Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition Letters 26 (2005) 1926–1936

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Sign-based learning schemes for pattern classification

A.D. Anastasiadis [a,b,*], G.D. Magoulas [b], M.N. Vrahatis [c]

[a] School of Computer Science and Information Systems, Birkbeck College, University of London, London Knowledge Lab, 23–29 Emerald Street, London WC1N 3QS, United Kingdom

[b] School of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London WC1E 7HX, United Kingdom

[c] Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece

Available online 3 May 2005

Abstract

This paper introduces a new class of sign-based training algorithms for neural networks that combine the sign-based updates of the Rprop algorithm with the composite nonlinear Jacobi method. The theoretical foundations of the class are described and a heuristic Rprop-based Jacobi algorithm is empirically investigated through simulation experiments in benchmark pattern classification problems. Numerical evidence shows that this new modification of the Rprop algorithm exhibits improved learning speed in all cases tested, and compares favorably against the Rprop and a recently proposed modification, the improved Rprop.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Supervised learning; Nonlinear iterative methods; Nonlinear Jacobi; Subminimization; Feed-forward neural networks; Rprop; Pattern classification

1. Introduction

Gradient descent is the most widely used class of algorithms for supervised learning of neural networks. The most popular training algorithm of this category is the batch backpropagation (BP) (Rumelhart and McClellend, 1986). This is a first-order method that minimizes the error function by updating the weights using the steepest descent method (Battiti, 1992):

$$w^{(t+1)} = w^{(t)} - \eta \nabla E(t), \tag{1}$$

where E is the batch error measure defined as the sum of squared differences error function (SSE) over the entire training set; $\nabla E(t)$ is the vector of partial derivatives with respect to all weights.

[*] Corresponding author. Address: School of Computer Science and Information Systems, Birkbeck College, University of London, London Knowledge Lab, 23–29 Emerald Street, London WC1N 3QS, United Kingdom.
E-mail address: aris@dcs.bbk.ac.uk (A.D. Anastasiadis).

0167-8655/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2005.03.013

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.

One problem inherent with gradient descent methods relates to convergence to local minima. These techniques use only gradient information, e.g. the partial derivative of the error with respect to the weights, to perform adaptation. While some local minima can provide acceptable solutions, they often result in poor performance. This problem can be overcome through the use of global optimization. Various algorithms of this category have been employed, including simulated annealing (SA) (Fang and Li, 1990), evolutionary methods (Fogel et al., 1990), random methods, and deterministic searches (Tang and Koehler, 1994). Global optimization, however, is considered computationally expensive, which could be a significant problem particularly for large networks (Treadgold and Gedeon, 1998).

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 (Magoulas et al., 1997, 1999). This is done by controlling the weight update of each weight in order to minimize oscillations and 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 parameters, is the Resilient backpropagation (Rprop) algorithm introduced by Riedmiller and Braun (Riedmiller and Braun, 1993; Riedmiller, 1994). Recently a modification of the Rprop, the so-called improved Rprop (iRprop) has been proposed. Empirical evaluations of iRprop gave good results, showing that iRprop outperforms in several cases the quickprop and conjugate gradient algorithms (Igel and Husken, 2003; Husken and Stagge, 2003).

In this paper, we introduce a new class of sign-based algorithms that are based on nonlinear iterative methods. These algorithms combine the computationally cheap information of the sign of the gradient along each weight direction, like the Rprop algorithm does, with the "global" information of the overall batch error, in order to improve the learning speed. This paper is organized as follows. First, we give a brief outline of the Rprop algorithm and discuss its parameters. Next, the new class is described and one of its algorithms is presented. Then we conduct an empirical evaluation of the new algorithm by comparing it with the classic Rprop, the recently proposed modification iRprop, and with two second-order training algorithms, namely the BFGS quasi-Newton method (BFGS) and the scaled conjugate gradient (SCG) algorithm. Finally our results are discussed and conclusions are drawn.

2. Sign-based training with the Rprop algorithm

The basic principle of Rprop is to eliminate the harmful influence of the size of the partial derivative on the weight step. As a consequence, only the sign of the derivative is considered to indicate the direction of the weight update. The size of the weight change is exclusively determined by a weight-specific "update-value"

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

where $\partial E(t)/\partial w_{ij}$ denotes the partial derivative of E with respect to w_{ij} where E is summed over all patterns of the training set. The second step of Rprop learning is to determine the new update-values.

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

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

Thus every time the partial derivative of E with respect to weight $w_{ij}^{(t)}$ changes its sign, which indicates that the last update was too big and the algorithm has jumped over the local minimum, the update-value $\Delta_{ij}(t)$ is decreased by the factor η^-. If the derivative retains its sign, the update-value is slightly increased in order to accelerate convergence in shallow regions. Additionally, in case of a change in sign, there should be no adaptation in the succeeding learning step. In practice this can be achieved by setting $\partial E(t)/\partial w_{ij} = 0$ in the adaptation rule. Finally the weight update and the adaptation are performed after the gradient information of the whole pattern set is computed.

Rprop is based on the assumption that a change of sign of the partial derivative implies a jump over a local minimum along the direction that corresponds to the weight $w_{ij}^{(t)}$, but does not take into account whether the weight update has caused an increase or decrease of the error. Notice that a change of sign of the partial derivative could also imply a jump over a local maximum. Also in Rprop, if the derivatives retain their signs, the update-value is slightly increased in order to accelerate convergence in shallow regions. This strategy helps to speed up convergence when the derivative is negative but may be inefficient when the two derivatives are positive, as in this case the weight updates may lead the weight trajectory far away from the minimum or in regions with higher error function values. In an attempt to alleviate these situations Rprop employs a heuristic parameter $\Delta_{\max}$, which constraints the size of the update step.

In fact the Rprop algorithm requires setting the following parameters: (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 weight step, which is used in order to prevent the weights from becoming too large, is $\Delta_{\max} = 50$, and the minimum step-size is constantly fixed to $\Delta_{\min} = 10^{-6}$ (Riedmiller and Braun, 1993; Riedmiller, 1994).

A high level description of the weight update procedure, which shows the kernel of the Rprop adaptation and learning process, is described below. The minimum/maximum operator is supposed to deliver the minimum/maximum of two numbers; the sign operator returns +1, when the argument is positive; −1, when the argument is negative and 0 otherwise, as suggested by Riedmiller and Braun (Riedmiller and Braun, 1993, Riedmiller, 1994).

```
**Rprop weight update procedure:**
  repeat
    compute the gradient vector $\nabla E(t)$
    for all weights and biases
    if $\frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} > 0$ then
      $\Delta_{ij}(t) = \min\{\Delta_{ij}(t-1) \cdot \eta^+, \Delta_{\max}\}$
      $\Delta w_{ij}^{(t)} = -\text{sign}\frac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$
      $w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$
      $\frac{\partial E(t-1)}{\partial w_{ij}} = \frac{\partial E(t)}{\partial w_{ij}}$
    else if $\frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} < 0$ then
      $\Delta_{ij}(t) = \max\{\Delta_{ij}(t-1) \cdot \eta^-, \Delta_{\min}\}$
      $\frac{\partial E(t-1)}{\partial w_{ij}} = 0$
    else if $\frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} = 0$ then
      $\Delta w_{ij}^{(t)} = -\text{sign}\frac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$
      $w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$
      $\frac{\partial E(t-1)}{\partial w_{ij}} = \frac{\partial E(t)}{\partial w_{ij}}$
    end if
  until termination criterion is met
```

where sign defines the well known triple valued sign function.

Recently, Igel and Husken (2003) proposed a modification of the Rprop, named iRprop. Their method takes under consideration the change of the error for the special case of a derivative's change of sign.

3. A class of nonlinear Jacobi methods

In this section, first-order algorithms with an adaptive learning rate for each weight are analyzed as composite *nonlinear Jacobi* methods applied to the gradient of the error function. The class of nonlinear Jacobi methods is widely used for the solution of a system of nonlinear equations:

$$F(x_1, x_2, \ldots, x_n) = \Theta^n \equiv (0, 0, \ldots, 0), \qquad (2)$$

where $F = (f_1, f_2, \ldots, f_n)$: $\mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}^n$.

Along this line, in a function minimization problem all local minima $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ of a continuous differentiable function f should satisfy the necessary conditions:

$$\nabla f(x^*) = \Theta^n. \tag{3}$$

Eq. (3) represents a set of n nonlinear equations which must be solved to obtain x^*. Therefore, one approach to the minimization of the function f is to seek the solutions of the set of Eq. (3) by including a provision to ensure that the solution found does, indeed, correspond to a local minimizer. Solving Eq. (3) is equivalent to solving the following system of equations:

$$\begin{aligned} &\partial_1 f(x_1, x_2, \ldots, x_n) = 0, \\ &\partial_2 f(x_1, x_2, \ldots, x_n) = 0, \\ &\qquad\vdots \\ &\partial_n f(x_1, x_2, \ldots, x_n) = 0, \end{aligned} \tag{4}$$

where $\partial_i f(x_1, \ldots, x_i, \ldots, x_n)$ denotes the partial derivative of f with respect to the ith coordinate.

3.1. The composite nonlinear Jacobi

The nonlinear Jacobi process applies a parallel update of the variables (Ortega and Rheinboldt, 1970). Starting from an arbitrary initial vector $x^0 \in \mathscr{D}$, one can subminimize at the kth iteration the function:

$$f(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_n^k), \tag{5}$$

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

$$\partial_i f(x_1^k, \ldots, x_{i-1}^k, \hat{x}_i, x_{i+1}^k, \ldots, x_n^k) = 0. \tag{6}$$

This is a one-dimensional subminimization because all the components of the vector x^k, except from the ith component, are kept constant. Then the ith component is updated according to the equation:

$$x_i^{k+1} = x_i^k + \tau_k(\hat{x}_i - x_i^k), \tag{7}$$

for some relaxation factor τ_k. The objective function in (5) is subminimized in parallel for all i. When exact one-dimensional submimization is applied and $\tau_k = 1$ for all k the following result is available for strictly convex functions.

Theorem 1 (Brewster and Kannan, 1984). *Suppose that the objective function f: $\mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable on a convex domain $\mathscr{D}$ and that f is a strictly convex function. Assume that there exists $\gamma \in \mathbb{R}$ such that $\mathscr{S}_\gamma = \{x \in \mathscr{D}\colon f(x) \leqslant \gamma\}$ is nonempty and compact and that $\partial_{ii}^2 f(y) \neq 0$ for $i = 1, 2, \ldots, n$ and $y \in \mathscr{S}_\gamma$, unless y is the point at which f attains its minimum, where $\partial_{ij}^2 f(y)$ denotes the h_{ij} element of the Hessian matrix of f at y, $H = [h_{ij}]$. Suppose further, that from any point $x^0 = (x_1^0, x_2^0, \ldots, x_n^0) \in \mathscr{S}_\gamma$ a sequence $\{x^k\}_{k=0}^\infty$ is generated:*

$$x_j^{k+1} = x_j^k, j \neq i_k \quad \textit{and}$$

$$x_{i_k}^{k+1} \textit{ is the solution of}$$

$$\partial_{i_k} f(x_1^k, \ldots, x_{i_k-1}^k, x_{i_k}, x_{i_k+1}^k, \ldots, x_n^k) = 0,$$

where i_k is any one of the integers $1, 2, \ldots, n$. Such a sequence $\{x^k\}_{k=0}^\infty$ is uniquely defined and converges to x^*, the unique global minimizer of f, provided that in the above iterative process every coordinate direction i is chosen an infinite number of times.

Various composite nonlinear Jacobi training algorithms can be obtained depending on the one-dimensional minimization method applied. In case an inexact one-dimensional subminimization is applied, the number of the iterations or steps of the subminimization method is related to the requested accuracy in obtaining the subminimizer approximations. Thus, significant computational effort is needed in order to find very accurate approximations of the subminimizer along each variable direction at each iteration.

Moreover, this computational effort is increased for problems with a high number of variables, as for example when training neural networks with several hundred weights. Taking into account that training neural networks involves minimizing nonconvex functions, it is not certain that this large computational effort speeds up the minimization process for nonconvex functions when far from a minimizer x^*. Similar situations can also occur in the iterative solution of nonlinear equations (Ortega and Rheinboldt, 1970).

3.2. The Jacobi-bisection method

In this section we synthesize a composite Jacobi method that is inspired from the Rprop algorithm. The method, named JRprop, combines "individual" information about the error surface, i.e. the sign of the partial derivative of the error function with respect to each one of the weights, with more "global" information, i.e. the magnitude of the network's learning error at each epoch t, in order to decide for each weight individually whether or not to revert/reduce a step.

Following the nonlinear Jacobi prescription, one-dimensional subminimization is applied along each weight direction. 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 epochs (t) and $(t-1)$ the corresponding interval brackets a local minimum simply by looking the function values $E(t-1)$, $E(t)$ and gradient values $\partial E(t-1)/\partial w_{ij}$, $\partial E(t)/\partial w_{ij}$ at the endpoints of the considered interval (see Scales, 1985, for a general discussion on the problem).

The conditions that have to be satisfied (Scales, 1985) are:

$$\begin{aligned}
&\text{(a)}\quad \frac{\partial E(V1)}{\partial w_{ij}} < 0 \quad \text{and} \quad \frac{\partial E(V2)}{\partial w_{ij}} > 0,\\
&\text{(b)}\quad \frac{\partial E(V1)}{\partial w_{ij}} < 0 \quad \text{and} \quad E(V1) < E(V2),\\
&\text{(c)}\quad \frac{\partial E(V1)}{\partial w_{ij}} > 0 \quad \text{and} \quad \frac{\partial E(V2)}{\partial w_{ij}} > 0 \quad \text{and}\\
&\qquad E(V1) > E(V2),
\end{aligned} \tag{8}$$

where $V1$ and $V2$ determine the sets of weights for which the coordinate that corresponds to the weight w_{ij} is replaced by $a_i = \min\left\{w_{ij}^{(t-1)}, w_{ij}^{(t)}\right\}$, and $b_i = \max\left\{w_{ij}^{(t-1)}, w_{ij}^{(t)}\right\}$ correspondingly. Notice that, at this instance, between two successive epochs $(t-1)$ and (t) all the other coordinates remain the same (this is because we follow the nonlinear Jacobi prescription). 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. By computing the midpoint $m_i = \frac{1}{2}(a_i + b_i)$ of the interval $[a_i, b_i]$ we take as the next interval whichever of $[a_i, m_i]$ and $[m_i, b_i]$ that still brackets a minimizer according to the criteria mentioned above.

For the case of the first condition of (8) we will consider here the bisection method which has been modified to the following version described in (Vrahatis, 1988a,b):

$$w_i^{p+1} = w_i^p + h_i \,\mathrm{sign}\, \partial_i E(w^p)/2^{p+1}, \tag{9}$$

where $p = 0, 1, \ldots$ is the number of subminimization steps and $w_i^0 = a_i$; $h_i = \mathrm{sign}\, \partial_i E(w^0)(b_i - a_i)$; w^0 determines the set of weights at the $(t-1)$ epoch while w^p is obtained by replacing the coordinate of w^0 that corresponds to the weight w_{ij} by w_i^p. Of course, the iterations (9) converge to $\hat{w}_i \in (a_i, b_i)$ if for some w_i^p, $p = 1, 2, \ldots$, the first one of the conditions (8) 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) 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 (7) within the interval $[a_i, b_i]$ to a predetermined accuracy ε is known beforehand and is given by

$$v = \lceil \log_2[(b_i - a_i)\varepsilon^{-1}] \rceil. \tag{10}$$

Moreover it has a great advantage since it is worst-case optimal, i.e. it possesses asymptotically the best possible rate of convergence in the worst-case (Sikorski, 1982, 2001). 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 relation (10) it is easy to have beforehand the number of iterations that are required for the attainment of an approximate minimizer $\hat{w}_i$ to a predetermined accuracy. Finally, it requires the algebraic signs of the values of the gradient to be computed.

3.2.1. *Theoretical approach of the JRprop*

Next, we give a theoretical result that ensures the composite Jacobi method that uses a multistep bisection method for reducing the intervals of uncertainty converges to a solution. In particular this result shows that there is a neighborhood of a minimizer of the objective function for which convergence to the local minimizer can be guaranteed. Notice that this result does not require exact one-dimensional subminimization but only an approximation of the local minimizer.

Corollary 1. *Let* $E: \mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}$ *be twice continuously differentiable in an open neighborhood* $\mathscr{S}_0 \subset \mathscr{D}$ *of a point* $w^* \in \mathscr{D}$ *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* $\mathscr{S} = \mathscr{S}(w^*, r)$ *in* $\mathscr{S}_0$ (*where* $\mathscr{S}(w^*, r)$ *denotes the open ball centered at* w^* *with radius* r), *such that any sequence* $\{w^k\}_{k=0}^\infty$ *generated by the nonlinear Jacobi process converges to* w^* *which minimizes* E.

The proof of the corollary originates from the application of Theorem 1, derived in (Vrahatis et al., 2003), on the error function. As mentioned above Corollary 1 guarantees only local convergence, and naturally imposes some conditions on the Hessian matrix. 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^*. Finding such a point is equivalent to minimizing the nonlinear function (5) by applying the nonlinear Jacobi process and employing any one-dimensional method for the subminimization process.

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

$$H(w^*) = D(w^*) - L(w^*) - L^\top(w^*). \tag{11}$$

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

$$\Phi(w^*) = D(w^*)^{-1}\left[L(w^*) + L^\top(w^*)\right], \tag{12}$$

are real and $\rho(\Phi(w^*)) < 1$ (Axelsson, 1996) (where $\rho(A)$ indicates the spectral radius of the matrix A); then there exists an open ball $\mathscr{S} = \mathscr{S}(w^*, r)$ in $\mathscr{S}_0$, such that, for any initial vector $w^0 \in \mathscr{S}$, there is a sequence $\{w^k\}_{k=0}^\infty \subset \mathscr{S}$ which satisfies the nonlinear Jacobi process such that $\lim_{k\to\infty} w^k = w^*$ (Ortega and Rheinboldt, 1970).

Remark 1. *The Property A*: Young (1954) has discovered a class of matrices described as having property A that can be partitioned into block tridiagonal form, possibly after a suitable permutation (Axelsson, 1996). An algorithmic procedure for transforming a symmetric matrix to a tridiagonal form is presented in (Stewart, 1973).

3.2.2. Implementation of the JRprop

Based on the above theoretical discussion below we will consider obtaining $\hat{w}_i$ by minimizing the function (5) with one-step of a subminimization method. In particular, we propose an Rprop-based heuristic scheme that uses one-step of the bisection method to locate an approximation of the subminimizer $\hat{w}_{ij}$ along each weight direction. As mentioned in a previous subsection, one step of the subminimization method helps to reduce the computational effort for high dimensional nonconvex functions when far from a minimizer, as it usually happens in neural network training (Vrahatis et al., 2003).

Next, we present a high level description of the proposed algorithm that implements a heuristic version of the JRprop. It is based on the idea of *function comparison methods* (Scales, 1985) taking into account $E(t-1) < E(t)$, and exploits the signs of the gradient values. The parameter q is a reduction factor that is used to update the midpoint of the considered interval; choice of q has an influence on the number of error function evaluations required to obtain an acceptable weight vector (Magoulas et al., 1999).

JRprop algorithm:

```
  repeat
    compute the gradient vector $\nabla E(t)$
    if $E(t) \leqslant E(t-1)$ then
      for all weights and biases
      if $\frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} > 0$ then
        $\Delta_{ij}(t) = \min\{\Delta_{ij}(t-1) \cdot \eta^+, \Delta_{\max}\}$
        $\Delta w_{ij}^{(t)} = -\mathrm{sign}\frac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$
        $w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$
        $\frac{\partial E(t-1)}{\partial w_{ij}} = \frac{\partial E(t)}{\partial w_{ij}}$
        $\Delta w_{ij}^{(t-1)} = \Delta w_{ij}^{(t)}$
```

```
            else if $\frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} < 0$ then
                $\Delta_{ij}(t) = \max\{\Delta_{ij}(t-1) \cdot \eta^-, \Delta_{\min}\}$
                $\frac{\partial E(t-1)}{\partial w_{ij}} = 0$
            else if $\frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} = 0$ then
                $\Delta w_{ij}^{(t)} = -\mathrm{sign}\frac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$
                $w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$
                $\frac{\partial E(t-1)}{\partial w_{ij}} = \frac{\partial E(t)}{\partial w_{ij}}$
                $\Delta w_{ij}^{(t-1)} = \Delta w_{ij}^{(t)}$
            end if
            $q = 1$
        end if
        if $E(t) > E(t-1)$ then
            $w_{ij}^{(t+1)} = w_{ij}^{(t)} + \frac{1}{2^q}\Delta w_{ij}^{(t-1)}$
            $q = q + 1$
        end if
    until termination criterion is met
```

The particular implementation of the JRprop does not take a special consideration for the third condition of (8). This condition requires special treatment as it may lead the algorithm to converge to an undesired extremum. As shown in the pseudocode description, in our heuristic JRprop this is handled by the standard Rprop. Finally, we set for all the parameters the same values as suggested by Riedmiller and Braun for the Rprop algorithm (Riedmiller and Braun, 1993; Riedmiller, 1994).

Various termination criteria can be used in the weight update procedure, e.g. meeting a predefined training error goal $E(t)$, reaching the maximum number of epochs t, or having the norm of the gradient vector close to zero.

In case the initial weights are far from the neighborhood of a local minimizer, then it is possible to equip the algorithm with a strategy for adapting the direction of search to a descent one. In this way, a decrease of the function value can be ensured at each iteration and convergence to a local minimizer of the objective function from remote initial points can be achieved as has been shown in (Magoulas et al., 2002). Here we will not consider this case.

4. Experimental study

In this section, we evaluate the performance of the heuristic JRprop in four pattern classification problems and compare it with the original Rprop (Riedmiller and Braun, 1993; Riedmiller, 1994), the improved Rprop (iRprop) proposed recently by Igel and Husken (2003), the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm (Gill et al., 1981), and the scaled conjugate gradient backpropagation (SCG) proposed by Moller (1992).

The BFGS and SCG algorithms have been applied in an attempt to use second derivative related information to accelerate the learning process as suggested in (Battiti, 1992; Moller, 1992). The BFGS quasi-Newton method is an alternative to the direct application of the second-order Newton's method, which is considered of limited applicability in neural network training, particularly when the networks have a large number of weights; the one-step BFGS is a memoryless quasi-Newton algorithm (Moller, 1992). Another approach that performs well in practical applications is the SCG method. This method has superlinear convergence rate (Moller, 1992). It is a variation of the well known conjugate gradient method that avoids the line-search per learning iteration by using a Levenberg–Marquardt approach (Fletcher, 1975) in order to scale the step-size. It is a second-order algorithm, which combines the benefits of the Conjugate methods and the Levenberg–Marquardt scheme (Moller, 1992).

We have used well-studied problems from the UCI repository of machine learning databases (Murphy and Aha, 1994), namely cancer1, diabetes1, thyroid1, genes2, as described in the PROBEN1 benchmark collection (Prechelt, 1994).

The literature suggests specific network architectures for these problems; thus we considered them appropriate for our study as we wanted to reduce, as much as possible, biases introduced by the size of the weight space. We also decided not to enhance the algorithms tested with add-on techniques for improving the classification success in the testing phase (i.e. the generalization ability of the trained neural network) as this would require introducing, and fine tuning or even optimizing additional heuristics depending on the learning task. The implementation of the algorithms has been done in Matlab 6.5.

In all experiments, we used the same parameters as suggested in (Riedmiller and Braun, 1993; Riedmiller, 1994) for the classic Rprop heuristics. Feed-forward neural networks (FNN) with sigmoid hidden and output nodes were applied for the cancer1, diabetes1, thyroid1 problems, and an FNN with tansig hidden nodes were used in the genes2 problem.

The notation I–H–O is used below to denote a network architecture with I inputs, H hidden layer nodes and O outputs nodes. All results reported are based on 100 independent trials for each problem. Moreover, the 100 random weight initializations were the same for all the learning algorithms. In all cases, the training and testing sets were created according to the guidelines of PROBEN1 (see Prechelt, 1994, for details).

To analyze the statistical significance of the results we implemented the Wilcoxon test as suggested by Snedecor and Cochran (1989). This is a nonparametric method that is considered an alternative to the paired *t*-test. This test assumes that there is information in the magnitudes of the differences between paired observations, as well as the signs. Firstly, we take the paired observations, we calculate the differences and then we rank them from smallest to largest by their absolute value. After adding all the ranks associated with positive and negative differences giving T_+ and T_- statistic respectively. Finally, the probability value associated with this statistic is found from the appropriate table. All statements refer to a significance level of 0.05, which corresponds to $Z_c < 1.95$ for the convergence speed and $Z_{cr} < 1.95$ for the testing classification success.

Convergence (learning) speed is a critical factor when it comes to decide which algorithm to use. Classification success in testing (generalization performance) is another crucial factor. Below we present experimental results for both factors, and highlight cases where the results of JRprop (in terms of convergence speed and testing classification success) are statistically significant with respect to corresponding results of the other tested methods.

4.1. The cancer1 problem

This is a breast cancer diagnosis problem based on nine inputs describing a tumor 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 relevant literature (Igel and Husken, 2003). The termination criterion was an $E < 0.02$ within 2000 epochs.

The results for this pattern classification problem are summarized in Table 1. The new algorithm performs significantly better than the other tested methods. The differences between iRprop and Rprop are not important. In Table 1, we represent the average time (“Speed”, measured in seconds), the average classification success with the testing set (“Class. Success”, measured by the percentage of testing patterns that were classified correctly), and the average convergence success in the training phase (“Conv.”, measured by the percentage of simulation runs that converged to the error goal) for an algorithm. All the results are based on those runs that algorithms meet the error goal. For each comparison we apply the Wilcoxon rank test to calculate the significance of the results of the JRprop with respect to the other methods. Statistically significant cases are marked with (+).

Judging from Table 1, we can notice that all first-order methods converge to the error goal and there is no significant difference in the testing classification success percentage. The second-order algorithms BFGS and SCG meet the error goal 72 and 88 times out of the 100 runs respectively. The SCG algorithm converges faster in terms of epochs than the Rprop and iRprop but needs more time

Table 1
Results for the cancer problem

Cancer			Average	
Algorithm	Epochs	Speed (s)	Class. success (%)	Conv. (%)
Rprop	242	1.7 (+)	97.64 (−)	100
iRprop	241	1.5 (+)	97.63 (−)	100
BFGS	632	10.7 (+)	94.00 (+)	72
SCG	198	2.2 (+)	95.99 (+)	88
JRprop	151	1.1	97.60	100

as it requires on average more time at each iteration for the computations. Finally the BFGS algorithm has the worst performance as many times during training the Hessian matrix is close to singular or is badly scaled.

The new algorithm is faster than the other methods. The percentage of improvement that the new algorithm achieved over Rprop and iRprop in terms of learning speed is 34.5% and 29.4% respectively. The results show slightly less classification success but the Wilcoxon rank test did not show any statistical significance with respect to the first-order method. Instead, it shows slightly better generalization compared to the second-order algorithms.

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 has been also suggested by others (Igel and Husken, 2003). The termination criterion is $E < 0.14$ within 2000 epochs.

In the diabetes classification problem both Rprop and iRprop behave similarly. Table 2 gives the average convergence speed and success of the tested algorithms. The table also includes the results of the Wilcoxon Rank test. The increased convergence speed does not seem to affect the classification success of the new method in testing.

4.3. The genes2 problem

It is a binary problem. The goal of this classification task is to decide, from a window of 60 DNA sequence elements (nucleotides), whether the middle is either an intron/exon boundary (a donor), or an exon/intron boundary (an acceptor), or none of these. Each nucleotide is encoded using bipolar inputs (i.e. -1 and $+1$). The data set consists of 1588 patterns. This data set was created based on the "splice junction" problem from the UCI repository. We have used an architecture with 120–4–2–3 nodes as suggested in the PROBEN1 benchmark collection. The termination criterion is $E < 10^{-5}$ within 10000 epochs.

In Table 3, the average learning time, the convergence success and the results of the Wilcoxon Rank Test are exhibited; a plus, (+), indicates cases where the JRprop outperforms another method. Thus the improvement of JRprop is statistically significant when compared to Rprop and iRprop with respect to both learning speed and classification success. Furthermore, it is important to notice that the testing classification success of JRprop is 100% in this binary problem. This is particularly important as the Wilcoxon rank test is satisfied. The overall improvement achieved by the JRprop over Rprop and iRprop is 25% and 23.8% in terms of convergence. However it is clear that the second-order algorithms achieve significantly improved learning speed in this problem with respect to the first-order

Table 2
Results for the diabetes problem

Diabetes			Average	
Algorithm	Epochs	Speed (s)	Class. success (%)	Conv. (%)
Rprop	321	2.1 (+)	75.64 (−)	95
iRprop	317	1.9 (+)	75.67 (−)	95
BFGS	360	6.2 (+)	74.68 (+)	64
SCG	375	4.3 (+)	74.64 (+)	76
JRprop	172	1.1	75.78	100

Table 3
Results for the genes problem

Genes			Average	
Algorithm	Epochs	Speed (s)	Class. success (%)	Conv. (%)
Rprop	3240	48.2 (+)	98.76 (+)	95
iRprop	3235	47.2 (+)	98.76 (+)	94
BFGS	379	14.6	96.70 (+)	92
SCG	217	7.6	96.20 (+)	90
JRprop	2698	39.9	100.00	100

methods with the SCG outperforming over all the other algorithms.

4.4. The thyroid problem

This problem is based on patient query data and patient examination data. The task is to decide whether the patient's thyroid has over function, normal function, or under function. The data set consists of 3600 patterns. We use the thyroid1, which is not a permutation of the original data, but retains the original order. We have used an architecture with 21–4–3 nodes, as suggested by Treadgold and Gedeon (1998). The termination criterion is $E < 0.0036$ within 2000 epochs.

The comparative results of the tested algorithms are given in Table 4. The table gives the average learning speed, the average classification success in testing and the result of the Wilcoxon Rank Test. In our tests, the SCG did not converge to the desired error goal within the specified number of epochs in any of the 100 runs; this is indicated with a *D* in the table.

In an attempt to explore the reasons for this behavior of the SCG, we run again the experiments using the same initial weights as before but we set the error goal to a higher value, i.e. $E < 0.01$. The SCG exhibited a 90% average percentage of convergence success and an average number of epochs of 500; in that case Rprop and JRprop needed an average of 80 and 70 epochs respectively.

The BFGS outperforms in terms of learning speed over the Rprop and iRprop but exhibits a small percentage of convergence (only 16 out the 100 runs); this is because the approximations of the Hessian matrix were close to singular. JRprop outperforms the other algorithms both in learning speed and classification success (the average improvement in learning speed achieved by the new method over Rprop, iRprop and BFGS is 55.4%, 54.8%, and 46.9% respectively).

Table 4
Results for the thyroid problem

Thyroid			Average	
Algorithm	Epochs	Speed (s)	Class. success (%)	Conv. (%)
Rprop	781	26.2 (+)	98.12 (−)	100
iRprop	778	25.8 (+)	98.12 (−)	100
BFGS	516	22.2 (+)	98.12 (−)	16
SCG	*D*	*D*	*D*	0
JRprop	669	11.6	98.23	100

5. Conclusions

In this paper we introduced a new class of sign-based schemes that are based on the composite nonlinear Jacobi process. An algorithm of this class that applies the bisection method to locate subminimizer approximations along each weight direction has been derived, and a simplified heuristic version has been proposed. This new heuristic algorithm constitutes and efficient improvement of the Rprop algorithm that is built on a theoretical basis.

We presented results on the behavior of the new algorithm in four pattern classification problems from the PROBEN1 repository and compared its performance with the Rprop, the iRprop (a recently introduced modification of the Rprop algorithm), the BFGS and the SCG algorithms. Additional experiments have been performed on other PROBEN1 problems (e.g. the ecoli and the yeast classification problems). The JRprop exhibited significantly better convergence speed than the Rprop and iRprop in all cases. There is of course need to conduct further research into the performance of JRprop in other pattern recognition problems to fully explore its advantages and identify possible limitations.

References

Axelsson, O., 1996. Iterative Solution Methods. Cambridge University Press, New York.

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

Brewster, M.E., Kannan, R., 1984. Nonlinear successive over-relaxation. Numer. Math. 44, 309–315.

Fang, L., Li, T., 1990. A globally optimal annealing learning algorithm for multilayer perceptrons with applications. In: Proc. Australian Joint Conf. on Artificial Intelligence. World Scientific, Perth, Australia, pp. 201–206.

Fletcher, R., 1975. Practical Methods of Optimization. John Wiley & Sons.

Fogel, B.D., Fogel, J.L., Porto, W.V., 1990. Evolving neural networks. Biol. Cybernet. 63, 487–493.

Gill, P.E., Murray, W., Wright, M.H., 1981. Practical Optimization. Academic Press, NY.

Husken, M., Stagge, P., 2003. Recurrent neural networks for time series classification. Neurocomputing 50, 223–235.

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

Magoulas, G.D., Vrahatis, M.N., Androulakis, G.S., 1997. Effective back-propagation training with variable stepsize. Neural Networks 10 (1), 69–82.

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 (7), 1769–1796.

Magoulas, G.D., Plagianakos, V.P., Vrahatis, M.N., 2002. Globally convergent algorithms with local learning rates. IEEE Trans. Neural Networks 13 (3), 774–779.

Moller, M.F., 1992. A scaled conjugated gradient algorithm, for fast supervised learning. Neural Networks 6, 525–533.

Murphy, P.M., Aha, D.W., 1994. UCI repository of machine learning databases, University of California, Department of Information and Computer Science, Irvine, CA. Available from: <http://www.ics.uci.edu/mlearn/MLRepository.html>.

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

Prechelt, L., 1994. PROBEN1—a set of benchmarks and benchmarking rules for neural network training algorithms. Technical report 21/94, Fakultät für Informatik, Universität, Karlsruhe.

Riedmiller, M., 1994. Rprop—Description and Implementation Details Technical Report, University of Karlsruhe, January.

Riedmiller, M., Braun, H., 1993. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: Proc. Internat. Conf. on Neural Networks, San Francisco, CA, pp. 586–591.

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

Scales, L.E., 1985. Introduction to Non-linear Optimization. MacMillan Publishers, pp. 34–35.

Sikorski, K., 1982. Bisection is optimal. Numer. Math. 40, 111–117.

Sikorski, K., 2001. Optimal Solution of Nonlinear Equations. Oxford University Press, New York.

Snedecor, G., Cochran, W., 1989. Statistical Methods, 8th ed. Iowa State University Press.

Stewart, G.W., 1973. Introduction to Matrix Computations. Academic Press, New York.

Tang, Z., Koehler, J.G., 1994. Deterministic global optimal FNN training algorithms. Neural Networks 7, 1405–1412.

Treadgold, N.K., Gedeon, T.D., 1998. Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm. IEEE Trans. Neural Networks 9 (4), 662–668.

Varga, R., 2000. Matrix Iterative Analysis, 2nd ed. Springer-Verlag, Berlin.

Vrahatis, M.N., 1988a. Solving systems of nonlinear equations using the nonzero value of the topological degree. ACM Trans. Math. Software 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. Software 14, 330–336.

Vrahatis, M.N., Magoulas, G.D., Plagianakos, V.P., 2003. From linear to nonlinear iterative methods. Appl. Numer. Math. 45 (1), 59–77.

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