Pergamon

Nonlinear Analysis, Theory, Methods & Applications, Vol. 30, No. 7, pp. 4545–4550, 1997
Proc. 2nd World Congress of Nonlinear Analysts
© 1997 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0362-546X/97 $17.00 + 0.00

PII: S0362-546X(96)00369-0

ON THE ALLEVIATION OF THE PROBLEM OF LOCAL MINIMA IN BACK–PROPAGATION

G.D. MAGOULAS[†], M.N. VRAHATIS[‡] and G.S. ANDROULAKIS[‡]

[†]Department of Electrical & Computer Engineering, University of Patras, GR-261.10, Patras, Greece; and
[‡]Department of Mathematics, University of Patras, GR-261.10 Patras, Greece.

Key words and phrases: Feed–forward neural networks, supervised training, numerical optimization methods, back–propagation of error.

1. INTRODUCTION

Supervised training of a feed–forward neural network (FNN) can be viewed as the minimization of an error function that depends on the weights of the network. This perspective gives some advantage to the development of effective training algorithms because the problem of minimizing a function is well known in the field of numerical analysis.

If there is a fixed, finite set of input–output pairs, the square error over the training set which contains P representative cases is:

$$E(w) = \sum_{p=1}^{P}\sum_{j=1}^{N_L}\left(y_{j,p}^L - t_{j,p}\right)^2 = \sum_{p=1}^{P}\sum_{j=1}^{N_L}\left[\sigma^L\left(\sum_{i=1}^{N_{L-1}} w_{ij}^{L-1,L} y_{i,p}^{L-1} + \theta_j^L\right) - t_{j,p}\right]^2. \qquad (1.1)$$

This equation formulates the error function to be minimized, in which $t_{j,p}$ specifies the desired response at the j–th neuron of the output layer at the input pattern p and $y_{j,p}^L$ is the output at the j–th neuron of the output layer L that depends on the weights of the network and σ is a nonlinear activation function, such as the well known sigmoid $\sigma(x) = (1+e^{-x})^{-1}$. The weights in the network can be expressed using vector notation as:

$$w = \left(\ldots, w_{ij}^{l-1,l}, w_{i+1\,j}^{l-1,l}, \ldots, w_{N_{l-1}\,j}^{l-1,l}, \theta_j^l, w_{i\,j+1}^{l-1,l}, w_{i+1\,j+1}^{l-1,l}, \ldots\right)^\top,$$

where $w_{ij}^{l-1,l}$ is the connection weight from the i–th neuron ($i = 1,\ldots,N_{l-1}$) at the $(l-1)$ layer to the j–th neuron at the l–th layer, θ_j^l denotes the bias of the j–th neuron ($j = 1,\ldots,N_l$) at the l–th layer ($l = 2,\ldots,L$). This formulation defines the weight vector as a point in the N–dimensional real Euclidean space $\mathbb{R}^N$, where N denotes the total number of weights and biases in the network.

Minimization of $E(w)$ is attempted by updating the weights using a training algorithm. The weight update vector describes a direction in which the weight vector will move in order to reduce the network training error. The weight update equation for any training algorithm is thus:

$$w^{k+1} = w^k + \Delta w^k, \quad k = 0,1,\ldots, \qquad (1.2)$$

where w^{k+1} is the new weight vector, w^k is the current weight vector and Δw^k the weight update vector.

The commonly used training methods are gradient based algorithms such as the popular back–propagation (BP) algorithm [1]. It is well known that the BP algorithm leads to slow training and often yields suboptimal solutions [2]. This contribution presents a technique that alleviates the problem of occasional convergence to local minima in BP training.

2. BACK-PROPAGATION TRAINING AND LOCAL MINIMA

The BP minimizes the error function $E(w)$ using the Steepest Descent (SD) [3] with constant stepsize μ, i.e. :

$$w^{k+1} = w^k - \mu \nabla E(w^k), \quad k = 0, 1, \ldots . \tag{2.1}$$

The optimal value of the stepsize μ depends on the shape of the N-dimensional error function. The gradient, ∇E, is computed by applying the chain rule on the layers of the FNN (see [1]).

Attempts to speed up back-propagation training have been made by dynamically adapting the stepsize μ during training [4,5], or by using second derivative related information [6,7,8]. However, these BP-like training algorithms are based on local minimization methods. They have no mechanism that allows them to escape the influence of a local minimum. Convergence of an algorithm to a local minimum prevents a network from learning the entire training set and results in inferior network performance.

Intuitively, the existence of local minima is due to the fact that the error function is the superposition of nonlinear activation functions that may have minima at different points which results in nonconvex error surface. Convergence to local minima can be caused by the insufficient number of hidden nodes as well as improper initial weight vector.

Recently, several researchers have presented conditions on the network architecture, the training set and the initial weight vector that allow BP to reach the optimal solution [2,9,10]. However, conditions such as the linear separability of the patterns and the pyramidal structure of the FNN [2] as well as the need for a great number of hidden neurons (as many neurons as patterns to learn), make these interesting results not easily interpretable in practical situations even for problems as simple as the XOR.

3. THE PROCEDURE OF SIMULATED ANNEALING

Simulated annealing refers to the process in which random noise in a system is systematically decreased at a constant rate so as to enhance the response of the systems.

In the numerical optimization framework, Simulated Annealing (SA) [11] is a procedure that has the capability to move out of regions near local minima. SA is based on random evaluations of the cost function, in such a way that transitions out of a local minimum are possible. It does not guarantee of course, to find the global minimum, but if the function has many good near-optimal solutions, it should find one.

The performance of the SA [12] as observed on typical neural network training problems, is not the appropriate one. SA is characterized by the need for a number of function evaluations greater than that commonly required for a single run of common training algorithms and by the absence of any derivative related information. In addition, the problem with minimizing the neural network error function is not the well defined local minima but the broad regions that are nearly flat. In this case the so-called Metropolis move is not strong enough to move the algorithm out of these regions [13].

In [14] SA is incorporated in the weight update vector as follows:

$$\Delta w^k = -\mu \nabla E(w^k) + nc2^{-dk}$$

where n is a constant controlling the initial intensity of the noise, $c \in (-0.5, +0.5)$ is a random number and d is the noise decay constant. In our experiments we have applied this technique, named SA1, for updating the weights from the beginning of the training as proposed by Burton et al. [14]. Alternatively, we update the weights using BP until convergence to a global or local minimum is achieved. In the latter case, we switch to SA1. This combined BP with SA1 is named BPSA.

4. THE DEFLECTION PROCEDURE

It is well known that in order to minimize the error function E we require a sequence of weight vectors $\{w^k\}_0^\infty$, where k indicates iterations, that converges to a minimizer of E. Assuming that this sequence converges to a local minimum $r \in \mathbb{R}^N$ we can formulate the following function:

$$F(w) = S(w; r, \lambda)^{-1} E(w), \tag{4.1}$$

where $S(w; r, \lambda)$ is a function depending on a weight vector w and on the local minimizer r of E; λ is a relaxation parameter. Assuming that there exist m local minima $r_1, \dots, r_m \in \mathbb{R}^N$, Relation (4.1) is reformulated as:

$$F(w) = S(w; r_1, \lambda_1)^{-1} \cdots S(w; r_m, \lambda_m)^{-1} E(w). \tag{4.2}$$

Our goal is to find a "proper" $S(\cdot)$ such that $F(w)$ will not obtain a minimum at $r_i, i = 1, \dots, m$, while keeping all other minima of E locally "unchanged". In other words, we have to construct functions S that provide F with the property that any sequence of weights converging to r_i (a local minimizer of E) will not produce a minimum of F at $w = r_i$. In addition, this function F will retain all other minima of E. We call this property *deflection property.*

The following function:

$$S(w; r_i, \lambda_i) = \tanh\left(\lambda_i \|w - r_i\|\right), \tag{4.3}$$

provides F with the above mentioned deflection property as will be explained in the sequel.

Assuming that a local minimum r_i has been determined, then

$$\lim_{w \to r_i} \frac{E(w)}{\tanh\left(\lambda \|w - r_i\|\right)} = +\infty, \tag{4.4}$$

which means that r_i is no longer a local minimizer of F. Moreover, it is easily verified that for $\|w - r_i\| \geq \varepsilon$, where ε is a small positive constant, holds that :

$$\lim_{\lambda \to +\infty} F(w) = \lim_{\lambda \to +\infty} \frac{E(w)}{\tanh\left(\lambda \|w - r_i\|\right)} = E(w), \tag{4.5}$$

since the denominator tends to unity. This means that the error function remains unchanged in the whole weightspace.

However, for an arbitrary value of λ there is a small neighborhood $\mathcal{R}(r, \rho)$ with center r and radius ρ, with $\rho \propto \lambda^{-1}$, that for any $x \in \mathcal{R}(r, \rho)$ holds that $F(x) > E(x)$. To be more specific, when the value of λ is small (say $\lambda < 1$) the denominator in (4.5) becomes one for w "far" from r. Thus, the deflection procedure affects a large neighborhood around r in the weightspace. On the other hand, when the value of λ is large, new local minima is possible to be created near the computed minimum r (like a mexican hat). These minima have function values greater than $F(r)$ and can be avoided easily by taking a proper stepsize or by changing the value of λ.

At this point it is instructive to provide a simple example in order to illustrate the improvement achieved by incorporating the new technique in the BP method. The problem consists of a neuron with two weights to be learned, on a set of 8, two-dimensional patterns. The two weights, w_1, w_2 are initialized from uniform distributions in the intervals $(-3, 3)$ and $(-7.5, 7.5)$, respectively. The error surface is shown in Figure 1. The global minimum is located at the center and there are two valleys that lead to local minima.

In Figure 2 we illustrate the weight trajectory when the initial weight vector leads BP to a local minimum. In Figures 3 and 4 we illustrate the deflected trajectory of weights drawn on the contour lines of the error function E and on the contour lines of the function F, respectively.

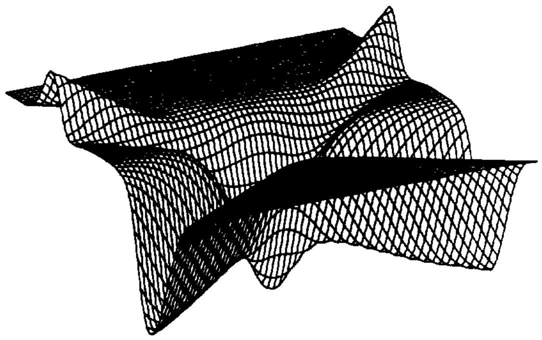
Figure 1: The error surface

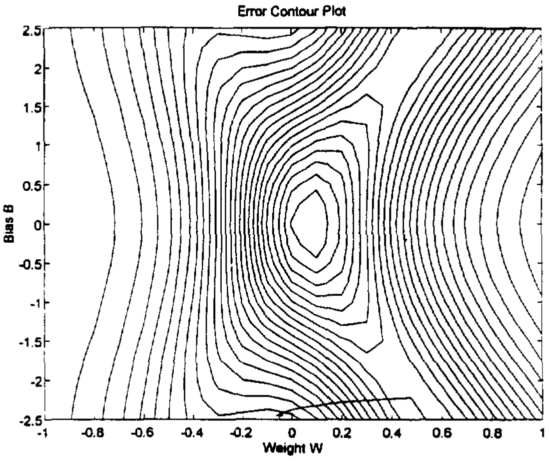

Figure 2: Weight trajectory when the initial weight vector leads BP to a local minimum

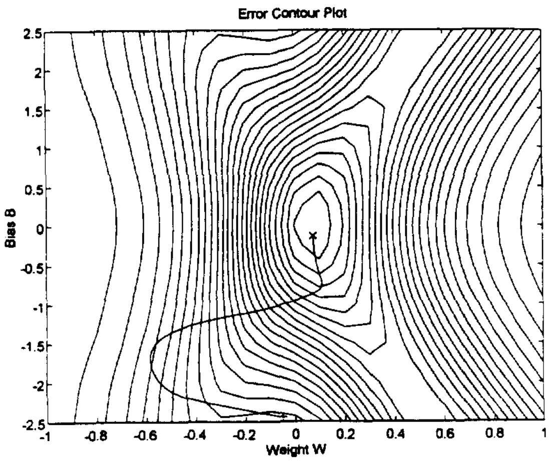

Figure 3: Deflected trajectory of weights drawn on the contour lines of the error function

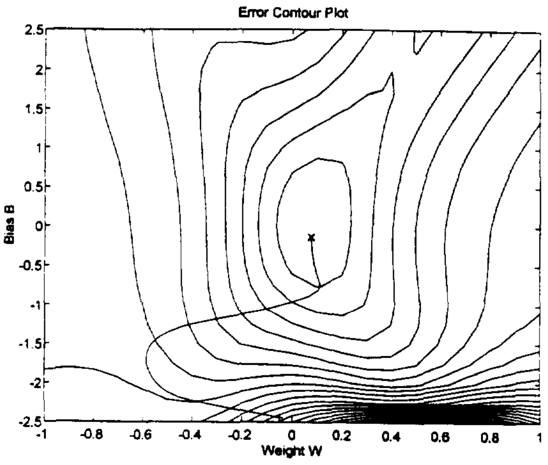

Figure 4: Deflected trajectory of weights drawn on the contour lines of the error function subject to deflection

5. EXPERIMENTAL EVALUATION AND CONCLUDING REMARKS

Several experiments have been performed to evaluate the deflection procedure (DEflactive TRajectory Algorithm - DETRA) and compare its performance with the batch versions of BP, SA1 and BPSA. The algorithms have been tested using the same initial weight vectors chosen from the uniform distribution in the interval $(-1, +1)$. BP and SA1 termination condition has been $E \leq 0.04$. Note also that, BPSA and DETRA updated weights using BP until convergence to a global or local minimum. Global convergence has been obtained when $E \leq 0.04$, while local convergence has been considered when the stopping condition $|\nabla E(w^k)| \leq 10^{-3}$ has been met and w^k has been taken as a local minimum r_i of the error function E.

1. Exclusive-OR classification problem: classification of the four XOR patterns in one of two classes, $\{0,1\}$ using a 2-2-1 FNN is a classical test problem [1,7,5]. The XOR problem is sensitive to initial weights and presents a multitude of local minima [15]. The stepsize is taken equal to 1.5 and the heuristics for SA1 and BPSA are tuned to $n = 0.3$ and $d = 0.002$. In all instances, 100 simulations have been run and the results are summarized in Table 1.

Training	Exclusive-OR Problem			Parity Problem		
Method	Success	Mean	Std	Success	Mean	Std
BP	42 %	144.1	112.6	22 %	8818.5	6422.9
SA1	43 %	424.2	420.8	11 %	8051.4	4207.1
BPSA	65 %	1661.9	2775.7	33 %	26340.0	34334.0
DETRA	100 %	575.1	387.3	100 %	14409.0	5575.9

Table 1: Comparative results in terms of function evaluations for the test problems

2. The four bit parity problem [1]*:* a 4–4–1 FNN receives 16, 4–dimensional binary input patterns and must output a "1" if the inputs have an odd number of ones and "0" if the inputs have an even number of ones. This is a very difficult problem for an FNN because the network must determine the proper parity (the value at the output) for input patterns which differ only by Hamming distance 1. It is well known that network's weightspace contains "bad" local minima. The stepsize has been taken equal to 0.5 and the heuristics for SA1 and BPSA have been tuned to $n = 0.1$ and $d = 0.00025$. In all instances, 100 simulations have been run and the results are summarized in Table 1.

The results of Table 1 suggest that DETRA escapes local minima and converges to the global minimum in all cases. A consideration that is worth mentioning is that the number of function evaluations in BPSA and DETRA contains the additional evaluations required for BP to satisfy the local minima stopping condition.

In conclusion, a simple procedure for the alleviation of the problem of local minima in BP training has been presented. This procedure can be incorporated in any training algorithm and allows escaping the influence of local minima. Preliminary results on two notorious for their local minima problems are promising. We currently investigate techniques in order to find a proper relaxation parameter λ for each case.

REFERENCES

1. RUMELHART D. E., HINTON G. E. & WILLIAMS R. J., Learning internal representations by error propagation, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1,* Edited by D .E. RUMELHART and J. L. McCLELLAND, pp. 318-362, MIT Press, (1986).
2. GORI M. & TESI A., On the problem of local minima in backpropagation, *IEEE Trans. Pattern Analysis and*

Machine Intelligence, **14**, 76–85, (1992).
3. ORTEGA J. M. & RHEINBOLDT W. C. , *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, (1970).
4. VOGL T. P., MANGIS J. K., RIGLER A. K., ZINK W. T. & ALKON D. L., Accelerating the convergence of the back-propagation method, *Biological Cybernetics*, **59**, 257–263, (1988).
5. MAGOULAS G. D., VRAHATIS M. N. & ANDROULAKIS G. S., Effective backpropagation training with variable stepsize, *Neural Networks*, accepted for publication, (1996).
6. MÖLLER M. F., A scaled conjugate gradient algorithm for fast supervised learning, *Neural Networks*, **6**, 525–533, (1993).
7. VAN DER SMAGT P. P., Minimisation methods for training feedforward neural networks, *Neural Networks*, **7**, 1–11, (1994).
8. MAGOULAS G. D., VRAHATIS M. N., GRAPSA T. N. & ANDROULAKIS G. S., Neural network supervised training based on a dimension reducing method, *Annals of Mathematics and Artificial Intelligence*, in press, (1996).
9. LEE Y., OH S. H. & KIM M., An analysis of premature saturation in backpropagation learning, *Neural Networks*, **6**, 719–728, (1993).
10. YU X.-H. & CHEN G.-A., On the local minima free condition of backpropagation learning, *IEEE Trans. Neural Networks*, **6**, 1300–1303, (1995).
11. KIRKPATRICK S. , GELATT C. D. Jr. & VECCHI M. P., Optimization by simulated annealing, *Science*, **220**, 671–680, (1983).
12. CORANA A., MARCHESI M., MARTINI C. & RIDELLA S., Minimizing multimodal functions of continuous variables with the "Simulated Annealing" algorithm, *ACM Trans. Math. Soft.*, **13**, 262–280, (1987).
13. WESLSTEAD S. T., *Neural network and fuzzy logic applications in C/C++*, Wiley, (1994).
14. BURTON R. M. Jr. & MPITSOS G. J., Event-dependent control of noise enhances learning in neural networks, *Neural Networks*, **5**, 627–637, (1992).
15. BLUM E. K., Approximation of boolean functions by sigmoidal networks: Part I: XOR and other two variable functions, *Neural Computation*, **1**, 532–540, (1989).