SA-Prop: Optimization of Multilayer Perceptron Parameters using Simulated Annealing

P.A. Castillo[1], J. González[1], J.J. Merelo[1], A. Prieto[1], V. Rivas[2] and G. Romero[1]

[1] Department of Architecture and Computer Technology
University of Granada
Campus de Fuentenueva
E. 18071 Granada (Spain)
[2] Departament of Computer Science
University of Jaén
Avda. Madrid, 35
E. 23071 Jaén (Spain)

e-mail: `geneura@kal-el.ugr.es` phone: +34 958 24 31 62

Abstract. A general problem in model selection is to obtain the right parameters that make a model fit observed data. If the model selected is a Multilayer Perceptron (MLP) trained with Backpropagation (BP), it is necessary to find appropriate initial weights and learning parameters. This paper proposes a method that combines Simulated Annealing (SimAnn) and BP to train MLPs with a single hidden layer, termed SA-Prop. SimAnn selects the initial weights and the learning rate of the network. SA-Prop combines the advantages of the stochastic search performed by the Simulated Annealing over the MLP parameter space and the local search of the BP algorithm.
The application of the proposed method to several real-world benchmark problems shows that MLPs evolved using SA-Prop achieve a higher level of generalization than other perceptron training algorithms, such as QuickPropagation (QP) or RPROP, and other evolutive algorithms, such as G-LVQ.

1 Introduction

Whatever the application using them, artificial neural networks (ANNs) must be designed (structured in layers and correctly connected), their initial parameters must be established and then trained. Either BP or a variant of this algorithm is normally used as the training mechanism; examples include QuickProp (QP) [4] and RPROP (Riedmiller and Braun [14, 13]), among other evolutionary based approaches.

Whichever method is chosen, the training mechanism is an iterative gradient descent algorithm designed to minimize step by step the difference between the actual output vector of the network and the desired output vector. Although this method is used successfully in many fields, especially for pattern recognition, due to its learning ability, it does encounter certain difficulties in practice:

1. The convergence tends to be extremely slow.
2. Convergence to the global optimum is not guaranteed: it may reach the closest local minimum to the search space point where the method started.
3. Learning constants must be guessed heuristically.

The necessity of a method able to solve this learning and optimization problem arises from the fact that the parameters to be adjusted depend on the problem to solve, on the type of network (number of hidden units to use) and on the training/test sets.

In this paper we propose to use Simulated Annealing [1], which is a method inspired by Nature: it is an analogy between the way in which a material cools and freezes into a minimum energy crystal (the annealing process), thus providing maximum resistance, and the search for the minimum of a problem.

Metropolis et al. [17] originally proposed this method as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. The connection between this algorithm and mathematical minimization was proposed by Kirkpatrick et al. [18] as an optimization technique for combinatorial and other problems.

The primary advantage of SimAnn is its ability to avoid becoming trapped at local minima. Thus, the ability to find the global optimum is not related to the initial conditions (i.e., the starting point). SimAnn is also very simple to implement.

The proposed method is an example of a *hybrid approach* of a SimAnn method and BP to train ANNs.

The use of SimAnn to design ANNs has been applied in several approaches:

– *Search for the optimal set of weights* of a pre-established topology net, as used by Tambouratzis in [16], with a method that uses SimAnn to optimize a network (two layers of nodes with binary activation values and a sigmoid activation function) to solve the satisfiability (SAT) problem of propositional calculus.
– *Search for the optimal learning parameters*, including weights, having previously established the number of neurons and the connectivity between them, as described by Vergara et al. in [10], using a method to optimize different parameters of feedforward neural networks (topology and learning parameters) by the use of global search methods like SimAnn.

Other *genetic approaches* modify the BP algorithm, as Kinnebrock proposes in [5], where a mutation operator adds or substracts an amount to the weight values after each iteration. The disadvantage of this method is that such modifications do not necessarily improve the net classification error.

Our approach combines the first two described above, including changes of weights and of learning parameters as the SimAnn change operator.

The aim of this paper is to present an algorithm to tune parameters and set the initial weights of a MLP, based on Simulated Annealing and BP, that obtains better results than the BP alone. We show that the proposed algorithm, **SA-Prop**, obtains a better degree of generalization than that obtained by BP. We intend to make use of the capacity of both algorithms: the ability of SimAnn to find a solution close to the optimum and the ability of the BP to tune a solution and reach the nearest local minimum by means of local search from the solution found by SimAnn. A neural net with a pre-established hidden layer size evolves, searching through the MLP space with that architecture, to reach the optimum.Thus, SimAnn searches and optimizes the initial weight setting and the learning rate to train the MLP.

The rest of this paper is structured as follows: Section 2 presents the SA-Prop algorithm. Section 3 describes the results obtained, followed by a brief conclusion in Section 4.

2 Method

In a classical SimAnn problem, a *cost function* (objective function f) to be minimized is defined and then, from an initial random solution, different solutions (k, called number of changes) are derived and compared with the retained solution. The better solution (least cost) is kept, but retaining a solution with a greater cost is allowed with a certain probability, which decreases according to the temperature value.

The algorithm search is as follows: it generates a new state from the old one by means of a change operator (new state generator). If the new one is better, then it is acceptsed. However, if the new one is worse, the algorithm will accept it with an acceptation probability of:

$$p_a = e^{-\frac{\Delta f}{T}} \tag{1}$$

where Δf is defined as the increment of the cost function and T is the temperature control parameter. The temperature is then decreased using the temperature reduction function f_T.

The simplest decrement rule is

$$T_{n+1} = \alpha T_n \tag{2}$$

where α is a constant smaller than 1. This *exponential cooling scheme* was proposed by Kirkpatrick et al. [18] with $\alpha = 0.95$

Kirkpatrick [6] suggested that T_0 depends on the scaling of f according to

$$T_0 = -\frac{\Delta f^*}{ln p_a} \tag{3}$$

where Δf^* is the average objective increase observed and p_a is the initial acceptance probability (0.8 is usually used).

The designed algorithm is an enhanced version of the described in [8], and it is described in the following pseudocode:

Algorithm 1

```
$n = 0$
initialize temperature T
select an state (MLP) $s_{old}$ at random
evaluate $s_{old}$ (train it using training set and obtaining the testing error)
repeat
    for j=1 to k
        select a new MLP $s_{new}$ in the neighborhood
            of $s_{old}$ by changing $s_{old}$
        $\Delta f = f(s_{old}) - f(s_{new})$
        if $(\Delta f < 0)$ OR $(random(0,1) < e^{-\frac{\Delta f}{T}})$
            then $s_{old} = s_{new}$
    endfor
    $T = f_T(T, n)$
    $n = n + 1$
until $(T < T_{min})$
Use the last state (MLP) to obtain the validation error
```

In SA-Prop the **objective function** (f) is given by the classification accuracy, which is obtained by dividing the number of hits between the total number of examples in the validating set.

In SA-Prop, the initial weights of the network are evolved using the change operator (see below), which is made possible by the **EO** (Evolvable|Evolutionary Objects) library philosophy (see the end of this section): any object with a fitness (cost function) can be evolved. Moreover, this agrees with the spirit of Z. Michalewicz: $GA + DS = EP$ (GA plus Data Structures equals Evolutionary Programming) [8].

The **change operator** acts directly on the MLP object, but only *initial weights* and the *learning rate* are subjected to evolution, not the weights obtained after training (a clone of the MLP is created to compute its cost function value, thus the initial weights remain unchanged in the original MLP). This operator modifies the weights of certain neurons, at random, depending on the application rate. It is based on the algorithm presented in [5], which modifies the weights of the network after each epoch of network training, adding or subtracting a small random number that follows *uniform* distribution with the interval $[-0.1, 0.1]$. This operator is applied with a 40% probability that is, 40% of weights are changed.

The learning rate is modified by adding a small random number that follows *uniform* distribution in the interval $[-0.05, 0.05]$. This operator was used with an application probability of 40%, which was found empirically to generate better states than did lower probabilities.

As **temperature reduction function** (f_T) SA-Prop uses

$$f_T(T_0, n) = \frac{T_0}{1 + n} \tag{4}$$

This choice was made because of the smoothness with which this function reduces the temperature factor. Thus the acceptance of a worse solution is more closely controlled by the system.

The SimAnn algorithm sets the initial T and reduces this initial value step by step until it reaches the minimum temperature allowed (T_{min}), when it stops.

When evaluating each state (MLP) to obtain the cost function value, we can indicate either the limit of training epochs or the maximum error allowed. In the tests performed here, a limit of epochs was established.

We have used the BP variant known as the perceptron training algorithm *QuickProp* [4], which is one of the best in terms of avoiding local minima, one of the problems of BP. This algorithm, together with SimAnn, improves the probability of avoiding local minima.

3 Experiments and results

We used **EO** library as a toolbox to develop the SA-Prop, because of the facility that this library offers to evolve any object with a fitness function (cost function). It is a C++ toolbox which defines interfaces for many classes of algorithms used in evolutionary computation and, at the same time, provides some examples that use those interfaces. It is available at `http://geneura.ugr.es/~jmerelo/EO.html`. One of the objectives of using EO is to make it easier to reproduce results, which is why all objects evolved with EO should also be made available, together with the articles that mention them. SA-Prop is available as a part of the G-Prop package (see [12]) at `http://krypton.ugr.es/~pedro/G-Prop.htm`

First, we show the dynamics of SA-Prop as a SimAnn algorithm and then compare it with other methods.

The evolution of classification error on the validation set during a typical run is shown in figure 1. Initially, some large changes in *objective function* are accepted and some areas far from the optimum are explored. As execution continues (and temperature falls), changes are smaller. At the end of the run, the algorithm is searching around the optimum.

SA-Prop was run 10 times with the same parameter settings and a different random initialization for each benchmark used.

The tests used to assess the accuracy of both methods must be chosen carefully, because some tests (exclusive-or problem) are not suitable for certain capacities of the BP algorithm, such as generalization [3]. Our opinion, shared by Prechelt [11], is that to test an algorithm, at least two real world problems should be used.

The tests were applied as follows: each data set was divided into three disjointed parts, for training, validating and testing. Thus, in order to obtain the *cost or objective function value*, the MLP is trained using the training set and the classification error on the validating set is obtained. Once the SimAnn is finished (when T_{min} is reached), the classification error on the testing set is calculated, which is the result shown.

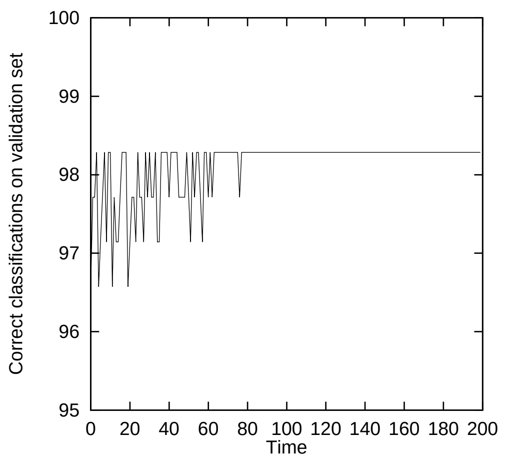

Fig. 1. Initially, some large changes in *objective function* are accepted and some areas far from the optimum are explored. As execution continues (and temperature falls), changes are smaller. Halfway through the run, the algorithm is already searching around the optimum.

To obtain the results with the program that implements QP, the methodology used was to train as many MLPs as were trained by SA-Prop on a run (about 150 MLP), with the same topology as that used by SA-Prop, and with *the same learning parameter found* by the proposed method, for a particular benchmark. Then the validating set is used to obtain the test error for each MLP; once the best MLP is found, the testing error is obtained on the testing set.

Cancer. This dataset is from the UCI machine learning dataset "Wisconsin breast cancer database". This breast cancer database was obtained from Dr. William H. Wolberg [15]. An exhaustive report, by Prechelt, on this dataset (and others) is given in [11]. Each sample has 10 attributes plus the class attribute: Sample Code Number, Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, Mitoses, Class (0 for benign, 1 for malignant). The class distribution in the original set is the following: 65.5% Benign and 34.5% Malignant.

DNA Helicases. This is a problem of electron microscopy image classification, concerning the structure of a representative hexametric helicase: the large T antigen of Simian Virus 40 [2]. When observed in the electron microscope, large T antigen preparations mainly show either a characteristic roughly circular view with a stain penetrating region in the centre (called a top view) or a view with a rectangular shape (side view). This training/test set has been used in [7] to classify images of the large T antigen of Simian Virus 40. The set consists of 197 examples, divided into the training set (78 examples), testing set (60) and validation (59). Each example consists of 25 inputs, corresponding to the 25 blocks of the image to be classified, and an output: 0 is side view and 1 is top view.

The test was carried out, using the described benchmarks, with our own version of QP (available as part of the G-Prop package, see [12]) and the proposed method, SA-Prop.

For the *Cancer* problem the SimAnn was executed using an initial temperature of 0.2, a minimum temperature of 0.001, and 10 as the number of changes (parameter k, see algorithm 1), with a change probability of 0.4. The states, MLPs with 9 inputs, 2 outputs and a number of hidden units equal to that used by Prechelt in [11], were evaluated with 70 epochs and an initial learning coefficient of 0.01 (the method will search the optimum learning coefficient). Each execution with these parameters took several minutes.

The results obtained for the first test (error rate for the test set), compared with those obtained by Prechelt in [11], are shown in table 1.

Cancer		Error%± Std Dev	Hidden Units	Learning parameter
Cancer 1	QP	3.4 ± 0.5	6	0.1834
	SA-Prop	1.1 ± 0.5	6	0.18 ± 0.09
	Prechelt	1.149	$4 + 2$	-
Cancer 2	QP	5.5 ± 0.5	12	0.1188
	SA-Prop	4.2 ± 0.6	12	0.12 ± 0.09
	Prechelt	5.747	$8 + 4$	-
Cancer 3	QP	4.7 ± 0.2	8	0.1081
	SA-Prop	3.2 ± 0.9	8	0.11 ± 0.08
	Prechelt	2.299	$4 + 4$	-

Table 1. Results of evaluating QP and SA-Prop for the Cancer problem, together with the results obtained by Prechelt in [11]. This table shows the average error rate, the number of hidden units and the learning parameter found. The number of hidden units was established according to that found by Prechelt. The learning rate used to obtain the results with QP is the average (exact value) of those found by the SimAnn method.

It is evident that SA-Prop outperforms QP, even with the same learning parameter, which means that searching for the best initial weights also decreases the error rate. In general, SA-Prop obtains MLPs with a lower generalization error than other methods, except in Cancer 3, for which Prechelt [11] presents better results. Curiously enough, this is the same pattern that was obtained in a previous paper [12], which might indicate some systematic error in Prechelt's paper. On average, for all Proben Cancer problems, SA-Prop is slightly better than Prechelt's RPROP (2.8% with SA-Prop versus 3.1% with RPROP).

For the *DNA helicases* problem, the SimAnn was executed using an initial temperature of 0.2, a minimum temperature of 0.001, and 10 as the number of changes (k parameter), with a change probability of 0.4 (these were "default" parameters). The states, MLPs with 25 inputs, 2 outputs and 7 or 11 hidden units, were evaluated with 70 epochs and an initial learning coefficient of 0.01. Each execution with these parameters took several minutes.

Table 2 shows the results obtained with both algorithms and compares them to the ones shown in [7].

DNA Helicases		Error%± Std Dev	Hidden Units	Learning parameter
QP		6 ± 4	7	0.1637
		7 ± 5	11	0.1067
SA-Prop		5 ± 3	7	0.16 ± 0.06
		6 ± 3	11	0.11 ± 0.07
G-LVQ	20 generations	18 ± 2	2.3 ± 0.4	-
	50 generations	13 ± 3	2.2 ± 0.3	-
	100 generations	15 ± 5	2.2 ± 0.3	-

Table 2. Results of evaluating QP and SA-Prop on the classification of different views of large T antigen of Simian Virus 40 (error on test set, hidden layer size and the learning parameter found), together with the results obtained with G-LVQ [7]. The number of hidden units was established empirically. The learning rate used to obtain the results with QP is the average (exact value) of those found by the SimAnn method.

In the *DNA helicases* problem, it is evident that SA-Prop outperforms QP. The results obtained in [7] show that G-LVQ takes around 20 generations to achieve an error of 18 ± 2, while SA-Prop performs better, achieving an error of 5 ± 3 and 6 ± 3, for different hidden layer sizes.

The optimal values of the learning parameters are between 0.09 and 0.2. This result is not generalizable to all kind of perceptrons, since it is the change of this value and the adaption of the initial weights that minimizes the error rate, as can be seen in the results. However, compared with the previous values obtained with QP in [12], the evolved learning parameter by itself improves results.

4 Conclusions

This paper presents SA-Prop, an algorithm to train MLPs based on SimAnn and BP. It is seen that the proposed method achieves better results than does BP alone.

In particular, SA-Prop obtains a much higher degree of generalization (that is, error on a previously unseen test set) than that achieved by other BP algorithms such as QP or RPROP.

SimAnn evaluates each state, taking into account the classification ability. The weights and the learning rate of the first state are generated randomly and these are the weights and learning rate that SimAnn will evolve, not the weights obtained after training (a clone of the MLP is created to compute its cost function value, so that the initial weights remain unchanged in the original MLP).

This strategy attempts to avoid Lamarckism (i.e., the fact that MLPs inherit the trained weights from their parents) but, at the same time, it is a good strategy to avoid local minima.

A change operator has been designed to generate new states, changing the weights and learning rate by adding or subtracting a small random number, depending on the application rate.

Several real-world benchmarks (cancer 1, 2 and 3, and DNA helicases) have been used to test the proposed algorithm and to compare it with others [7, 11]. The results show that SimAnn obtains a MLP whose classification accuracy is better than that obtained by training a MLP using only conventional procedures.

Future work will extend the presented method and will include the improvement of the change operator described here to allow the search over topology space (hidden layer size). The QP algorithm could be used as an operator in SimAnn, as suggested in [9], and the presented algorithm applied to solve real problems.

5 Acknowledgements

This work has been supported in part by the CICYT project BIO96-0895 (Spain), the DGICYT project PB-95-0502 and the FEDER I+D project 1FD97-0439-TEL1.

References

1. E.H.L. Aarts and J. Korst. Simulated Annealing and Boltzmann Machines. *John Wiley, Chichester, U.K.*, 1989.
2. C. San Martin; C. Gruss; J.M. Carazo. Six molecules of SV40 large t antigen assemble in a propeller-shaped particle around a channel. *Journal of Molecular Biology, 268, 15-20*, 1997.
3. S. Fahlman. An empirical study of learning speed in back-propagation networks. Technical report, Carnegie Mellon University, 1988.
4. S.E. Fahlman. Faster-Learning Variations on Back-Propagation: An Empirical Study. *Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann*, 1988.
5. Werner Kinnebrock. Accelerating the standard backpropagation method using a genetic approach. *Neurocomputing, 6, 583-588*, 1994.
6. S. Kirkpatrick. Optimization by Simulated Annealing - Quantitative Studies. *J. Stat. Phys. 34, 975-986*, 1984.
7. J.J. Merelo; A. Prieto; F. Moran; R. Marabini and J.M Carazo. Automatic Classification of Biological Particles from Electron-microscopy Images Using Conventional and Genetic-algorithm Optimized Learning Vector Quantization. *Neural Proccessing Letters 8: 55-65, 1998*, 1998.
8. Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs , Second, Extended Edition.* Springer-Verlag, 1994.
9. D.J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. *Proc. 11th Internat. Joint Conf. on Artificial Intelligence, 762-767*, 1989.
10. V. Vergara; S. Sinne; C. Moraga. Optimal Identification Using Feed-Forward Neural Networks. *Lectures Notes in Computer Science, vol. 930, 1052-1059*, 1995.

11. Lutz Prechelt. PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, D-76128 Karlsruhe, Germany, September 1994. Anonymous FTP: /pub/papers/techreports/1994/1994-21.ps.Z on ftp.ira.uka.de.
12. P.A. Castillo; J. Gonzalez; J.J. Merelo; V. Rivas; G. Romero; A. Prieto. G-Prop: Global Optimization of Multilayer Perceptrons using GAs. *submitted to Neurocomputing*, 1998.
13. M. Riedmiller. Description and Implementation Details. Technical report, University of Karlsruhe, 1994.
14. M. Riedmiller and H. Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. *In Ruspini, H., (Ed.) Proc. of the ICNN93, San Francisco, pp. 586-591*, 1993.
15. O. L. Mangasarian; R. Setiono and W.H. Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis. *Large-scale numerical optimization, Thomas F. Coleman and Yuying Li, editors, SIAM Publications, Philadelphia 1990, pp 22-30*, 1990.
16. T. Tambouratzis. Simulated Annealing Artificial Neural Networks For The Satisfiability (SAT) Problem. *Artificial Neural Nets and Genetic Algorithms, 340-343*, 1995.
17. N. Metropolis; A.W. Rosenbluth; M.N. Rosenbluth; A.H. Teller; E. Teller. Equations of State Calculations by Fast Computing Machines. *J. Chem. Phys. 21,1087-1092*, 1958.
18. S. Kirkpatrick; C.D. Gerlatt; M.P. Vecchi. Optimization by Simulated Annealing. *Science 220, 671-680*, 1983.