Elsevier Editorial System(tm) for Neurocomputing
Manuscript Draft

Manuscript Number: NEUCOM-D-14-01891R1

Title: Convergence of RProp and variants

Article Type: Brief Paper/ Short Communication/ Letter

Keywords: Supervised learning; First-order training algorithms; Global convergence property; Rprop; GRprop; neuralnet

Corresponding Author: Dr. Todd M. Bailey, PhD

Corresponding Author's Institution: Cardiff University

First Author: Todd M. Bailey, PhD

Order of Authors: Todd M. Bailey, PhD

Abstract: This paper examines conditions under which the Resilient Propagation-Rprop algorithm fails to converge, identifies limitations of the so-called Globally Convergent Rprop-GRprop algorithm which was previously thought to guarantee convergence, and considers pathological behaviour of the implementation of GRprop in the neuralnet software package. A new robust convergent back-propagation-ARCprop algorithm is presented. The new algorithm builds on Rprop, but guarantees convergence by shortening steps as necessary to achieve a sufficient reduction in global error. Simulation results on four benchmark problems from the PROBEN1 collection show that the new algorithm achieves similar levels of performance to Rprop in terms of training speed, training accuracy, and generalization.

Revision summary and response to reviews

1. The paper has been shortened as suggested by the Associate editor, by rewriting throughout and by dropping two figures (original Figures 4 and 5). The body of the paper now has less than 4000 words.
2. Equations have been revised to show vectors in bold font as suggested by R1 and R3.
3. A fuller explanation of all figures has been provided in the text, as suggested by R2 and R3.
4. Figures have been embedded within the text as well as supplied in separate files, as suggested by R2.
5. Text in Section 3 around (ii) has been specifically clarified (and also shortened) as suggested by R2.
6. The text in Section 3 is now explicit that sign(g) indicates a column vector, so that d= -diag{……}sign(g) is also a column vector. R3 identified the need for this clarification.
7. Scare quotes have been added around “network” in Section 4 to emphasise that it is a degenerate case considered as an illustration precisely because it is transparently simple. R2 suggested that this illustration was inappropriately simplistic because this system is incapable of solving the XOR problem. Crucially, the input-output mappings considered here do not involve XOR or other violations of linear separability. The linear system considered here is a degenerate special case, and there actually is an exact solution for the specified “network” and training set. An optimisation algorithm which cannot solve this problem will not generally be robust in the face of more realistic architectures and mappings. Shortcomings of neuralnet’s GRprop could be illustrated with a more complex network, but this would add no additional force to the argument and would probably be harder to follow.
8. A formal proof of termination has been added to Section 5 as suggested by R1.
9. Additional simulations are reported in Section 6 as suggested by R1. These now cover four problem sets instead of the previous single problem. To save space, the simulations now dispense with GRprop, whose performance was far inferior and only of secondary interest.
10. Analysis of the results in Section 6 now takes advantage of the related samples design and focuses on the performance of ARCprop relative to Rprop. This change in analysis avoids lengthening the paper and multiplying the number of tables, though it has been necessary to add one new figure. More importantly, the new analysis addresses the concern of R1 that it was previously impossible to tell whether the distributions of results differed across algorithms. Significant differences are flagged in the new Figure 5.

Convergence of RProp and variants

Todd M. Bailey

Cardiff University

School of Psychology, Cardiff University, Tower Building, Park Place, Cardiff CF10 3AT, United Kingdom; email BaileyTM1@cardiff.ac.uk; phone +44 29 2087 5375

Abstract

This paper examines conditions under which the Resilient Propagation-Rprop algorithm fails to converge, identifies limitations of the so-called Globally Convergent Rprop-GRprop algorithm which was previously thought to guarantee convergence, and considers pathological behaviour of the implementation of GRprop in the neuralnet software package. A new robust convergent back-propagation-ARCprop algorithm is presented. The new algorithm builds on Rprop, but guarantees convergence by shortening steps as necessary to achieve a sufficient reduction in global error. Simulation results on four benchmark problems from the PROBEN1 collection show that the new algorithm achieves similar levels of performance to Rprop in terms of training speed, training accuracy, and .

Keywords

Supervised learning; First-order training algorithms; Global convergence property; Rprop; GRprop; neuralnet

1. **Introduction**

Back-propagation is a supervised learning technique for multi-layer connectionist models. It seeks to minimise the total error of the network output, across a batch of training inputs, by iteratively adjusting connection weights in a direction that will reduce error at each layer of the network [1]. The classic back-propagation algorithm takes a step of a certain global length on each iteration, in the direction of steepest descent. In contrast, "resilient back-propagation" (Rprop) determines step sizes for individual weight changes based on whether the current direction of the partial error derivative for that weight is the same as or different to the previous iteration [2]. If the gradient with respect to a particular weight has the same sign from one iteration to the next, the step size for that weight is increased. Conversely, if the individual gradient reverses direction, the step size is decreased. RProp is widely used for training connectionist models (e.g. [3-5]), and can also be used for other applications that require non-linear optimisation [6,7].

A major strength of RProp is that its learning rates self-adapt to the topology of the error surface, largely obviating the problem of identifying a single global learning rate or step size that will perform well on a particular problem. For example, consider the error landscape shown by the dark line in Figure 1, and the sequence of Rprop weight changes indicated by the arrows traversing the horizontal axis. Each step moves in the direction that will reduce the error. The step size increases if the descent direction for that particular weight is the same as the previous step (as from the left side of the figure). Steps grow shorter when the proximity of a local minimum is revealed by reversals of the error gradient from one step to the next, as illustrated near the right side of Figure 1.

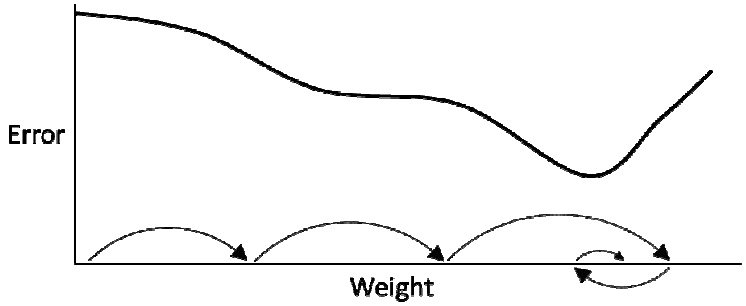


Figure 1. Schematic Rprop strategy. Arrows show weight changes on each iteration, with steps in the direction of descent as indicated by the slope of Error at the start of each step. Step sizes increase if the direction is the same as the previous, or decrease rapidly if the direction reverses.

2. **Non-convergence**

A "globally convergent" algorithm is guaranteed to find a solution which is at least locally optimal, in a finite amount of time, starting from almost anywhere in the

problem space [8]. RProp often converges quickly in comparison to a variety of other learning algorithms, across a range of problem domains [3,6,9-10] (but see [4]). Nevertheless, Rprop's strategy does not guarantee convergence (e.g. [9,11-13]). By way of illustration, Figure 2 shows hypothetical error contours for a two-dimensional weight space, *x* (horizontal) and *y* (vertical), with four hills separated by horizontal and vertical valleys around an arbitrary central hillock. Points A, C, E, F and G are high; point B is relatively low in this error landscape. The arrows indicate individual step sizes at each point in a potential infinite loop of Rprop steps. Starting from point A, Rprop races downhill towards B, increasing the step sizes on each iteration, up to the maximum step size (these steps are not shown). The step from point B combines maximal step sizes in each direction, leading to C. At point C, Rprop changes direction because of the reversal in the sign of the error gradient with respect to *y*. Velocity in the *x* direction is maintained, but the previous step in the *y* direction is back-tracked, leading to point D. Motion in the *y* direction then proceeds with shorter steps, lengthening as subsequent steps maintain the same gradient direction along this dimension. At E, there is a reversal in the sign of the gradient in the *x* direction, and again at G. In between, there is a second reversal of gradient in the *y* direction at point F, which counter-balances C. The cycle will now repeat indefinitely, failing to converge to a local error minimum.

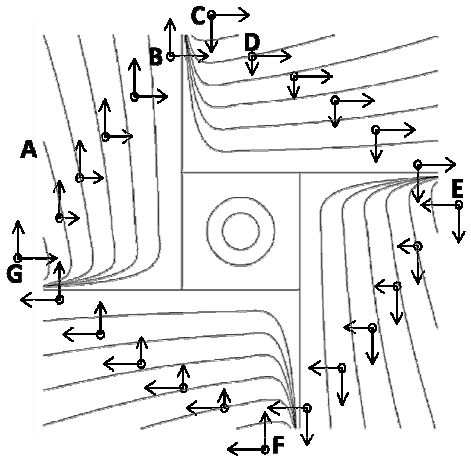


Figure 2. Contours on which Rprop may fail to converge, showing four hills separated by horizontal and vertical valleys (around an arbitrary central hillock). Points A, C, E, F and G are high; point B is relatively low in this error landscape. The arrows indicate individual step sizes at each point in a potential infinite loop of Rprop steps.

A change in the sign of an error gradient (as at C in Figure 2) indicates that the previous step skipped over an error minimum [2]. Intuitively, the solution is to take a shorter step from point B, but that is not what Rprop does. Instead, Rprop adjusts step lengths for the individual weight associated with the gradient reversal.

Unfortunately, it does not follow from a gradient reversal for one weight, that the reason the minimum was missed was that the step size *for that particular weight* was too large. For example, the jump from point B to C in Figure 2 reverses the gradient along the y axis, but this reversal is not a simple reflection of the local error topology relative to y. In general, the gradient associated with one dimension depends on the value of other dimensions as well, but Rprop ignores such interactions and consequently can fail to converge for some problems.

3. **GRprop**

A modified Rprop has been suggested in an effort to guarantee convergence [14]. The "globally convergent Rprop" (GRprop) is widely cited (e.g. [6,9,10,12,13,15-20]), and is an option in at least one prominent software package for neural network modelling [21]. However, it has apparently gone unnoticed that the proof of GRprop's convergence is mistaken.

Like Rprop, GRprop is an iterative descent procedure. Following a strategy proposed by [22], the intention of GRprop is to identify a descent direction for each step, and then use a global step length that satisfies the Wolfe conditions [23,24]. If these conditions are satisfied, then convergence is guaranteed because a local or global minimum will always be reached in a finite number of steps (assuming that the error is bounded from below and the error derivative is well-behaved). Conceptually, the Wolfe conditions ensure that

(i) the step be short enough to actually go downhill (i.e. to not leap across a valley and up onto an area with higher error than the previous iteration), and

(ii) the step length be long enough to reach the floor of a local minimum in a finite number of steps (i.e. the step length should not shrink so fast on each iteration that the minimum is never attained).

However, [14] do not specify a mechanism for adjusting the global step length to ensure that the Wolfe conditions are met. The GRprop algoritihm is therefore incomplete. Moreover, even if a suitable mechanism were specified, convergence would still not be assured. The Wolfe conditions presuppose a descent direction for each step. However, GRprop's direction selection process will sometimes mistakenly choose a direction of ascent rather than descent. The GRprop direction selection scheme uses the Rprop scheme to determine individual direction components $\eta_m > 0$ for each weight dimension except for one. The last direction component η_l is calculated to satisfy the constraint that $\sum_j \eta_j g_j = -\delta$, where g_j is the partial derivative of error with respect to weight *j*, and δ is a small positive number. Thus η_l is chosen according to (1), as recommended by [22].

$$\eta_l = -\frac{\sum_{m \neq l} \eta_m g_m + \delta}{g_l} \tag{1}$$

Having identified direction components η_j, the direction of weight updates for GRprop is $\boldsymbol{d} = -\mathrm{diag}\{\eta_1, \ldots, \eta_l, \ldots, \eta_n\}\mathrm{sign}(\boldsymbol{g})$, where $\mathrm{sign}(\boldsymbol{g})$ denotes the column vector of signs of the components of $\boldsymbol{g}$. The resulting column vector $\boldsymbol{d}$ will be a descent direction if the slope of the error surface in that direction is negative. The slope is $\boldsymbol{g}^{\mathrm{T}}\boldsymbol{d}/\|\boldsymbol{d}\|$, which is negative if $\boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} < 0$. Unfortunately, Equation (1) does not guarantee that $\boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} < 0$, especially if all the individual gradients are positive. For example, consider a trivial system with two weights, with partial derivatives of error $g_1 = g_2 = 1$. If direction component $\eta_1 > 0$ is assumed, then GRprop would set $\eta_2 = -\eta_1 - \delta < 0$. The direction of weight update would be $-\mathrm{diag}\{\eta_1, \eta_2\}\mathrm{sign}(\boldsymbol{g}) = -[\eta_1, \eta_2]^{\mathrm{T}} = [-\eta_1, \eta_1 + \delta]^{\mathrm{T}}$. The slope in this direction is proportional to $\boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} = -\eta_1 + \eta_1 + \delta = \delta > 0$. This is a direction of ascent, not descent, which renders the purported proof of GRprop convergence invalid.

In general, the slope in the direction chosen by GRprop is proportional to

$$\begin{aligned}\boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} &= -\sum_{m\neq l} g_m \eta_m \mathrm{sign}(g_m) - g_l \eta_l \mathrm{sign}(g_l) \\ &= -\sum_{m\neq l} \eta_m |g_m| + \left(\sum_{m\neq l} \eta_m g_m + \delta\right)\mathrm{sign}(g_l) \\ &= -\sum_{m\neq l} \eta_m [|g_m| - \mathrm{sign}(g_l) g_m] + \mathrm{sign}(g_l)\delta\end{aligned}$$

The absolute values either reinforce positive partial derivatives and cancel out negative ones, or vice versa, depending on whether g_l is positive or negative, as expressed in Equation (2).

(2)

$$\boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} = \begin{cases} -2\sum_{\{p|g_p>0\}} \eta_p g_p - \delta & g_l < 0 \\ 2\sum_{\{n|g_n<0\}} \eta_n g_n + \delta & g_l > 0 \end{cases}$$

Equation (2) has a positive upper bound of $\boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} \leq \delta$, which corresponds to a direction of ascent. The upper bound will be reached whenever all the component gradients are positive. In practice, GRprop will often identify a descent direction, but only if some of the individual gradients are negative.

The direction selection process of GRprop is the victim of an error in the original scheme from which it derives [22]. Conceptually, the intention of Equation (1) seems to be to identify a direction of *slight* descent. This could be achieved by calculating the last direction component η_l so that $\sum_j \eta_j |g_j| = \delta$ or $\sum_j \eta_j g_j^2 = \delta$, replacing Equation (1) with Equation (3) or Equation (4), respectively.

$$\eta_l = -\frac{\sum_{m\neq l} \eta_m |g_m| - \delta}{|g_l|} \tag{3}$$

$$\eta_l = -\frac{\sum_{m \neq l} \eta_m g_m^2 - \delta}{g_l^2} \quad (4)$$

A scheme incorporating Equations (3) or (4) might achieve theoretical global convergence, but may make too little progress on each iteration to converge in a reasonable number of iterations. An alternative might be to simply dispense with Equations (1), (3) and (4) entirely. Since Rprop always chooses a descent direction, it remains only to ensure an appropriate step length. We return to this idea shortly.

4. **Neuralnet**

Researchers looking for a neural network training package in the R language are likely to come across *neuralnet* [21]. Neuralnet implements a number of back-propagation algorithms, including Rprop and GRprop. Neuralnet's implementation of GRprop avoids inappropriate directions of the sort discussed above, but makes exceedingly slow progress towards the minimum in some circumstances.

Borrowing step size limits from Rprop, GRprop inappropriately applies them to the last direction component in GRprop. In Rprop, step sizes are always positive, and limits are imposed to avoid overflow and underflow of floating point computations [2]. Typical limits are $10^{-6} \leq \eta_i \leq 50$. Similar limits are appropriate for GRprop, except for the last direction component, which is chosen according to Equation (1) and will often be negative. However, neuralnet applies the same step size limits to all GRprop components, including the last. As a result, whenever Equation (1) produces a negative value, neuralnet imposes its lower limit instead ($\eta_{\min} = 10^{-10}$ by default).

The lower limit imposed by neuralnet ensures that a descent direction is always chosen. This can be verified by checking that $\boldsymbol{g}^{\mathrm{T}}\boldsymbol{d} < 0$, where $\boldsymbol{d} = -\mathrm{diag}\{\eta_1, \ldots, \eta_n\}\mathrm{sign}(\boldsymbol{g})$, with $\eta_i > 0$. We have

$$\boldsymbol{g}^T\boldsymbol{d} = -\sum_i g_i \eta_i \mathrm{sign}(g_i) = -\sum_i \eta_i |g_i| \leq 0$$

Unfortunately, if the step size lower limit gets applied to all dimensions of a particular problem, the performance of neuralnet's GRprop is perversely slow. For example, consider a trivial "network" consisting of a single input unit and a single linear output unit with a bias parameter. The two weights in this network are associated with (1) the connection from input to output unit, and (2) the bias of the output unit (conceptualised as a connection from a bias unit whose activation is fixed at 1). This network computes $y = w_0 + w_1 x$, where x and y are input and output activations, respectively, w_1 is the weight of the connection from x to y, and w_0 is the output unit bias. If we train with the target input→output mappings x=-1 → y=-1 and x=1 → y=1, neuralnet will seek values of w_0 and w_1 that minimise the sum squared error between the network output and the target values. The error surface has circular contours, as shown in Figure 3. The horizontal and vertical axes show potential

values of w_0 and w_1, respectively. The minimum is marked by the 'x' labelled C at $(w_0 = 0, w_1 = 1)$.

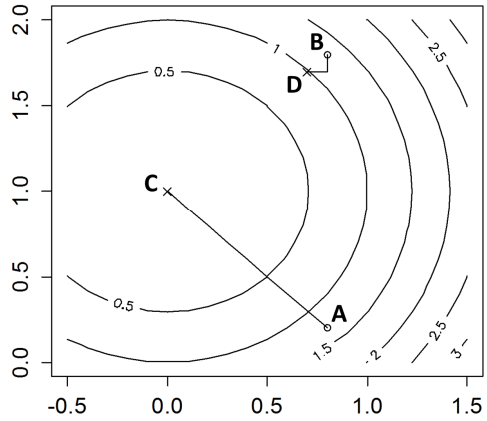


Figure 3. An error topology for which neuralnet's GRprop implementation converges quickly (from A to the minimum at C in 9 steps), or very slowly (from B towards C, requiring 10000 iterations to get as far as D). Circular contour lines indicate increasing error as a function of the radial distance from point C.

In addition to the circular error contours, Figure 3 also shows trajectories through weight space, beginning at two different starting points which are marked by small circles at A and B. From A at (w0 = 0.8, w1 = 0.2), neuralnet's GRprop algorithm follows a straight path and requires just nine steps to converge on the solution marked by an 'x' at C. In contrast, from B at (w0 = 0.8, w1 = 1.8), the algorithm makes some progress with its first two steps, but then gets bogged down with infinitesimal direction components which are substituted for negative values generated by equation (1). After 10,000 iterations the algorithm has progressed only as far as the 'x' labelled D.

Neuralnet's GRprop algorithm fails to converge in a reasonable number of steps when starting from virtually anywhere above and to the right of the target minimum in Figure 3. Whenever the individual gradients are all positive (as at point B), Equation (1) produces a negative value, and neuralnet imposes the lower limit $\eta_{\min}$. As a result, neuralnet's GRprop converges quickly when starting from point A, but not when starting from point B.

5. **A Robust Convergent backProp (ARCprop)**

As noted above, Rprop cannot cope with interactions between dimensions, and can consequently fail to converge. This section proposes a variant of Rprop that backtracks along all dimensions if a particular step does not achieve a sufficient decrease in error, and offers a proof of the convergence of the new algorithm.

Let $w_i^{(t)}$ be the weight of a particular connection at iteration *t*, and let $g_i^{(t)}$ be the partial derivative of error with respect to $w_i^{(t)}$, that is $g_i^{(t)} = \partial E^{(t)} / \partial w_i^{(t)}$. At each iteration, Rprop updates dimension weights in the direction of descent for each weight, as expressed in Equation (5).

$$w_i^{(t+1)} := w_i^{(t)} - \Delta_i^{(t)} sign\left(g_i^{(t)}\right) \tag{5}$$

The step size for each weight, $\Delta_i^{(t)}$, is derived from the previous iteration, increasing geometrically when the gradient direction remains the same and decreasing when the direction reverses, as expressed in Equation (6). For the sake of simplicity, Equation (6) omits terms that restrict $\Delta_i^{(t)}$ to the range $(\Delta_{\min}, \Delta_{\max})$.

$$\Delta_i^{(t)} := \begin{cases} \eta^+ \Delta_i^{(t-1)} & \text{if } g_i^{(t)} g_i^{(t-1)} > 0 \\ \eta^- \Delta_i^{(t-1)} & \text{if } g_i^{(t)} g_i^{(t-1)} < 0 \\ \Delta_i^{(t-1)} & \text{otherwise} \end{cases} \tag{6}$$

In addition, standard Rprop back-tracks individual weights when the gradient direction reverses. However, as illustrated in Figure 2, local back-tracking is insufficient; reverting just those weights that experience reversals of direction does not necessarily ensure a reduction of error at each iteration. Rather, if the global error worsens on some iteration it is necessary to back-track globally, and try a shorter step in the same direction. A variant of Rprop with global back-tracking and a number of other valuable features has been proposed recently by [7], but the resulting algorithm has not been proven to be globally convergent.

The basic idea of the algorithm proposed here is to take a series of downhill steps from the initial set of weights, until a vanishing error gradient indicates arrival at a local minimum. The algorithm terminates on iteration *t* if all $\left|g_i^{(t)}\right| \leq \delta \ll 1$. If this vanishing gradient condition is not met, then a step of size $\Delta_i^{(t)}$ is made in the direction of descent along each dimension *i*. If the step results in a sufficient decrease in global error, then the search continues from that new, improved location, according to Equation (6). A sufficient decrease is taken to be greater than the target gradient threshold, δ, multiplied by the shortest component step length, $\min_i \Delta_i^{(t)}$. If the error does not go down by at least that much, then the mean value theorem implies that some intermediate point must have a gradient small enough to satisfy the vanishing gradient criterion. In that case, a shorter step is initiated from the previous location; this assures convergence, as it prevents the algorithm from pursuing either increases in error or vanishingly small improvements in error, which could continue indefinitely. The ARCprop algorithm is described by pseudo-code in Figure 4.

if $\max_i \left|g_i^{(t)}\right| \leq \delta$ then return $w^{(t)}$
if $E^{(t)} > E^{(t-1)} - \delta \min_i \Delta_i^{(t-1)}$ then {

if $\max_i \Delta_{\mathrm{i}}^{(t-1)} \leq \Delta_{\min}$ then return $\mathrm{w}^{(t-1)}$

for each $\mathrm{w}_{\mathrm{i}}^{(t)}$ do {

$\mathrm{w}_{\mathrm{i}}^{(t)} := \mathrm{w}_{\mathrm{i}}^{(t-1)}$; $g_i^{(t)} := g_i^{(t-1)}$

$\Delta_{\mathrm{i}}^{(t)} := \max\left\{\eta^- \Delta_i^{(t-1)}, \Delta_{\min}\right\}$

}

} else {

for each $\mathrm{w}_{\mathrm{i}}^{(t)}$ do {

$$\Delta_i^{(t)} := \begin{cases} \min\left\{\eta^+ \Delta_i^{(t-1)}, \Delta_{\max}\right\} & \text{if } g_i^{(t)} g_i^{(t-1)} > 0 \\ \max\left\{\eta^- \Delta_i^{(t-1)}, \Delta_{\min}\right\} & \text{if } g_i^{(t)} g_i^{(t-1)} < 0 \\ \Delta_i^{(t-1)} & \text{otherwise} \end{cases}$$

}

}

for each $\mathrm{w}_{\mathrm{i}}^{(\mathrm{t})}$ do {

$\mathrm{w}_{\mathrm{i}}^{(t+1)} := \mathrm{w}_{\mathrm{i}}^{(t)} - \Delta_{\mathrm{i}}^{(t)} \mathrm{sign}\left(g_i^{(t)}\right)$

}

Figure 4. ARCprop algorithm. $\mathrm{E}^{(\mathrm{t})}$ is the global error at iteration *t*; $\mathrm{w}_{\mathrm{i}}^{(\mathrm{t})}$is the weight of connection *i*; $g_i^{(t)} = \partial E^{(t)} / \partial \mathrm{w}_{\mathrm{i}}^{(t)}$. The algorithm terminates when all gradients $g_i^{(t)} \leq \delta$, or when all step sizes $\Delta_i^{(t-1)} \leq \Delta_{\min}$.

Termination Theorem. Let $\mathrm{E}^{(t)} \geq 0$ be the global error on iteration *t*, $g_i^{(t)} = \partial E^{(t)} / \partial \mathrm{w}_{\mathrm{i}}^{(t)}$ be the gradient with respect to weight *i*, $\Delta_{\mathrm{i}}^{(t)}$ be a step size for weight *i*, η^+ and η^- be step size adjustment factors such that $0 < \eta^- < 1 < \eta^+$, and δ be a gradient threshold value such that $0 < \delta \ll 1$. Then the ARCprop algorithm described in Figure 4 is guaranteed to terminate after a finite number of iterations.

Proof. On each iteration *t* one of four alternatives transpires:

I. If all $\left|g_i^{(t)}\right| \leq \delta$ then the algorithm terminates, having found a local minimum.

II. If $\mathrm{E}^{(t)} > \mathrm{E}^{(t-1)} - \delta \min_i \Delta_{\mathrm{i}}^{(t-1)}$ **and** all step lengths are at the minimum length $\Delta_{\min}$, then the algorithm terminates, having bracketed a local minimum by the shortest allowed step.

III. If $\mathrm{E}^{(t)} > \mathrm{E}^{(t-1)} - \delta \min_i \Delta_{\mathrm{i}}^{(t-1)}$ **and** at least one step length can be shortened further, then the algorithm backtracks, having bracketed a local minimum loosely, and initiates another, shorter step.

IV. Otherwise $\mathrm{E}^{(t)} \leq \mathrm{E}^{(t-1)} - \delta \min_i \Delta_{\mathrm{i}}^{(t-1)}$, so step *t*-1 made a sufficient decrease in global error; then the algorithm adjusts step lengths up or down, and initiates another step.

Lemma 1. Case IV occurs at most $k \leq E^{(0)}/\delta\Delta_{\min}$ times.

On each occurrence of Case IV the error is reduced by an amount which is no less than the threshold gradient δ multiplied by the minimum step length $\Delta_{\min}$. If $E^{(0)}$ is the initial global error, then Case IV can occur at most $k \leq E^{(0)}/\delta\Delta_{\min}$ times before the global error vanishes. Once the error vanishes, further steps cannot satisfy the sufficient decrease condition and Case IV can no longer occur.

Lemma 2. Case III occurs at most $m \leq \log(\Delta_{\max}/\Delta_{\min})/\log(1/\eta^{-})$ times after the last occurrence of Case IV.

After the last occurrence of Case IV, step lengths can only decrease. On each occurrence of Case III, all step sizes are reduced by a factor of η^{-}, where $\eta^{-} < 1$. After *m* such occurrences, $\Delta_{\min} \leq \Delta_i^{(t)} \leq \max[\Delta_{\min}, \Delta_{\max}(\eta^{-})^{m}]$. At most $m \leq \log(\Delta_{\max}/\Delta_{\min})/\log(1/\eta^{-})$ reductions are required before all steps are as short as possible, at which point the condition for Case II will be satisfied instead of Case III.

Lemmas 1 and 2 establish that Cases III and IV can occur only a finite number of times before either Case I or Case II must occur, whereupon the algorithm terminates. In particular, if t^{*} is the iteration on which termination occurs, then from Lemmas 1 and 2 we have

$$t^{*} \leq 1 + \mathrm{E}^{(0)}/\delta\Delta_{\min} + \log(\Delta_{\max}/\Delta_{\min})/\log(1/\eta^{-})$$

The Termination Theorem is thus proved. The proof assumes that the error surface is differentiable, but it need not be continuous. If the error gradient is Lipschitz continuous with Lipschitz constant $K \leq 2\delta/\Delta_{\min}$, then Case II can never occur so the algorithm must eventually find a local minimum and terminate via Case I.

6. **Simulations**

To explore the practical viability of ARCprop, it was implemented within a revised neuralnet framework, and its performance was compared to the neuralnet "rprop+" algorithm, which implements the standard Rprop algorithm with local backtracking at gradient reversals. These simulations used four neural network classification problems from the PROBEN1 web site [25], namely cancer1, diabetes1, thyroid1, and genes2. These problems were among those benchmarked by [14].

Networks had sigmoid hidden and output nodes. Network configurations (numbers of input, hidden and output units) are given in Table 1, along with the total number of weights to be learned, the numbers of training and test items, and the termination thresholds. For each problem, ARCprop and Rprop were run from the same 100 sets of starting weights. Error was measured as sum squared deviation. Neuralnet

terminates when the maximum absolute error gradient reaches a specified threshold; this was arbitrarily set to 0.0075 times the square root of the number of training items in each particular problem, as shown in Table 1. Key training parameter values were $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_i^{(0)} = 0.1$, $\Delta_{\min} = 10^{-6}$, and $\Delta_{\max} = 50$.

Table 1. Characteristics of neural network problems, including I-H-O network structure (I input nodes, H hidden nodes, O output nodes), total number of weights (including a bias for each hidden and output node), the number of training and test items, and the termination threshold (maximum absolute error gradient).

Problem	I-H-O	N weights	Train-test	Threshold
Cancer	9-4-2-2	56	350-349	0.14
Diabetes	8-2-2-2	30	384-384	0.15
Genes	120-4-2-3	503	1588-1587	0.30
Thyroid	21-4-3	103	3600-3600	0.45

Performance was assessed in terms of the number of steps required to reach the stopping criterion, the fraction of steps on which error increased, the total error at the end of training, and per cent correct generalization in classifying untrained test patterns. For generalization, classification probabilities for each test pattern were calculated from proportional output activations, following Luce's choice rule [26]. Thus, the per cent correct generalization for a given test pattern was 100 times the activation of whichever output unit represented the correct response category, divided by the sum of the activations of all output units.

For each performance measure, the relative performance of the two algorithms was assessed by calculating the fraction of the 100 runs on which each algorithm achieved a higher score than the other. Results are shown in Figure 5. The shaded bars on the lower portion of the graph indicate the fraction of runs on which Rprop yielded a higher score than ARCprop on a particular performance measure. The unshaded bars above show the relative frequency of the alternative outcome. Percentages below 35% or above 65% represent significant deviations from chance across the 100 sets of starting weights, with family-wise error rate controlled at $p < .05$ by the Bonferroni correction for 16 comparisons. Significant comparisons are indicated in Figure 5 by asterisks attached to the corresponding labels along the horizontal axis.

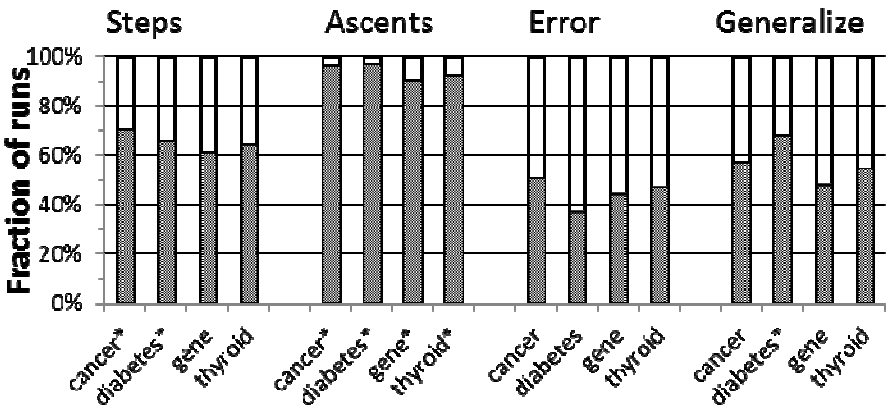


Figure 5. Relative performance of Rprop and ARCprop on four problems (cancer, diabetes, gene, and thyroid). Performance measures were number of steps required to reach the stopping criterion (Steps), the fraction of steps on which error increased (Ascents), the total error at the end of training (Error), and per cent correct generalization (Generalize). Bars show the fraction of runs on which Rprop (shaded) and ARCprop (unshaded) yielded a higher score than the other on a particular performance measure. Significant differences are indicated by asterisks.

The number of steps required tended to be greater for Rprop than for ARCprop, on all four problems. This difference was significant for the cancer and diabetes problems, but not for the genes and thyroid problems. The fraction of steps on which error increased was higher for Rprop than ARCprop, for most sets of starting weights for all four problems. This difference was significant for all four problems. The total error tended to be less for Rprop than for ARCprop, but the difference was not significant for any of the four problems. There was no consistent pattern of relative generalization performance, but Rprop achieved higher generalization scores than ARCprop from a majority of starting weights for the diabetes problem. This difference was significant.

Overall, the performance of ARCprop was generally comparable to Rprop, except that ARCprop almost always made fewer steps that increased the total error. It thus appears that the guaranteed convergence of ARCprop is achieved without incurring any substantial loss of performance.

7. **Conclusions**

The original Rprop algorithm automatically adapts to the topology of the error surface, and usually converges on a solution quickly. However, it is not guaranteed to find a solution in all cases. Published variants are similarly limited. In particular, GRprop, the “globally convergent Rprop”, is not; it sometimes takes steps in inappropriate directions, and when it moves in an appropriate direction it implicitly relies on an unspecified mechanism to ensure that steps are not too large. The

neuralnet implementation of GRprop always moves in an appropriate direction, but sacrifices speed to the point of being impractical in some situations.

A globally convergent algorithm can be obtained by modifying Rprop to detect increases in global error, so that a shorter step can be taken instead. The resulting algorithm, designated ARCprop (A Robust Convergent Rprop), achieved similar levels of performance to Rprop in four test problems.

Acknowledgements

Figure 2 was greatly facilitated by a free trial version of Rhinoceros 5 from http://www.rhino3d.com.

References

[1] D.E. Rumelhart, J.L. McClellend, Learning internal representations by error propagation, in: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge, 1986, 318–362.
[2] 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, 586–591.
[3] W. Schiffmann, M. Joost, and R. Werner. Optimization of the backpropagation algorithm for training multilayer perceptrons. Technical report, University of Koblenz, Institute of Physics, 1994.
[4] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Improving the convergence of the backpropagation algorithm using learning rate adaptation methods, Neural Comput. 11 (1999) 1769–1796.
[5] C. Igel, M. Hüsken, Empirical evaluation of the improved Rprop learning algorithms, Neurocomputing 50 (2003) 105–123.
[6] L. Kocsis, C. Szepesvari, Universal parameter optimisation in games based on SPSA, Machine learning 63 (2006) 249-286.
[7] A. Kotsialos, Nonlinear optimisation using directional step lengths based on RPROP, Optimization Letters 8 (2014) 1401-1415.
[8] J. Nocedal, S.J. Wright, Numerical Optimization, 2nd ed., Springer, New York, 2006.
[9] S. Mandal, P.V. Sivaprasad, S. Venugopal, Capability of a feed-forward artificial neural network to predict the constitutive flow behavior of as cast 304 stainless steel under hot deformation, Trans. ASME J. Eng. Mater. Technol. 129 (2007) 242–247.
[10] D. Kanevsky, G. Heigold, S. Wright, H. Ney, Overview of large scale optimization for discriminative training in speech recognition, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (2012) 5233-5236, DOI: 10.1109/ICASSP.2012.6289100.

[11] A. Kotsialos, M. Papageorgiou, Nonlinear optimal control applied to coordinated ramp metering, IEEE Trans. Control Syst. Technol. 10 (2004) 920–933.
[12] G.D. Magoulas, M.N. Vrahatis, Adaptive algorithms for neural network supervised learning: a deterministic optimization approach, International Journal of Bifurcation and Chaos, 16 (2006) 1929-1950.
[13] S. Wiesler, A. Richard, R. Schluter, H. Ney, A critical evaluation of stochastic algorithms for convex optimization, Acoustics, IEEE International Conference on Speech and Signal Processing (ICASSP), (2013) 6955-6959.
[14] A.D. Anastasiadis, G.D. Magoulas, M.N. Vrahatis, New globally convergent training scheme based on the resilient propagation algorithm, Neurocomputing, 64 (2005) 253–270.
[15] S. Winter, B. Brendel, I. Pechlivanis, K. Schmieder & C. Igel, Registration of CT and intraoperative 3-D ultrasound images of the spine using evolutionary and gradient-based methods. IEEE Transactions on Evolutionary Computation, 12 (2008) 284-296.
[16] D. Kalamatianos, A.D. Anastasiadis, P. Liatsis, A nonextensive method for spectroscopic data analysis with artificial neural networks, Brazilian Journal Of Physics 39 (2009) 488-494.
[17] A. Kostopoulos, T. Grapsa. Self-scaled conjugate gradient training algorithms, Neurocomputing, 72 (2009) 3000-3019.
[18] H. Schaeben. Comparison of mathematical methods of potential modeling, Mathematical Geosciences, 44 (2012) 101-129.
[19] C. Bergmeir, J.M. Benítez, Neural networks in R using the Stuttgart neural network simulator: RSNNS. Journal Of Statistical Software, 46 (2012) 1-26.
[20] I.E. Livieris, P. Pintelas, A new conjugate gradient algorithm for training neural networks based on a modified secant equation, Applied Mathematics And Computation, 221 (2013) 491-502.
[21] F. Günther, S. Fritsch, neuralnet: Training of neural networks, The R Journal 2 (2010) 30-38.
[22] M.N. Vrahatis, G.D. Magoulas, V.P. Plagianakos, From linear to nonlinear iterative methods, Appl. Numer. Math. 45 (2003) 59–77.
[23] P. Wolfe, Convergence conditions for ascent methods, SIAM Rev. 11 (1969) 226–235.
[24] P. Wolfe, Convergence conditions for ascent methods. II: some corrections, SIAM Rev. 13 (1971) 185–188.
[25] 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).
[26] R.D. Luce, Individual choice behavior: A theoretical analysis. New York: Wiley, (1959).

Tables

Table 1. Characteristics of neural network problems, including I-H-O network structure (I input nodes, H hidden nodes, O output nodes), total number of weights (including a bias for each hidden and output node), the number of training and test items, and the termination threshold (maximum absolute error gradient).

Problem	I-H-O	N weights	Train-test	Threshold
Cancer	9-4-2-2	56	350-349	0.14
Diabetes	8-2-2-2	30	384-384	0.15
Genes	120-4-2-3	503	1588-1587	0.30
Thyroid	21-4-3	103	3600-3600	0.45

Figure captions

Figure 1. Schematic Rprop strategy. Arrows show weight changes on each iteration, with steps in the direction of descent as indicated by the slope of Error at the start of each step. Step sizes increase if the direction is the same as the previous step (as from the left side of the figure), or decrease rapidly if the direction reverses (as shown near the right side).

Figure 2. Contours on which Rprop may fail to converge, showing four hills separated by horizontal and vertical valleys (around an arbitrary central hillock). Points A, C, E, F and G are high; point B is relatively low in this error landscape. The arrows indicate individual step sizes at each point in a potential infinite loop of Rprop steps.

Figure 3. An error topology for which neuralnet's GRprop implementation converges quickly (from A to the minimum at C in 9 steps), or very slowly (from B towards C, requiring 10000 iterations to get as far as D). Circular contour lines indicate increasing error as a function of the radial distance from point C.

Figure 4. ARCprop algorithm. $\mathrm{E}^{(\mathrm{t})}$ is the global error at iteration *t*; $\mathrm{w}_{\mathrm{i}}^{(\mathrm{t})}$is the weight of connection *i*; $g_i^{(t)} = \partial E^{(t)} / \partial \mathrm{w}_{\mathrm{i}}^{(t)}$. The algorithm terminates when all gradients $g_i^{(t)} \leq \delta$, or when all step sizes $\Delta_i^{(t-1)} \leq \Delta_{\min}$.

Figure 5. Relative performance of Rprop and ARCprop on four problems (cancer, diabetes, gene, and thyroid). Performance measures were number of steps required to reach the stopping criterion (Steps), the fraction of steps on which error increased (Ascents), the total error at the end of training (Error), and per cent correct generalization (Generalize). Bars show the fraction of runs on which Rprop (shaded) and ARCprop (unshaded) yielded a higher score than the other on a particular performance measure. Significant differences are indicated by asterisks.

Figure 1

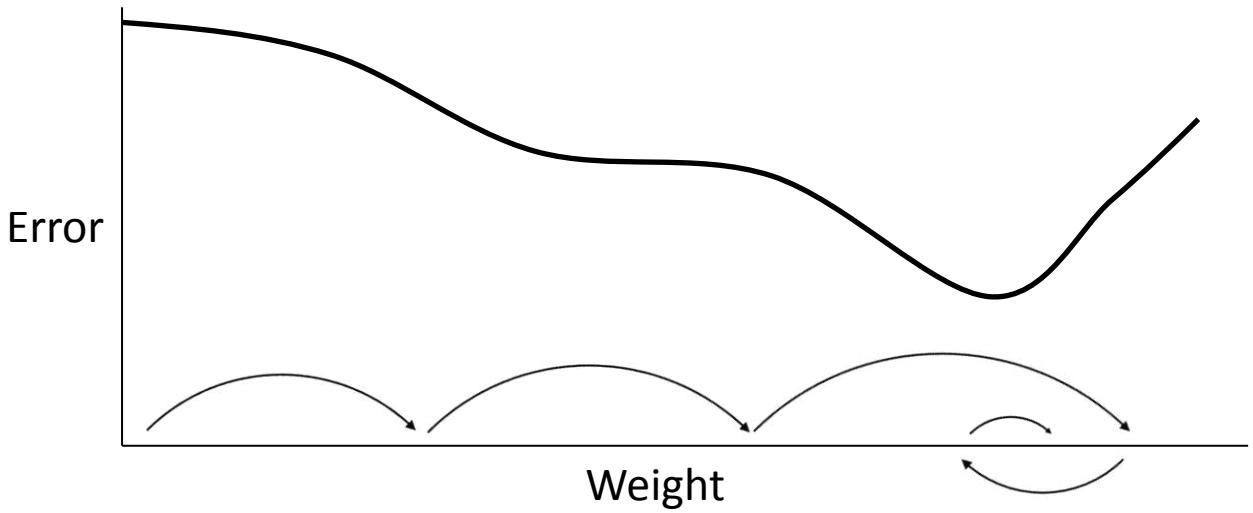


Figure 2

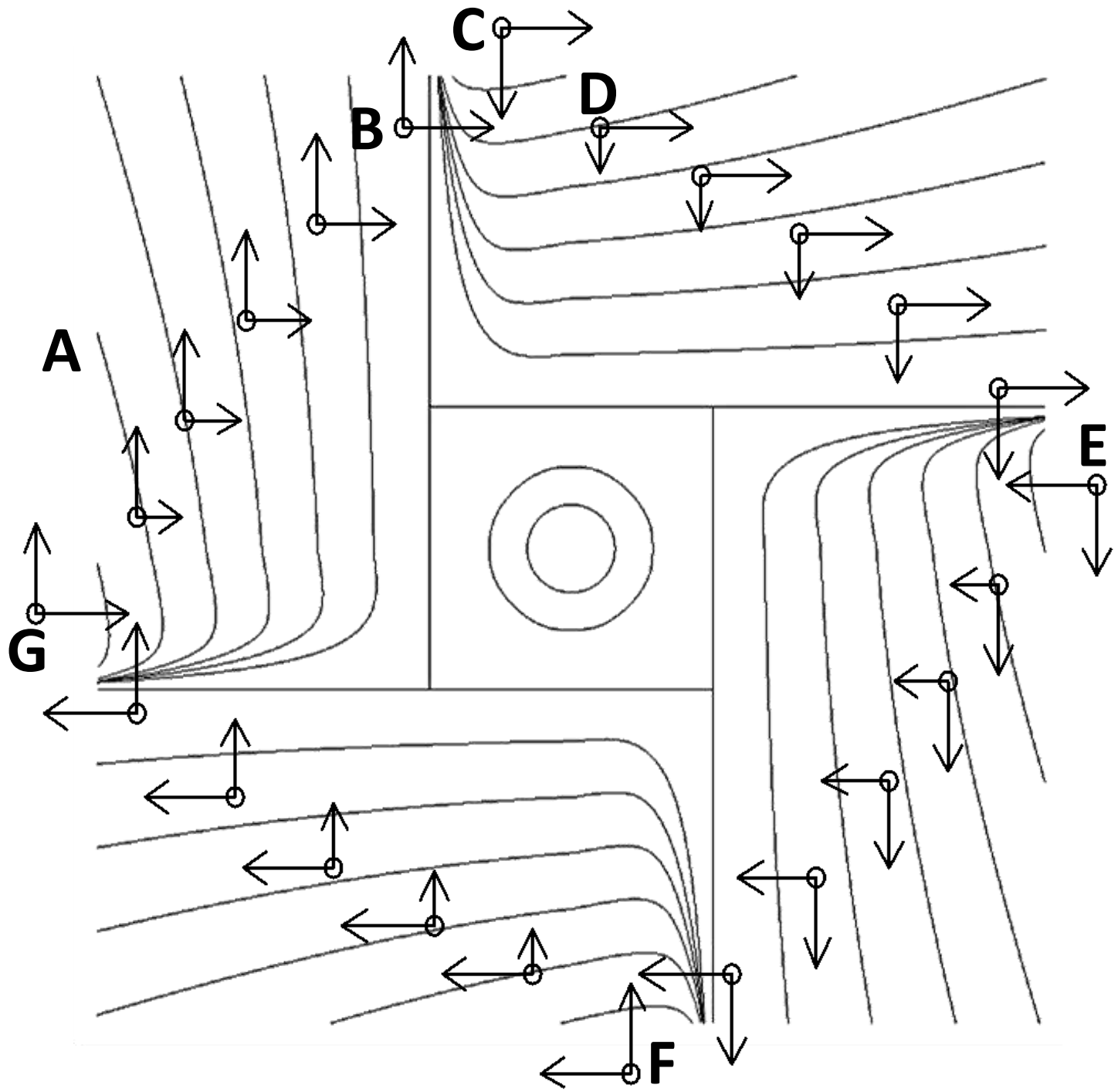


Figure 3

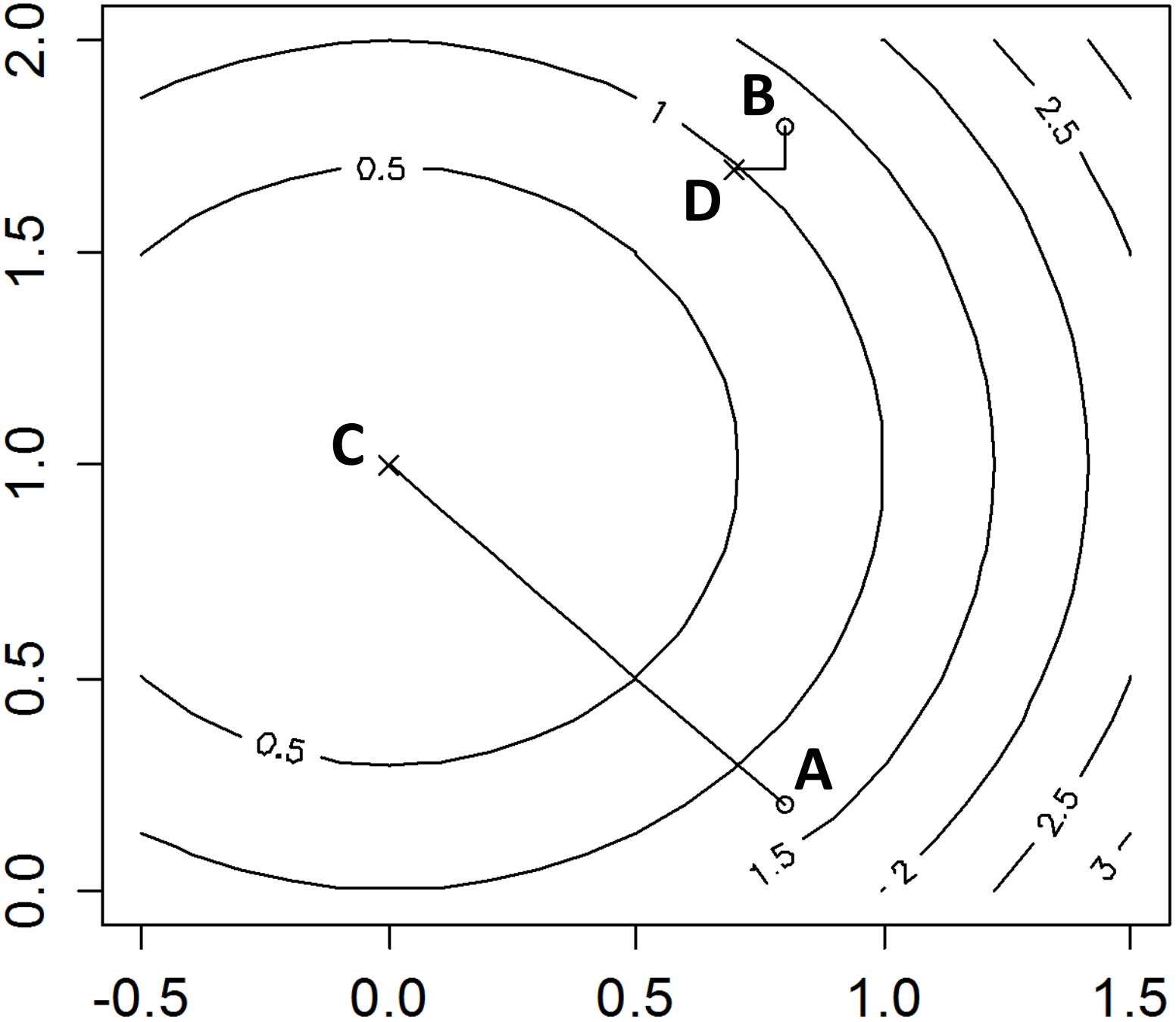


Figure 4

if $\max_i \left|g_i^{(t)}\right| \leq \delta$ then return $\mathrm{w}^{(t)}$

if $\mathrm{E}^{(t)} > \mathrm{E}^{(t-1)} - \delta \min_i \Delta_{\mathrm{i}}^{(t-1)}$ then {

    if $\max_i \Delta_{\mathrm{i}}^{(t-1)} \leq \Delta_{\min}$ then return $\mathrm{w}^{(t-1)}$

    for each $\mathrm{w}_{\mathrm{i}}^{(t)}$ do {

        $\mathrm{w}_{\mathrm{i}}^{(t)} := \mathrm{w}_{\mathrm{i}}^{(t-1)}$; $g_i^{(t)} := g_i^{(t-1)}$

        $\Delta_{\mathrm{i}}^{(t)} := \max\left\{\eta^- \Delta_i^{(t-1)}, \Delta_{\min}\right\}$

    }

} else {

    for each $\mathrm{w}_{\mathrm{i}}^{(t)}$ do {

$$\Delta_i^{(t)} := \begin{cases} \min\left\{\eta^+ \Delta_i^{(t-1)}, \Delta_{\max}\right\} & \text{if } g_i^{(t)} g_i^{(t-1)} > 0 \\ \max\left\{\eta^- \Delta_i^{(t-1)}, \Delta_{\min}\right\} & \text{if } g_i^{(t)} g_i^{(t-1)} < 0 \\ \Delta_i^{(t-1)} & \text{otherwise} \end{cases}$$

    }

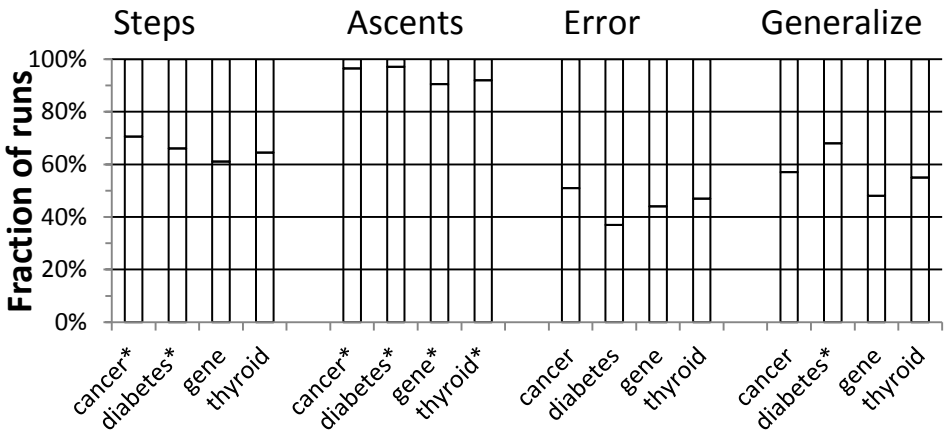
}

for each $\mathrm{w}_{\mathrm{i}}^{(\mathrm{t})}$ do {

    $\mathrm{w}_{\mathrm{i}}^{(t+1)} := \mathrm{w}_{\mathrm{i}}^{(t)} - \Delta_{\mathrm{i}}^{(t)} \mathrm{sign}\left(g_i^{(t)}\right)$

}

Figure 5


Steps
Ascents
Error
Generalize
Fraction of runs
100%
80%
60%
40%
20%
0%
cancer*
diabetes*
gene
thyroid
cancer*
diabetes*
gene*
thyroid*
cancer
diabetes
gene
thyroid
cancer
diabetes*
gene
thyroid

Vitae

Todd M. Bailey is a member of the academic staff in the School of Psychology, Cardiff University. He has previously been a researcher in the Department of Experimental Psychology and the McDonnell-Pew Centre for Cognitive Neuroscience, at the University of Oxford. He has also developed computer-assisted engineering tools as a software engineer for Hewlett-Packard in Colorado Springs. He received the BSc Computer Science, MA Linguistics, and PhD Linguistics with Cognitive Science at the University of Minnesota. He is particularly interested in computational models of language processing and categorisation.

Photo of the author(s)
Click here to download high resolution image

Highlights

- Rprop can go up as well as down, and can get stuck in loops.
- GRprop sometimes chooses an ascent direction and hence is not globally convergent.
- neuralnet's implementation of GRprop can be exceedly slow on some simple problems.
- New variant of Rprop is proposed which adjusts step length to assure convergence.