatives' magnitude on the weight updates, i.e. the direction of the update along each weight direction only depends on the sign of the corresponding derivative. This approach is considered eminently suitable for applications where the gradient is numerically estimated or the error is noisy [8]; it is easy to implement it in hardware and is not susceptible to numerical problems [18]. The size of the update step along a weight direction is exclusively determined by a weight-specific "update-value"

$$\Delta w_{ij}^k = \begin{cases} -\varDelta_{ij}^k & \text{if } \dfrac{\partial E(w^k)}{\partial w_{ij}} > 0, \\ +\varDelta_{ij}^k & \text{if } \dfrac{\partial E(w^k)}{\partial w_{ij}} < 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $\partial E(w^k)/\partial w_{ij}$ denotes the partial derivative of the batch error with respect to w_{ij} at the kth iteration. The second step of Rprop learning is to determine the new update-values

$$\Delta_{ij}^k = \begin{cases} \eta^+ \Delta_{ij}^k & \text{if } \dfrac{\partial E(w^{k-1})}{\partial w_{ij}} \dfrac{\partial E(w^k)}{\partial w_{ij}} > 0, \\ \eta^- \Delta_{ij}^k & \text{if } \dfrac{\partial E(w^{k-1})}{\partial w_{ij}} \dfrac{\partial E(w^k)}{\partial w_{ij}} < 0, \\ \Delta_{ij}^{k-1} & \text{otherwise,} \end{cases}$$

where $0 < \eta^- < 1 < \eta^+$.

Thus, every time the partial derivative of the corresponding weight w_{ij} changes its sign between two consecutive iterations, which is considered an indication that the last update was too large and the algorithm has jumped over a local minimum, the update-value Δ_{ij}^k is decreased by the factor η^-. To enforce returning to the region of the local minimum, w_{ij} is not updated at this iteration and $\partial E(w^k)/\partial w_{ij} = 0$. If the derivative retains its sign, the update value is slightly increased in order to accelerate convergence in shallow regions of the error surface.

Rprop's assumption that a change of sign of the partial derivative implies a jump over a local minimum along the direction of the weight w_{ij} does not take into account whether the weight update has caused an increase or decrease of the batch error. Note that a change of sign of the partial derivative could also imply a jump over a local maximum. Also the Rprop strategy of slightly increasing the update value when the derivatives retain their signs may accelerate convergence in shallow regions when the derivatives are negative but may be inefficient when the two derivatives are positive. In the case of positive derivatives, large weight updates may lead the weight trajectory far away from the minimum or in regions with higher error function values. These types of problems are treated empirically by employing two heuristic parameters $\Delta_{\max}$ and $\Delta_{\min}$, which act as constraining conditions on the size of the update steps.

The total number of parameters in Rprop is five [21]: (i) the increase factor is set to $\eta^+ = 1.2$; (ii) the decrease factor is set to $\eta^- = 0.5$; (iii) the initial update-value is set to $\Delta_0 = 0.1$; (iv) the maximum step, which is used in order to prevent the weights from becoming too large, is $\Delta_{\max} = 50$; (v) the minimum step, which is used to avoid too small weight changes, is $\Delta_{\min} = 10^{-6}$.

Recently, the *improved Rprop—IRprop* algorithm [8], which applies a backtracking strategy (i.e. it decides whether to take back a step along a weight direction or not by means of a heuristic), has been proposed. The idea of this Rprop modification is that weight updates that cause changes to the signs of the corresponding partial derivatives should be reverted only in case of error increase. Thus, this training scheme combines the local information, i.e. the sign of the partial derivative of the error with respect to a weight like Rprop, with more global information, i.e. the error value at each iteration, in order to decide for each weight individually whether or not to revert an update step [8].

3. New learning rates adaptation strategy for Rprop

In our approach, Rprop's convergence to a local minimizer is treated with principles from unconstrained minimization theory. At this point, it is important to clarify that in our context, the term global convergence is used in the same sense as in Dennis and Schnabel [4], where the authors use it "to denote a method that is designed to converge to a *local* minimizer of a nonlinear function, *from almost any starting point*" [4, p. 5]. 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 [17, p. 200], defines a globally convergent algorithm as an algorithm with iterates that converge from a remote starting point. Thus, in this context, global convergence is totally different from global optimization [23].

Our approach builds on the following assumptions from the unconstrained minimization theory: (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 neighborhood $\mathscr{N}$ of the level set $\mathscr{L} = \{x : f(x) \leqslant f(x^0)\}$, (iii) the gradient of f denoted by g is Lipschitz continuous on $\mathbb{R}^n$ that is for any two points x and $y \in \mathbb{R}^n$, g satisfies Lipschitz condition $\|g(x) - g(y)\| \leqslant L\|x - y\|$, $\forall x, y, \in \mathscr{N}$, where $L > 0$ denotes the Lipschitz constant, and x^0 is the starting point of the following iterative scheme

$$x^{k+1} = x^k + \tau^k d^k, \quad k = 0, 1, \dots . \tag{2}$$

Convergence of the general iterative scheme (2), in which d^k is the search direction and $\tau^k > 0$ is a step length, requires that the adopted search direction d^k satisfies condition $g(x^k)^\top d^k < 0$, which guarantees that d^k is a descent direction of $f(x)$ at x^k. The step length in (2) can be defined by means of a number of rules, such as Armijo's rule [4,25], Goldstein's rule [4], or Wolfe's rule [27,28], and guarantees the convergence in certain cases. For example, when the step length is obtained through Wolfe's rule [27,28]

$$f(x^k + \tau^k d^k) - f(x^k) \leqslant \sigma_1 \tau^k g(x^k)^\top d^k, \tag{3}$$

$$g(x^k + \tau^k d^k)^\top d^k \geqslant \sigma_2 g(x^k)^\top d^k, \tag{4}$$

where $g(x)$ is the gradient of f at x, and $0 < \sigma_1 < \sigma_2 < 1$, then a theorem by Wolfe [27,28] is used to obtain convergence results. Moreover, Wolfe's Theorem [4,17] suggests that if the cosine of the angle between the search direction d^k and $-g(x^k)$ is positive, then $\lim_{k\to\infty} g(x^k) = 0$, which means that the sequence of gradients converges to zero. For an iterative scheme (2), $\lim_{k\to\infty} g(x^k) = 0$ is the best type of global convergence result that can be obtained (see [17] for a detailed discussion). Evidently, no guarantee is provided that (2) will converge to a global minimizer, x^*, but only that it possesses the global convergence property [4,17] to a local minimizer.

In batch training, E is bounded from below, since $E(w) \geqslant 0$. For a given training set and network architecture, if 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.

Theorem 1. *Suppose that assumptions* (i)–(iii) *are fulfilled, then for any* $w^0 \in \mathbb{R}^n$ *and any sequence* $\{w^k\}_{k=0}^{\infty}$ *generated by the Rprop scheme*

$$w^{k+1} = w^k - \tau^k \operatorname{diag}\{\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k\} \operatorname{sign}(g(w^k)), \quad k = 0, 1, \ldots, \tag{5}$$

where $\operatorname{sign}(g(w^k))$ *denotes the column vector of the signs of the components of* $g(w^k) = (g_1(w^k), g_2(w^k), \ldots, g_n(w^k))$, $\tau^k > 0$ *satisfying Wolfe's conditions,* $\eta_m^k (m = 1, 2, \ldots, i-1, i+1, \ldots, n)$ *are small positive real numbers generated by Rprop's learning rates schedule:*

$$\textit{if } (g_m(w^{k-1}) \cdot g_m(w^k) > 0) \quad \textit{then } \eta_m^k = \min(\eta_m^{k-1} \cdot \eta^+, \Delta_{\max}), \tag{6}$$

$$\textit{if } (g_m(w^{k-1}) \cdot g_m(w^k) < 0) \quad \textit{then } \eta_m^k = \max(\eta_m^{k-1} \cdot \eta^-, \Delta_{\min}), \tag{7}$$

$$\textit{if } (g_m(w^{k-1}) \cdot g_m(w^k) = 0) \quad \textit{then } \eta_m^k = \eta_m^{k-1}, \tag{8}$$

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

$$\eta_i^k = -\frac{\sum_{\substack{j=1 \\ j \neq i}}^{n} \eta_j^k g_j(w^k) + \delta}{g_i(w^k)}, \quad 0 < \delta \ll \infty, \quad g_i(w^k) \neq 0, \tag{9}$$

it holds that $\lim_{k \to \infty} g(w^k) = 0$.

Proof. Evidently, E is bounded below on $\mathbb{R}^n$. The sequence $\{w^k\}_{k=0}^{\infty}$ generated by the iterative scheme (5) follows the direction

$$d^k = -\operatorname{diag}\{\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k\} \operatorname{sign}(g(w^k)),$$

which is a descent direction if η_m^k, $m = 1, 2, \ldots, i-1, i+1, \ldots, n$ are positive real numbers derived from Relations (6)–(8), and η_i^k is given by Relation (9), since $g(w^k)^\top d^k < 0$. Following the proof of [26, 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 (3)–(4) [4,17]. Moreover, Wolfe's Theorem [4,17] suggests that if the cosine of the angle between the descent direction d^k and the $-g(w^k)$ is positive then $\lim_{k \to \infty} g(w^k) = 0$. In our case, indeed $\cos \theta_k = (-g(w^k)^\top d^k)/(\|g(w^k)\| \|d^k\|) > 0$. Thus, the theorem is proved. $\square$

The modified Rprop, named *GRprop*, is implemented through relations (5)–(9). It is worth mentioning that relation (9) can be applied cyclically over the local learning rates, or randomly. In some cases, it may be better to select the ith coordinate with the absolute smallest no zero $g_i(w^k)$ value, as this may result in a larger η_i^k value. Another choice, which has been used in all experiments reported in this paper, is to replace each time the smallest learning rate value that yields from the Rprop's schedule with an η_i^k value calculated from relation (9).

The role of δ in relation (9) 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. The length of the minimization step can be regulated through τ^k tuning to satisfy (3)–(4). Checking (4) at each iteration requires additional gradient evaluations; thus, in practice (4) can be enforced simply by placing the lower bound on the acceptable values of the learning rates [12, p. 1772], i.e. $\Delta_{\min}$.

4. Empirical study

A simple problem is used first to visualize the behavior of the GRprop and compare it with the original method. It is a single node with two weights and logistic activation function. Fig. 1 (top row) shows that under the same initial weights and heuristic values, [21], GRprop locates the feasible minimum at the center of the contour plot successfully (Fig. 1, left), while Rprop oscillates around the neighborhood of the minimizer (Fig. 1, right). Fig. 1 (second row) shows how GRprop locates the minimizer successfully, whilst Rprop's trajectory leads to a point with error value higher than the minimizer.

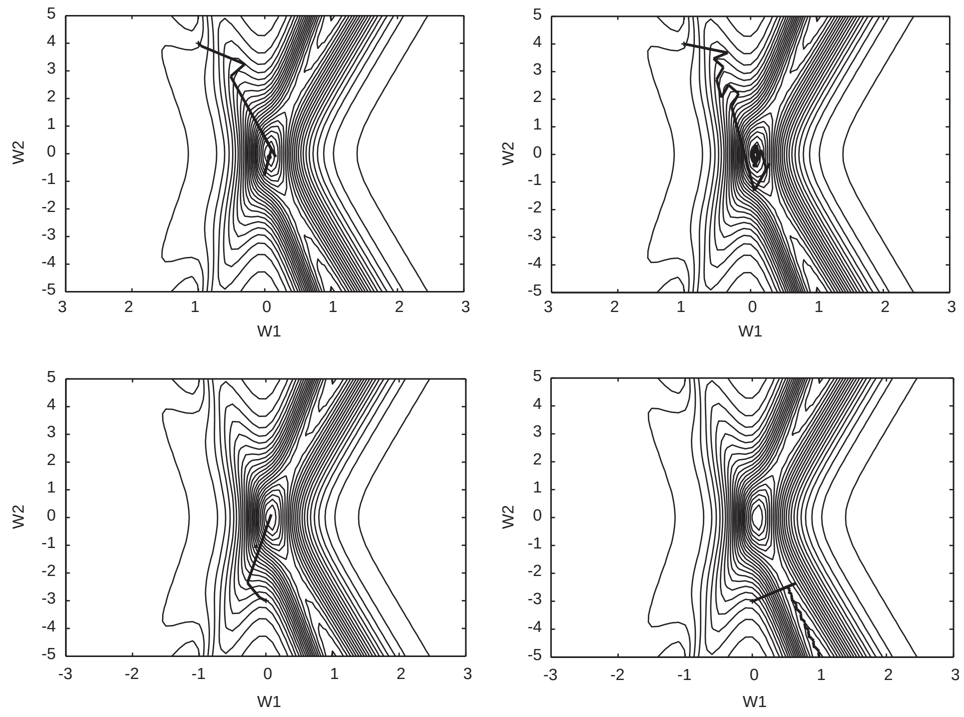

Fig. 1. Weight trajectories of GRprop (left) and Rprop (right).

Next, we evaluate the performance of the new method in six benchmarks and compare it with the original Rprop [21] and the IRprop [8]. We have used well-studied problems from the UCI Repository of Machine Learning Databases [14], as well as problems studied extensively by other researchers in an attempt to reduce as much as possible biases introduced by the size of the weights space. We decided not to enhance the algorithms tested with add-on techniques for improving the classification success in the testing phase (i.e. generalization ability of the trained neural network) as this would require introducing, and fine tuning or optimizing additional heuristics depending on the learning task.

Below, we report results from 100 independent trials for six neural network classification problems, namely cancer1, diabetes1, thyroid1, genes2, *Escherichia coli*, yeast. These 100 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.

The data sets for the cancer1, diabetes1, thyroid1, and genes2 problems were used as supplied on the PROBEN1 website. PROBEN1 provides explicit instructions for creating training and testing sets and choosing network architectures for many problems [20]. The data sets for the *E. coli* and yeast problems were used as supplied on the UCI repository and the sets for training and testing were created following guidelines published by Horton [7]. There are no standard neural architectures for these two problems so we have done our own preliminary experiments as will be described in the problem subsections.

In all experiments, the parameters have been set as follows: $\eta^+ = 1.2$; $\eta^- = 0.5$; $\Delta_{ij}^0 = \eta^0 = 0.1$; $\Delta_{\max} = 50$ [21]. 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.

4.1. The genes2 problem

The first benchmark is known as the *genes* problem. It is a binary problem. The data set consists of 1588 patterns. We used a 120-4-2-3 nodes network, and the testing and training data were created as suggested in PROBEN1 [20]. The error goal was set at 10^{-5} in an attempt to explore the effectiveness of the algorithms in reaching minimizers with high degree of accuracy.

Table 1
Comparison of algorithms performance in the genes problem for the runs which converged

Genes				
Algorithm	Epochs	Time	Generalization	Convergence
Rprop	2590	38.6 ± 20.3	99.1	97
IRprop	2574	37.7 ± 19.9	99.1	97
GRprop	2384	36.8 ± 15.3	100	100

Table 1 shows the performance of each algorithm in terms of average number of epochs (Epochs), average training time (time, in seconds) to reach the error goal $\pm$ the corresponding value of standard deviation, average generalization (generalization, percentage of correctly classified test patterns) and convergence success out of the 100 runs (convergence, percentage). For example, GRprop-trained networks always generalize slightly better than other networks: GRprop achieved 100% average generalization in the 100 runs which converged, while Rprop and IRprop achieved on average 99.1% generalization in the 97 runs that they converged. Fig. 2 (left side), shows how GRprop converges to a feasible solution ($E < 10^{-5}$), while Rprop to a minimizer with higher error value.

Table 2 presents comparative results in terms of training speed (in seconds) and generalization for the 100 runs. Table 2 highlights the number of runs for which an algorithm is better than the other methods. It also shows the average training time and the corresponding standard deviation for each algorithm calculated over the 100 runs; it is clear that the Rprop and IRprop's values show an increase compared with the corresponding values of Table 1 since all runs count. The GRprop outperforms Rprop and IRprop in 53 and 59 times, respectively, while Rprop is faster than GRprop and IRprop in 47 and 45 runs.

Finally, this table shows how many times an algorithm of the first column generalizes better than the other algorithms: the GRprop achieves better generalization than Rprop and IRprop 3 times, while IRprop has equal or lower generalization than Rprop and GRprop.

4.2. *The thyroid problem*

The second task is to decide whether the patient's thyroid has over function, normal function, or under function. We have used the *thyroid1* dataset (3600 patterns), a network with 21-4-3 nodes, and the error goal was set at 0.0036, as suggested in [23].

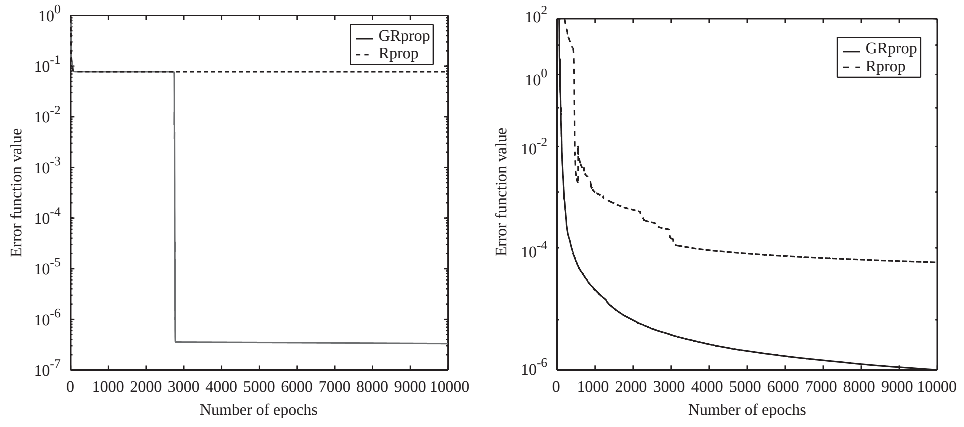

Fig. 2. GRprop and Rprop learning curves: genes (left) and thyroid (right).

Table 2
Number of times, out of 100 runs, each algorithm performs better than the other methods in the genes problem with respect to training speed and generalization

Genes	Times faster algorithm				Times better generalization		
Algorithm	Rprop	IRprop	GRprop	Time	Rprop	IRprop	GRprop
Rprop	—	45	47	41.9 ± 27.4	—	0	0
IRprop	55	—	41	41.3 ± 27.1	0	—	0
GRprop	53	59	—	36.8 ± 15.3	3	3	—

Table 3
Comparison of algorithms performance in the thyroid problem for the runs which converged

Thyroid				
Algorithm	Epochs	Time	Generalization	Convergence
Rprop	615	19.89 ± 12.4	98.12	87
IRprop	605	19.58 ± 12.1	98.12	87
GRprop	360	11.80 ± 2.5	98.23	100

Results are given in Table 3. GRprop outperforms the other algorithms particularly in training speed. Fig. 2 (right side) illustrates a case where GRprop converges to a minimizer while Rprop gets stuck at a local minimizer with higher error value.

The results in Table 4 show that average training speed and the corresponding standard deviation of the Rprop and IRprop have increased when the results of all the runs (100 cases) have been taken into account. This happens because the two algorithms converge only 87 times out of the 100 runs (see Table 3). Nevertheless, the performance of the GRprop appears significantly better: GRprop converges faster in 79 and 78 runs compared to the Rprop and the IRprop, respectively. Moreover, the value of the deviation of the new algorithm is significantly lower than the standard deviation of the other two methods (Table 4). In 27 runs, Rprop and IRprop achieve better generalization than GRprop, which outperforms in 58 runs.

4.3. The cancer1 problem

This is a breast cancer diagnosis problem based on 9 inputs describing a tumour as benign or malignant. The data set consists of 350 patterns. We have used a feed-forward neural network with 9-4-2-2 nodes as suggested in the PROBEN1 benchmark collection and in [8]. The error goal in training was $E<0.02$ to harmonize with the training errors obtained in [8]. The results for this pattern classification problem are summarized in Table 5. The new algorithm performs significantly better than the other two methods. The differences between IRprop and Rprop seems to be unimportant. Table 6 presents the number of times each

Table 4
Number of times, out of 100 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	IRprop	GRprop	Time	Rprop	IRprop	GRprop
Rprop	—	26	21	25.7 ± 18.9	—	0	27
IRprop	74	—	22	25.3 ± 18.6	0	—	27
GRprop	79	78	—	11.8 ± 2.5	58	58	—

Table 5
Comparison of algorithms performance in the cancer problem for the runs which converged

Cancer				
Algorithm	Epochs	Time	Generalization	Convergence
Rprop	187	1.3 ± 0.65	97.4	95
IRprop	185	1.2 ± 0.60	97.2	95
GRprop	127	0.9 ± 0.18	97.5	100

Table 6
Number of times, out of 100 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	IRprop	GRprop	Time	Rprop	IRprop	GRprop
Rprop	—	32	23	1.5 ± 1.73	—	1	23
IRprop	67	—	32	1.4 ± 1.67	0	—	23
GRprop	68	68	—	0.9 ± 0.18	25	26	—

algorithm outperforms the other methods in terms of training speed and generalization within 100 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.

4.4. 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 8 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

Table 7
Comparison of algorithms performance in the diabetes problem for the runs which converged

Diabetes				
Algorithm	Epochs	Time	Generalization	Convergence
Rprop	414	2.3 ± 2.3	75.5	97
IRprop	410	2.1 ± 2.1	75.4	97
GRprop	302	1.7 ± 0.6	75.7	100

Table 8
Number of times, out of 100 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	IRprop	GRprop	Time	Rprop	IRprop	GRprop
Rprop	—	49	38	4.8 ± 11.2	—	0	50
IRprop	48	—	45	4.7 ± 11.1	0	—	50
GRprop	51	50	—	1.7 ± 0.68	42	42	—

others [8]. The error goal in this case was set at 0.14 to conform to the training error obtained in [8].

Table 7 summarizes the performance of the tested algorithms. The increased training speed (Time, in seconds) does not affect the generalization performance of the new method. Table 8 gives an analytic view of the comparative results in the 100 trials. It is worth noting the standard deviation value of the GRprop is significantly less than the corresponding Rprop and IRprop values, which means the GRprop performance is closer to the average value.

4.5. The E. coli problem

This problem concerns the classification of protein localization patterns into eight classes. A drastically imbalanced data set of 336 patterns is used, where there are classes with 140 patterns and others with only 2 and 5 patterns.

E. coli, being a prokaryotic Gram-negative bacterium, is an important component of the biosphere. It colonizes the lower gut of animals and survives, when realize to the natural environment, allowing widespread to new hosts, as it is a faculties anaerobe [3,10]. 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 proses (11.5%) [9].

For this problem, we have used seven different attributes as in [7,15,16]. The first attribute is generated by applying McGeoch's method for signal sequence recognition. The second one is a result of the von Heijne's method for signal sequence recognition and the third one is the von Heijne's signal peptidase II consensus sequence score. The fourth attribute represents the presence of charge on N-terminus of predicted lipoproteins. The fifth attribute is the score of discriminant analysis of the amino-acid content of outer membrane and periplasmic proteins, and the sixth one is the score of the ALOM membrane spanning region prediction program. The last attribute gives the score of ALOM program after excluding putative cleavable signal regions from the sequence.

In particular, protein patterns in the *E. coli* data set are organized as follows: 143 patterns of cytoplasm (cp), 77 of inner membrane without signal sequence (im), 52 of periplasm (pp), 35 of inner membrane with uncleavable signal sequence (imU), 20 of outer membrane without lipoprotein (om), 5 of outer membrane with lipoprotein (omL), 2 of inner membrane with lipoprotein (imL) and 2 patterns of inner membrane with cleavable signal sequence (imS).

The literature suggests no standard architecture for the *E. coli* problem. To get an understanding of the requirements of problem, we conducted a set of preliminary experiments to find the most suitable FNN architecture in terms of training speed. In these experiments, the neural networks were tested using 4-fold cross validation, as this approach has been used before in the literature for training probabilistic and nearest neighbor classifiers in this problem [7]. We trained several networks with one and two hidden layers using the Rprop algorithm. In particular, we tried various combinations of hidden nodes, i.e. 8, 12, 14, 16, 24, 32, 64, 120 hidden nodes. Each FNN architecture was trained 10 times with different initial weights. The best available architecture found was a 7-16-8 FNN. Rprop-trained FNNs of this architecture achieved better generalization than the best results reported in Ref. [7], when the training error goal was $E < 0.02$ [1].

Results from 100 runs for three algorithms using the same architecture are given in Table 9. A detailed account of the algorithms' performance is exhibited in Table 10. The new learning scheme is faster than Rprop and IRprop 54 and 95 times, respectively. It is clear that there are no significant differences in the generalization performance of the three algorithms.

Table 9
Comparison of algorithms performance in the *E. coli* problem for the runs which converged

E. coli				
Algorithm	Epochs	Time	Generalization	Convergence
Rprop	130	1.25 ± 0.29	89.9	100
IRprop	128	1.15 ± 0.28	90.0	100
GRprop	120	1.0 ± 0.21	90.1	100

Table 10
Number of times, out of 100 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	IRprop	GRprop	Rprop	IRprop	GRprop
Rprop	—	65	46	—	0	34
IRprop	31	—	4	0	—	33
GRprop	54	95	—	33	33	—

4.6. The yeast problem

Saccharomyces cerevisiae (yeast) is the simplest Eukaryotic organism. Yeast, as a more complicated form of life than *E. coli*, possesses different types of proteins related to the cytoskeletal structure of the cell, the nucleus organization, membrane transporters and metabolic-related proteins (as mitochondrial proteins). The yeast membrane transporter proteins are of major importance as they are responsible for nutrient uptake, drug resistance, salt tolerance, control of cell volume, efflux of undesirable metabolites and sensing of extracellular nutrients [24].

In the yeast problem a pattern consists of 8 attributes [7,15,16]. The first attribute is the result of the McGeoch's method for signal sequence recognition. Applying the von Heijne's method for signal sequence recognition gives the second attribute. The score of the ALOM membrane spanning region prediction program is the third attribute and the score of discriminant analysis of the amino-acid content of the N-terminal region (20 residues long) of mitochondrial and non-mitochondrial proteins represents the fourth attribute. The fifth attribute is binary and indicates the presence of "HDEL" substring (thought to act as a signal for retention in the endoplasmic reticulum lumen). The value of the peroxisomal targeting signal in the C-terminus is another attribute. The last two attributes of the yeast data correspond to the score of discriminant analysis of the amino-acid content of vacuolar and extracellular proteins, and the score of discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins.

The data set is drastically imbalanced, and is organized as follows: there are 463 patterns of cytoplasm (cyt), 429 of nucleus (nuc), 244 of mitochondria (mit), 163 of membrane protein without N-terminal signal (me3), 51 of membrane protein with uncleavable signal (me2), 44 of membrane protein with cleavable signal (me1), 35 of extracellular (exc), 30 of vacuole (vac), 20 of peroxisome (pox) and 5 patterns of endoplasmic reticulum (erl).

We worked in a similar way to the *E. coli* problem to find a suitable architecture, as there is no specific advice in the literature. We conducted a set of preliminary experiments training several networks with one and two hidden layers using the Rprop algorithm. Combinations of 8, 12, 14, 16, 24, 32, 64, 120 hidden nodes were tried. Each FNN architecture was trained 10 times with different initial weights. The best available architecture found was an 8-16-10 FNN, and this was used in the rest

Table 11
Comparison of algorithms performance in the yeast problem for the runs which converged

Yeast				
Algorithm	Epochs	Time	Generalization	Convergence
Rprop	805	28.4 ± 6.8	61.9	100
IRprop	799	27.7 ± 6.2	61.8	100
GRprop	750	25.1 ± 4.0	62.2	100

Table 12
Number of times, out of 100 runs, each algorithm performs better than the other methods in the yeast problem with respect to training speed and generalization

Yeast	Times faster algorithm			Times better generalization		
Algorithm	Rprop	IRprop	GRprop	Rprop	IRprop	GRprop
Rprop	—	34	31	—	0	43
IRprop	65	—	35	0	—	43
GRprop	69	65	—	52	52	—

of the experiments. In these experiments 10-fold cross validation was applied, as this approach has been reported to produce higher generalization performance. Rprop-trained FNNs that reached training errors $E < 0.05$ produced better generalization on the average [1] than the best available classifier [7].

Comparative performance results are presented in Table 11. Table 12 shows the results in detail.

5. Discussion

It is widely accepted that the Rprop algorithm is one of the best performing sign-based learning algorithms for neural networks with arbitrary topology. The new globally convergent batch training algorithm constitutes an efficient improvement of the Rprop algorithm that is built on a theoretical basis. The GRprop has exhibited significantly better convergence speed than Rprop and IRprop, in all cases tested. Table 13 gives the summary of the results in terms of GRprop's percentage of improvement in training speed (in seconds) over Rprop and IRprop results.

The dimensionality of the search space seems to influence the performance of the tested methods, although additional experiments are needed to extract definite conclusions. The GRprop achieves improved results when the dimensionality of the search space is not high (e.g. cancer, diabetes, and thyroid problems). When the dimensionality ranges from medium to high there is still improvement as in the yeast and genes problem (where the number of the weights is significantly larger than the

Table 13
Summary of GRprop results in terms of learning speed improvement over Rprop and IRprop

Problem	Dimensionality	Rprop (%)	IRprop (%)
Genes2	503	+4.7	+2.4
Thyroid	103	+40.7	+39.5
Cancer	56	+29.7	+28
Diabetes	30	+26.1	+23.5
E. coli	264	+20	+13.0
Yeast	314	+11.6	+9.4

other problems). Moreover, the results of Table 13 are getting additional value when we take into account that the GRprop has also exhibited better convergence success in the majority of the 100 runs for all problems.

It is worth mentioning the high generalization performance of the Rprop-family in the bioinformatics problems, namely *E. coli* and yeast. The three members of the family produced on average results which are better than the ones reported in the literature [7]. Also, it is important to mention that the GRprop converges faster in both problems (see Tables 10 and 12). Finally, the standard deviation values in Tables 10 and 12 point out that GRprop produces more consistent behavior than the other algorithms.

6. Concluding remarks

In this paper, we have introduced a globally convergent modification of the Rprop algorithm, named GRprop. We have provided a theoretical justification for its development, and reported comparative results in six benchmark problems. In our tests, GRprop has exhibited better convergence speed and stability than Rprop and IRprop. There is of course the need to conduct further research into the performance of GRprop in other pattern recognition problems and test exhaustively the method in other classes of problems to fully explore its advantages and identify possible limitations. Nevertheless, it is important to highlight the fact that GRprop constitutes an efficient improvement of the original Rprop that builds on a theoretical basis. This makes GRprop a potentially useful component of a global optimization algorithms. That is an approach that we intend to investigate in the near future.

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, pp. 430–439.

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

[3] F.R. Blattner, G. Plunkett, C.A. Bloch, N.T. Perna, V. Burland, M. Riley, J. Collado-Vides, J.D. Glasner, C.K. Rode, G.F. Mayhew, J. Gregor, N.W. Davis, H.A. Kirkpatrick, M.A. Goeden, D.J. Rose, B. Mau, Y. Shao, The complete genome sequence of *Escherichia coli* K-12, Science 277 (5331) (1997) 1453–1462.

[4] J.E. Dennis, R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, SIAM, Philadelphia, 1996.

[5] P.E. Gill, W. Murray, M.H. Wright, Practical Optimization, Academic Press, New York, 1981.

[6] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the Marquardt algorithm, IEEE Trans. Neural Networks 5 (1994) 989–993.

[7] P. Horton, K. Nakai, Better prediction of protein cellular localization sites with the *k* nearest neighbors classifier, in: Proceedings of Intelligent Systems in Molecular Biology, 1997, pp. 368–383.

[8] C. Igel, M. Husken, Empirical evaluation of the improved Rprop learning algorithms, Neurocomputing 50 (2003) 105–123.

[9] P. Liang, B. Labedan, M. Riley, Physiological genomics of *Escherichia coli* protein families, Physiol. Genomics 9 (1) (2002) 15–26.

[10] H. Lodish, A. Berk, S.L. Zipursky, P. Matsudaira, D. Baltimore, J. Darnell, Molecular Cell Biology, fifth ed., Freeman, New York, 2003.

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

[12] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Improving the convergence of the back-propagation algorithm using learning rate adaptation methods, Neural Comput. 11 (1999) 1769–1796.

[13] M.F. Moller, A scaled conjugated gradient algorithm for fast supervised learning, Neural Networks 6 (1993) 525–533.

[14] P.M. Murphy, D.W. Aha, UCI Repository of machine learning databases, University of California, Irvine, CA, 1994. http://www.ics.uci.edu/mlearn/MLRepository.html.

[15] K. Nakai, M. Kanehisa, Expert system for predicting protein localization sites in gram-negative bacteria, PROTEINS: Structure, Function, and Genetics 11 (1991) 95–110.

[16] K. Nakai, M. Kanehisa, A knowledge base for predicting protein localization sites in eukaryotic cells, Genomics 14 (1992) 897–911.

[17] J. Nocedal, Theory of algorithms for unconstrained optimization, Acta Numer. (1992) 199–242.

[18] L.M. Patnaik, K. Rajan, Target detection through image processing and resilient propagation algorithms, Neurocomputing 35 (1–4) (2000) 123–135.

[19] M. Pfister, R. Rojas, Speeding-up backpropagation—a comparison of orthogonal techniques, in: Proceedings of the Joint Conference on Neural Networks, Nagoya, 1993, pp. 517–523.

[20] L. Prechelt, PROBEN1-A set of benchmarks and benchmarking rules for neural network training algorithms, Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.

[21] 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, 1993, pp. 586–591.

[22] D.E. Rumelhart, J.L. McClellend, in: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge, 1986, pp. 318–362.

[23] 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.

[24] D. Van Belle, B. Andre, A genomic view of yeast membrane transporters, Curr. Opin. Cell Biol. 13 (4) (2001) 389–398.

[25] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos, G.D. Magoulas, A class of gradient unconstrained minimization algorithms with adaptive stepsize, J. Comput. Appl. Math. 114 (2) (2000) 367–386.

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

[27] P. Wolfe, Convergence conditions for ascent methods, SIAM Rev. 11 (1969) 226–235.
[28] P. Wolfe, Convergence conditions for ascent methods. II: some corrections, SIAM Rev. 13 (1971) 185–188.

Aristoklis D. Anastasiadis received his first degree from The University of Patras, Greece, in Electrical and Computer Engineering (BEng/MEng). Since 2002, he had been working towards a Ph.D degree in artificial neural network training algorithms at the School of Computer Science and Information Systems, Birkbeck College, University of London, UK. His research interests include feedforward neural networks and evolutionary algorithms, with emphasis to applications on biological data and classification problems.

George D. Magoulas is Reader in Computer Science in the School of Computer Science and Information Systems, Birkbeck College, University of London, UK. He was educated at the University of Patras, Greece, in Electrical and Computer Engineering (BEng/MEng, PhD). He held Lecturer and Senior Lecturer posts at the Department of Information Systems and Computing, Brunel University, UK. His research is focused on learning and evolution algorithms with applications to intelligent adaptive systems. His work is currently funded by the E.P.S.R.C. and the A.H.R.B, UK. Dr. Magoulas is a member of the IEEE, the User Modeling Inc., the Technical Chamber of Greece, and the Hellenic Artificial Intelligence Society.

Professor Michael N. Vrahatis is with the Department of Mathematics at University of Patras, Greece. He received the Diploma and Ph.D. degrees in Mathematics from the University of Patras, in 1978 and 1982, respectively. He was a visiting research fellow at the Department of Mathematics, Cornell University (1987–88) and a visiting professor to the INFN (Istituto Nazionale di Fisica Nucleare), Bologna, Italy, (1992, 1994 and 1998); the Department of Computer Science, Katholieke Universiteit Leuven, Belgium, (1999); the Department of Ocean Engineering, Design Laboratory, MIT, Cambridge MA, USA, (2000), and the Collaborative Research Center "Computational Intelligence" (SFB 531) at the Department of Computer Science, University of Dortmund, Germany (2001). He was a visiting researcher at CERN (European Organization of Nuclear Research), Geneva, Switzerland, (1992), and at INRIA (Institut National de Recherche en Informatique et en Automatique), France (1998, 2003 and 2004). He is the author of more than 200 publications in his research areas, including optimization, neural networks, evolutionary algorithms and artificial intelligence. He has been principal investigator of several research grants from the European Union, the Hellenic Ministry of Education and Religious Affairs and the Hellenic Ministry of Industry, Energy and Technology. He is among the founders of the "University of Patras Artificial Intelligence Research Center" (UPAIRC), established in 1997, where currently he serves as director.