PERGAMON

Nonlinear Analysis 47 (2001) 3431–3436

Nonlinear Analysis

www.elsevier.nl/locate/na

Learning in multilayer perceptrons using global optimization strategies

V.P. Plagianakos [a,c], G.D. Magoulas [b,c], M.N. Vrahatis [a,c]

[a] *Department of Mathematics, University of Patras, GR-26110 Patras, Greece*

[b] *Department of Information Systems and Computing, Brunel University, Uxbridge UB8 3PH, United Kingdom*

[c] *University of Patras Artificial Intelligence Research Center (UPAIRC)*

Abstract

Learning algorithms for multilayer perceptrons are usually based on local minimization methods that can be often trapped in a local minimum of the error function. In this work, the use of global optimization strategies for training multilayer perceptrons is investigated. These methods are expected to lead to "optimal" or "near-optimal" weight configurations by allowing the network to escape local minima during training. The paper reviews the fundamentals of a recently proposed deflection procedure, simulated annealing, genetic and evolutionary algorithms, and introduces a new differential evolution strategy. Simulations and comparisons are presented.

1 Introduction

MultiLayer Perceptrons (MLPs) have been widely used in many application areas and have shown their strengths in solving hard problems in artificial intelligence. The training of an MLP is achieved by the incremental adaptation of connection weights that propagate information between simple processing units called *artificial neurons* and aims at minimizing the multi-variable error function of the network. Below, a unified notation for the weights is adopted aiming at simplifying the formulation of the equations; thus, for an MLP w denotes a column weight vector with components $w_1, w_2, \ldots, w_n$ and is defined in $\mathbb{R}^n$; E represents the batch error measure defined as the sum-of-squared-differences error function over the entire training set; $\nabla E(w)$ defines the gradient vector of the error function E at w.

0362-546X/01/$ - see front matter
PII: S0362-546X(01)00459-X

Commonly used training methods are gradient–based, such as the Back–Propagation (BP), which minimizes $E(w)$ using the steepest descent with constant stepsize [5]:

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

The optimal value of the stepsize μ depends on the shape of the N–dimensional error function and the gradient, ∇E, is computed by applying the chain rule on the layers of the MLP. Several modifications of this scheme based on local minimization methods have been proposed in the literature. However, these algorithms are often trapped in a local minimum of the error function. Convergence to a local minimum prevents the MLP from learning the entire training set and results in inferior network performance, or possibly to premature convergence. 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 sometimes results in a nonconvex error function [2]. The insufficient number of hidden nodes, as well as improper initial weight settings can cause convergence to a local minimum. 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 [2,8]. However, conditions such as the linear separability of the patterns and the pyramidal structure of the MLP [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 simple problems.

This papers presents methods that alleviate the problem of occasional convergence to local minima in MLP training by using Global Optimization (GO) strategies. The remaining of this paper is organized as follows. In the next section a recently proposed deflection procedure is briefly presented. The fundamentals of simulated annealing are presented in section 3, while genetic and evolutionary algorithms are reviewed in section 4 and 5 respectively. Section 6 presents simulation experiments and a discussion of the results obtained.

2 The deflection procedure

Assuming that m local minima, $r_1, \ldots, r_m \in \mathbb{R}^N$, exist, the *deflection* procedure suggests to formulate the function:

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

where $S(w; r_i, \lambda_i)$ is a function depending on the weight vector w and on the local minimizer r_i of E; λ_i, $i = 1, \ldots, m$, is a set of relaxation parameters. The goal is to find a "proper" function $S(\cdot)$, such that $F(w)$ will not have a minimum at $r_i, i = 1, \ldots, m$, while all other minima of E remain locally "unchanged". In other words, we have to construct functions S that provide F with the property that any sequence of weights, $\{w^k\}_0^\infty$, converging to r_i will not produce a minimum of F at

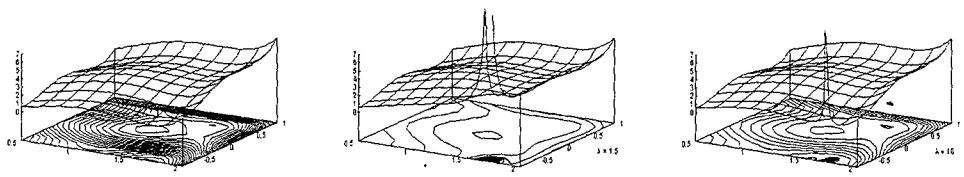

Fig. 1. Applying the deflection procedure to the Six Hump Camel Back test function.

$w = r_i$ and that the function F will retain all other minima of E. The function

$$S(w; r_i, \lambda_i) = \tanh(\lambda_i \|w - r_i\|)$$

provides F with this property, also called *deflection property*, as proved in [4].

As an application example we applied the deflection procedure on the Six Hump Camel Back test function

$$f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4.$$

The plot of the original function is illustrated in Fig. 1 (left). The effect of the deflection procedure for $\lambda = 1.5$ is shown in Fig. 1 (center) and for $\lambda = 10$ in Fig. 1 (right). When the value of λ_i is small (say $\lambda_i < 1$) the deflection procedure affects a large neighborhood around r_i. On the other hand, when the value of λ_i is large new local minima may be created near the minimizer r_i, like a Mexican hat, see Fig. 1 (right). These minima have greater function values than $F(r_i)$ and can be easily avoided by taking a proper stepsize, or by changing the value of λ_i.

The deflection procedure can be incorporated in any algorithm to alleviate convergence to local minima. In the experiments reported below the steepest descent method was equipped with the deflection procedure and the new method was named Steepest Descent with Deflection (SDD).

3 The simulated annealing

Simulated Annealing (SA) [5] refers to the process in which random noise in a system is systematically decreased at a constant rate so as to enhance the systems's response. The SA is based on random evaluations of the error function, in such a way that transitions out of a local minimum are possible. Unfortunately, the performance of the SA as has been observed on typical MLP training problems is not the appropriate one. SA is characterized by the need for a greater number of function evaluations 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 E 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 [7].

Burton and Mpitsos [1] proposed a modification of the SA that benefits from derivative–related information:

$$w^{k+1} = w^k - \mu \nabla E(w^k) + nc2^{-dk}, \tag{2}$$

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. This modified method, named BM, was used in our experiments as well as an alternative scheme: we start by updating the weights using BP (see Relation (1)); if BP converges to a local minimum, then we switch to BM. The combined scheme (1)–(2) is named BPBM.

4 Genetic algorithms

Genetic Algorithms (GAs) are simple and robust search algorithms based on the mechanics of natural selection and natural genetics [5]. Briefly, a simple GA processes a finite population of fixed length binary strings called *genes*. GAs have two basic operators, namely: *crossover* of genes and *mutation* for random change of genes. Another operator associated with each of these operators is *selection*, which produces survival of the fittest in the GA. The crossover operator explores different structures by exchanging genes between two strings at a crossover position. The mutation operation introduces diversity in the population by altering a bit position of the selected string and is used to escape the local minima in the weight space. The combined action of mutation and crossover is responsible for much of the effectiveness of GAs search, and allows GAs to act as a parallel, noise–tolerant hill–climbing algorithms, which search the whole weight space. The "Genetic Algorithm for Optimization Toolbox (GAOT)" [3] has been used for the experiments reported below. GAOT's default crossover and mutation schemes, and a real–valued encoding of the MLP's weights were employed.

5 Evolutionary algorithms

Evolutionary Algorithms (EAs) are global search methods that mimic the metaphor of natural biological evolution. EAs operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics.

In our experiments we have used *Differential Evolution* (DE) strategies [5] to train MLPs, because they can handle non differentiable, nonlinear and multimodal objective functions efficiently, and require few easily chosen control parameters. Also, empirical results support the argument that DE strategies possess good convergence properties and outperform other EAs [6]. In our case, we start with a specific number (*NP*) of weight vectors, as an initial weight population, and evolve them

over time; NP is fixed throughout the training process and the weight population is initialized randomly following a uniform probability distribution. For each weight vector w_g^i, $i = 1, \ldots, NP$, where g denotes the current generation, a new vector v_{g+1}^i (mutant vector) is generated according to one of the relations (3)–(8), described below, where w_g^{best} is the best member of the previous generation, $\xi > 0$ is a real parameter called mutation constant, which controls the amplification of the difference between two weight vectors, and $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, \ldots, i-1, i+1, \ldots, NP\}$ are random integers mutually different and different from the running index i.

$$\text{Algorithm DE1}: \quad v_{g+1}^i = w_g^{r_1} + \xi \left(w_g^{r_1} - w_g^{r_2}\right) \tag{3}$$

$$\text{Algorithm DE2}: \quad v_{g+1}^i = w_g^{\text{best}} + \xi \left(w_g^{r_1} - w_g^{r_2}\right) \tag{4}$$

$$\text{Algorithm DE3}: \quad v_{g+1}^i = w_g^{r_1} + \xi \left(w_g^{r_2} - w_g^{r_3}\right) \tag{5}$$

$$\text{Algorithm DE4}: \quad v_{g+1}^i = w_g^i + \xi \left(w_g^{\text{best}} - w_g^i\right) + \xi \left(w_g^{r_1} - w_g^{r_2}\right) \tag{6}$$

$$\text{Algorithm DE5}: \quad v_{g+1}^i = w_g^{\text{best}} + \xi \left(w_g^{r_1} - w_g^{r_2}\right) + \xi \left(w_g^{r_3} - w_g^{r_4}\right) \tag{7}$$

$$\text{Algorithm DE6}: \quad v_{g+1}^i = w_g^{r_1} + \xi \left(w_g^{r_2} - w_g^{r_3}\right) + \xi \left(w_g^{r_4} - w_g^{r_5}\right). \tag{8}$$

To increase further the diversity of the mutant weight vector, crossover is applied: for each component j ($j = 1, 2, \ldots, N$) of the mutant weight vector v_{g+1}^i, we randomly choose a real number r from the interval $[0, 1]$, and compare this number with ρ (crossover constant). If $r \leq \rho$, we substitute the j-th component of the trial vector u_{g+1}^i with the j–th component of v_{g+1}^i; otherwise, with the j-th component of the target vector w_{g+1}^i.

6 Simulation results and discussion

Experiments have been performed to evaluate the global search methods and compare their performance in two notorious for their local minima problems. Initial weight vectors chosen from a uniform distribution in the interval $(-1, +1)$ were used and global convergence was achieved when $E \leq 0.04$. Local convergence was considered when $\|\nabla E(w^k)\| \leq 10^{-3}$, and w^k was taken as a local minimizer r_i of E. The fixed values $\xi = 0.5$ and $\rho = 0.7$ were used for the mutation and the crossover, NP was twice the dimension of the problem considered, and 100 simulations were run for each case.

The *Exclusive-OR classification problem* concerns the classification of four patterns in one of two classes using a 2–2–1 MLP. It is a classical test problem, sensitive to initial weights, and presents a multitude of local minima. The stepsize was equal to 1.5 and the heuristics for BM and BPBM were $n = 0.3$.

In the *three bit parity problem*, a 3–3–1 MLP receives 8 binary patterns and must output a "1", for inputs that have an odd number of 1s, or "0", for inputs that have an even number of 1s. The stepsize was equal to 0.5, and the heuristics for BM and BPBM were tuned to $n = 0.1$ and $d = 0.00025$.

The results of the experiments, exhibited in Table 1, suggest that the proposed DE strategies and the SDD method provide a better probability of success than the BP. Especially the SDD method exhibits a very good average performance: escapes local minima and converges to the global minimum in all cases. It is worth mentioning that

Table 1
Comparative results

Training	*XOR Problem*			*Parity Problem*		
Method	Success	Mean	std.dev.	Success	Mean	std.dev.
BP	42%	144.1	112.6	91%	932.0	1320.8
BM	43%	424.2	420.8	22%	805.4	2103.1
BPBM	65%	1661.9	2775.7	66%	2634.0	6866.8
GA	95%	422.3	397.5	73%	1091.5	766.2
DE1	75%	192.9	124.7	91%	622.6	522.1
DE2	80%	284.9	216.2	61%	1994.1	657.6
DE3	97%	583.9	256.3	99%	896.3	450.6
DE4	98%	706.1	343.7	98%	1060.2	716.6
DE5	85%	300.5	250.2	26%	2112.0	644.9
DE6	93%	482.9	264.9	44%	2062.5	794.8
SDD	100%	575.1	387.3	100%	760.0	696.4

the mean number of function evaluations in BPBM and SDD contains the additional evaluations required for the BP to satisfy the local minima stopping condition. The GA and the DE3 and DE4 are promising and effective in the XOR problem, even when compared with other methods that require the gradient of the error function, in addition to the error function values. Note, however, that the performance of the GA in the parity problem is not very good compared to the fined tuned BP; it is possible to alleviate this situation by tuning GAOT's crossover and mutation schemes. In conclusion, global search methods provide techniques that alleviate the problem of occasional convergence to local minima in MLP training. Escaping from local minima is not always possible, however these methods exhibit a better chance in locating appropriate solutions.

References

[1] M. Burton Jr. and G.J. Mpitsos, Event dependent control of noise enhances learning in neural networks, Neural Networks, 5 (1992) 627.

[2] M. Gori and A. Tesi, On the problem of local minima in backpropagation, IEEE Tr. Pattern Analysis and Machine Intelligence, 14 (1992) 76.

[3] C. Houck, J. Joines, and M. Kay, A Genetic Algorithm for Function Optimization: A Matlab Implementation, NCSU–IE TR, 95–09, 1995.

[4] G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis, On the alleviation of local minima in backpropagation, Nonlinear Analysis: Theory, Methods and Applications, 30 (1997) 4545.

[5] Z. Michalewicz and D.B. Fogel, How to solve it: Modern Heuristics, Springer, 2000.

[6] V.P. Plagianakos and M.N. Vrahatis, Training Neural Networks with Threshold Activation Functions and Constrained Integer Weights, Proc. of the IEEE Int. J. Conf. on Neural Networks (IJCNN'2000), 2000.

[7] S.T. Weslstead, Neural network and fuzzy logic applications in C/C++, Wiley, 1994.

[8] X.-H. Yu, G.-A. Chen, On the local minima free condition of backpropagation learning, IEEE Tr. Neural Networks, 6 (1995) 1300.