THE SARPROP ALGORITHM: A SIMULATED ANNEALING ENHANCEMENT TO RESILIENT BACK PROPAGATION

N.K. Treadgold and T.D. Gedeon
School of Computer Science & Engineering
The University of New South Wales
Sydney N.S.W. 2052 AUSTRALIA
{ nickt | tom }@cse.unsw.edu.au

ABSTRACT
Back Propagation and its variations are widely used as methods for training artificial neural networks. One such variation, Resilient Back Propagation (RPROP), has proven to be one of the best in terms of speed of convergence. Our SARPROP enhancement, based on Simulated Annealing, is described in this paper and is shown to increase the rate of convergence for some problems. The extension involves two complementary modifications: weight constraints early in training combine with noise to force the network to perform a more thorough search of the initial weight space, before allowing the network to refine its solutions as training continues.

INTRODUCTION
There have been a number of refinements made to the BP algorithm (Tollenaere, 1990; Jacobs, 1988; Fahlman, 1988), with arguably the most successful in general being the Resilient Back Propagation method or RPROP (Riedmiller and Braun, 1993; Riedmiller 1994).

The two major differences between BP and RPROP are that RPROP modifies the size of the weight step taken adaptively, and the mechanism for adaptation in RPROP does not take into account the magnitude of the gradient ($\delta E/\delta w_{ij}$) as seen by a particular weight, but only the sign of the gradient (positive or negative). This allows the step size to be adapted without having the size of the gradient interfere with the adaptation process (Riedmiller, 1993). In a number of previous BP variants, the learning parameter, η, was varied adaptively (Tollenaere, 1990; Jacobs, 1988). There was, however, no account taken of the current gradient magnitude (which is combined with the learning parameter to give the step size). The size of the gradient is unforeseeable, and hence it has the potential to disrupt the adaptation of the learning parameter.

The RPROP algorithm works by modifying each weight by an amount $\Delta w_{ij}(t)$, termed the update value (or learning parameter), in such a way as to decease the overall error. All update values are initialised to the value Δ_0. The update value is modified in the following manner: if the current gradient ($\delta E/\delta w_{ij}(t)$) multiplied by the gradient of the previous step is positive (that is the gradient direction has remained the same), then the update value is multiplied by a value η^+ (which is greater than one). Similarly, if the gradient product is negative, the update value is multiplied by the value η^- (which is less

than one). The update value remains the same if the product equals zero. This results in the update value for each weight adaptively growing, or shrinking, as a result of the sign of the gradient seen by that weight. There are two limits placed on the updates values: a maximum Δ_{max}, and a minimum Δ_{min}.

SARPROP

While RPROP can be extremely fast in converging to a solution, it suffers from the same problem faced by all gradient descent based methods: it can often converge to local minima. SARPROP attempts to address this problem by using the method of Simulated Annealing (SA). SA methods are a well known technique in training artificial neural networks, and have been applied to the Back Propagation algorithm (Burton and Mpitsos, 1992) with good results in terms of speed of converge. SA, in general, involves the addition of a random noise factor during weight updates. The amount of noise added is often associated with a 'temperature' value which decreases the effect of the noise as training progresses. The addition of noise allows the network to move in a direction which is not necessarily the direction of steepest descent. The benefit this provides is that it can help the network escape from local minima. Burton and Mpitsos (1992) have also shown that SA can help increase the speed of convergence of the BP algorithm, and conclude that noise "...simply permits or facilitates greater access to such pathways that are not easily reached in the networks not containing noise..."

There are two enhancements made to the RPROP algorithm to obtain SARPROP. First, a noise factor is introduced. Noise is added to a weight when both the error gradient changes sign in successive epochs, and the magnitude of the update value is less than a value proportional to the current RMS error. The amount of noise added is proportional to both the current RMS error value and a temperature factor (which is based on the current epoch).

The reason for adding noise to the update value only when both the error gradient changes sign, and the update value is below a given setting, was to minimise the disturbance to the normal adaptation of the update value. Following this scheme means that the update value is only modified by noise when it has a relatively small value (indicating a number of previous gradient crossings). This can allow the weight to jump out of local minima (Figure 1), while minimising the disturbance to the adaptation process. The amount of noise added decreases as the training continues because of the temperature term. The noise added is also influenced by the current RMS error value, a technique used by Burton and Mpitsos (1992).

The second modification involves the addition of a weight decay term to the error function. The weight decay term is also associated with a temperature factor, which results in the influence of this term decreasing as training proceeds. The new error gradient is shown below:

$$\delta E/\delta w_{ij}^{\ SARPROP} = \delta E/\delta w_{ij} \ - \ k_1 * w_{ij} * 2^{-T*epoch}$$

The reason for adding a weight decay term to the error function is to constrain the weights to smaller values at the beginning of training. This permits a more complete search of the initial weight space before allowing the weight values to increase. The search is complemented by the simultaneous addition of noise to the weight updates. The intention is that by constraining the weights initially, a more promising solution can be obtained. This solution can then be expanded upon once the weight values are able to increase.

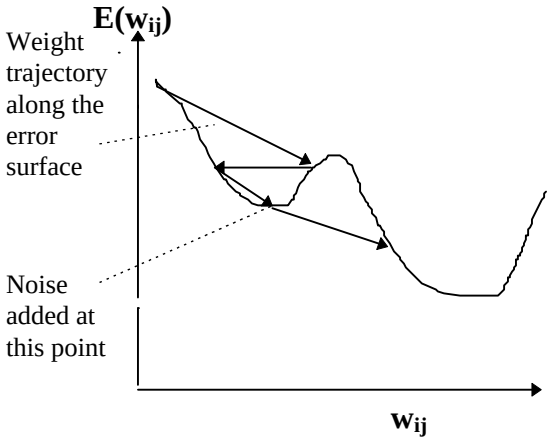

Figure 1: *An example of a weight trajectory along an error surface, and the result of SARPROP's noise addition.*

After enhancements have been incorporated, the resulting SARPROP algorithm is:

$\forall i,j: \Delta_{ij}(t) = \Delta_0$
$\forall i,j: \delta E/\delta w_{ij}(t-1) = 0$
Repeat
  Compute Gradient $\delta E/\delta w^{SARPROP}(t)$
  For all weights and biases
    if ($\delta E/\delta w_{ij}(t-1) * \delta E/\delta w_{ij}(t) > 0$**) then**
      $\Delta_{ij}(t) =$ **minimum (**$\Delta_{ij}(t-1) * \eta^{+}, \Delta_{max}$**)**
      $\Delta w_{ij}(t) =$ **- sign (**$\delta E/\delta w_{ij}(t)$**)** $* \Delta_{ij}(t)$
      $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
      $\delta E/\delta w_{ij}(t-1) = \delta E/\delta w_{ij}(t)$
    else if ($\delta E/\delta w_{ij}(t-1) * \delta E/\delta w_{ij}(t) < 0$**) then**
      if ($\Delta_{ij}(t-1) < k_2 * error^2$**) then**
        $\Delta_{ij}(t) = \Delta_{ij}(t-1) * \eta^{-} + k_3 * r * error * 2^{-T*epoch}$
      else
        $\Delta_{ij}(t) = \Delta_{ij}(t-1) * \eta^{-}$
        $\Delta_{ij}(t) =$ **maximum (**$\Delta_{ij}(t), \Delta_{min}$**)**
      $\delta E/\delta w_{ij}(t-1) = 0$
    else if ($\delta E/\delta w_{ij}(t-1) * \delta E/\delta w_{ij}(t) = 0$**) then**
      $\Delta w_{ij}(t) =$ **- sign (**$\delta E/\delta w_{ij}(t)$**)** $* \Delta_{ij}(t)$
      $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
      $\delta E/\delta w_{ij}(t-1) = \delta E/\delta w_{ij}(t)$
Until (converged)

In the algorithm *error* corresponds to (normalised) RMS error, *T* corresponds to the temperature, k_1 , k_2,, and k_3 are constants, and *r* is a random number between 0 and 1.

COMPARATIVE RESULTS AND DISCUSSION

To test the effectiveness of the SARPROP algorithm, we decided to compare its performance against that of the RPROP algorithm on a number of standard test problems. Since the two algorithms have a number of parameters in common, these were all set to the same values in both algorithms. The standard settings for these parameters are (Braun and Riedmiller, 1993): $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_{max} = 50$, $\Delta_{min} = 1x10^{-6}$. These values remained constant throughout the comparisons. The initial update value, Δ_0, was set to 0.1 for all problems in both RPROP and SARPROP. The value chosen for Δ_0 has been shown to be of little significance in regards to the performance of RPROP since it is quickly adapted (Braun and Riedmiller 1993).

There are a number of parameters introduced by SARPROP which need to be set before training commences. After some experimentation good values for the parameters k_1 , k_2 , and k_3 were found: 0.01, 0.1, and 3 respectively. These values appear to be largely problem independent and so are treated as constant for all tests. The value for the temperature, *T*, however determines the rate at which both noise and the weight decay terms are reduced. It was observed that this parameter contributed significantly to the convergence speed of the SARPROP algorithm, and thus was tuned to the specific problem. In order to gauge the effect of this parameter on the SARPROP algorithm, results were recorded for a number of SARPROP runs, each using a different temperature value.

To compare RPROP and SARPROP, each algorithm was used to train a network on a given problem, with both networks having identical starting weight values. Starting weight values were initialised using the formula: *r * 2.4 / (inputs)* where *r* is a random number between 0 and 1 and *inputs* is the number of weight connections to the neuron for which the current weight value is being calculated (the fan-in). For each problem, all training attempts were repeated using different random starting weights (50 times for less complex tasks and 25 times for more difficult tasks). A maximum number of training epochs was selected for each problem, and training was halted if this number was reached. In this case the training was defined to be non-converging.

In order to compare the two algorithms we decided to take three measurements: the average number of epochs required in training (Av.), the standard deviation of the number of epochs (S.D.), and the number of non-converged training runs (N.Cvg.). For classification problems the 40-20-40 threshold and margin criterion was used (Fahlman, 1988). In using the 40-20-40 criterion, a class has been learnt correctly if the neuron's output is in the correct upper or lower 40% of its output range. For regression problems, a RMS error value was specified to indicate convergence.

The first series of comparisons were performed using the odd parity datasets, with varying numbers of input bits. The training set was made up all of possible combinations of input bits. Training was halted when the 40-20-40 classification criterion was satisfied on the training set.

Task:	**Xor3 (Parity 3)**			**SARPROP**		**RPROP**
Network:	3-3-1	***T***	0.010	0.015	0.020	-
Activation:	sigmoid (+1 , -1)	**Av.**	33.9	32.28	34.88	36.42
Max Epochs:	300	**S.D.**	14.71	13.21	14.73	66.8
Test Runs:	50	**N.Cvg.**	0	0	0	3

Task:	**Parity 4**			**SARPROP**		**RPROP**
Network:	4-6-1	***T***	0.010	0.015	0.020	-
Activation:	sigmoid (+1 , -1)	**Av.**	150.22	104.78	113.7	505.32
Max Epochs:	1000	**S.D.**	93.32	53.73	132.03	439.73
Test Runs:	50	**N.Cvg.**	0	0	1	21

Task:	**Parity 5**			**SARPROP**		**RPROP**
Network:	5-7-1	***T***	0.010	0.015	0.020	-
Activation:	sigmoid (+1 , -1)	**Av.**	137.66	114.94	115.16	421.12
Max Epochs:	1000	**S.D.**	41.94	40.94	30.34	433.74
Test Runs:	50	**N.Cvg.**	0	0	0	17

Task:	**Parity 6**			**SARPROP**		**RPROP**
Network:	6-9-1	***T***	0.010	0.015	0.020	-
Activation:	sigmoid (+1 , -1)	**Av.**	493.16	391.0	427.72	857.72
Max Epochs:	1000	**S.D.**	217.62	287.09	270.23	286.53
Test Runs:	25	**N.Cvg.**	3	4	4	20

Task:	**Parity 7**			**SARPROP**		**RPROP**
Network:	7-11-1	***T***	0.010	0.015	0.020	-
Activation:	sigmoid (+1 , -1)	**Av.**	587.28	558.96	448.16	849.32
Max Epochs:	1000	**S.D.**	235.0	321.14	272.3	303.2
Test Runs:	25	**N.Cvg.**	5	8	4	20

The results on the parity datasets demonstrate SARPROP's increased average convergence speed over a number of different temperature values. Much of the increased convergence speed can be attributed to the much higher probability that SARPROP will converge to a solution, as compared with that of RPROP, which fails to converge much more frequently.

The SinCos problem consists of input angles ranging over the values 0 to 2π with output patterns consisting of the Sine and Cosine of these angles. The training set was made up of 315 patterns and was considered learnt at an RMS error of 0.06.

Task:	**SinCos**			**SARPROP**		**RPROP**
Network:	1-4-2	***T***	0.030	0.050	0.070	-
Activation:	sigmoid (+1 , -1)	**Av.**	648.0	659.08	793.14	885.02
Max Epochs:	4000	**S.D.**	793.95	815.13	1012.1	1192.7
Test Runs:	50	**N.Cvg.**	1	1	3	6

SARPROP is seen to converge on average more rapidly than RPROP for the SinCos problem. SARPROP, again, converges more often than RPROP.

The results obtained during the tests allow the construction of some heuristics which can be helpful in setting the temperature value in SARPROP. In general, the more simple the error surface (in terms of ease of traversal by a gradient descent method) the higher the temperature value should be in SARPROP (which equates to a reduced influence of both noise and weight decay). Conversely, problems which are more difficult to solve by gradient descent methods suggest the use of a smaller temperature value in SARPROP.

CONCLUSION

A Simulated Annealing addition to the RPROP algorithm, SARPROP, has been proposed. On a number of tests SARPROP is shown to have better convergence properties than RPROP. SARPROP has a number of additional advantages. First, SARPROP forces a more thorough search of the initial weight space, and thus is more likely to discover a solution with small weight values (which can help generalisation). Second, SARPROP is inherently less dependent on the initial weight values for finding a good solution, as can be seen from the large number of converged solutions in the test results. Finally, SARPROP is only slightly more complex computationally than RPROP, and hence it can be used to solve larger problems unsuited to other optimisation techniques. Further research on SARPROP's performance using more difficult problems is being pursued to confirm these results.

REFERENCES

Burton, R.M. and Mpitsos, G.J. (1992) Event-Dependent Control of Noise Enhances Learning in Neural Networks. *Neural Networks* **5**, 627-637.

Fahlman, S.E. (1988) An Empirical Study of Learning Speed in Back-Propagation Networks (CMU-CS-88-162), Technical Report, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Jacobs, R.A. (1988) Increased Rates of Convergence Through Learning Rate Adaption. *Neural Networks* **1**, 295-307.

Riedmiller, M. (1994) Rprop - Description and Implementation Details, Technical Report, University of Karlsruhe.

Riedmiller, M. and Braun, H. (1993) A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: Ruspini, H., (Ed.) *Proc. of the ICNN 93, San Francisco*, pp. 586-591.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) Learning internal representations by error propagation. In: Rumelhart, D.E. and McClelland, J.L., (Ed.) *Parallel distributed processing: Explorations in the microstructure of cognition. vol 1. Foundations*, pp. 318-362. Cambridge, MA: MIT Press.