# Global Search Methods for Neural Network Training

V.P. PLAGIANAKOS[1], G.D. MAGOULAS[2], G.S. ANDROULAKIS[1], M.N. VRAHATIS[1]
[1]Department of Mathematics, University of Patras and
University of Patras Artificial Intelligence Research Center–UPAIRC,
GR-261.10 Patras, GREECE.
e-mail: {vpp,gsa,vrahatis}@math.upatras.gr
URL: www.math.upatras.gr/~vrahatis
[2]Department of Informatics, University of Athens,
GR-157.84 Athens, GREECE and
University of Patras Artificial Intelligence Research Center–UPAIRC,
e-mail: magoulas@di.uoa.gr

*Abstract:* - In many cases the supervised neural network training using a backpropagation based learning rule can be trapped in a local minimum of the error function. These training algorithms are local minimization methods and have no mechanism that allows them to escape the influence of a local minimum. 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. In this work global search methods for feedforward neural network batch training are 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 simulated annealing, genetic algorithms as well as some recently proposed deflection procedures. Simulations and comparisons are presented.

*Keywords and phrases*: Alleviation of local minima, simulated annealing, genetic algorithms, deflection procedure, feedforward neural networks, batch training.

## 1 Introduction

In neural network batch training the objective is to minimize a cost function defined as the multi–variable error function of the network. More specifically, 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:

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

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 $\sigma$ 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, $\theta_j^l$ 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:

$$w^{k+1} = w^k + \Delta w^k, \quad k = 0, 1, \ldots,$$

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

The commonly used training methods are gradient based algorithms such as the popular back–propagation (BP) algorithm [17]. It is well known that the BP algorithm leads to slow training and often yields suboptimal solutions [4].

This contribution presents techniques that alleviate the problem of occasional convergence to local minima in BP training. Global search methods for feedforward neural network batch training are 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 is organized as follows. In section 2 the BP training algorithm is reviewed and its local minima problem is discussed. In section 3 a recently proposed deflection procedure [9] is briefly presented. The fundamentals of simulated annealing [2, 3, 6] are presented in section 4, while genetic and evolutionary algorithms [11] are reviewed in section 5. Section 6 summarizes and discusses our results as well as presents simulations and comparisons with the standard backpropagation algorithm.

## 2 Back–Propagation training and local minima

The BP procedure minimizes the error function $E(w)$ using the steepest descent [13] with constant stepsize $\mu$:

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

The optimal value of the stepsize $\mu$ depends on the shape of the $N$–dimensional error function. The gradient, $\nabla E$, is computed by applying the chain rule on the layers of the FNN (see [17]).

Attempts to speed up back–propagation training have been made by dynamically adapting the stepsize $\mu$ during training [8, 20], or by using second derivative related information [10, 12, 19]. However, these BP–like training algorithms are based on local minimization methods and they have no mechanism 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 or possibly to premature convergence.

It is well known that the supervised training using a BP based learning rule can be trapped in a local minimum of the error function. 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 non-convex error function [4]. The insufficient number of hidden nodes as well as improper initial weight settings can cause convergence to a local minimum, which prevents the network from learning the entire training set and results in inferior network performance. Gradient-based backpropagation training algorithms are local minimization methods and have no mechanism that allows them to escape the influence of a local minimum.

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 [4, 7, 22]. However, conditions such as the linear separability of the patterns and the pyramidal structure of the FNN [4] 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.

## 3 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 converging to a minimizer of $E$. Assuming that this sequence converges to a local minimum $r \in \mathbb{R}^N$, we formulate the following function:

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

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

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

Our goal is to find a "proper" $S(\cdot)$ such that $F(w)$ will not obtain a minimum at $r_i, i = 1, \ldots, m$, 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* [9].

The following function:

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

provides $F$ with the above mentioned deflection property, as it will be explained in the following.

Assuming that a local minimum $r_i$ has been determined, then

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

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 $\varepsilon$ is a small positive constant, it holds that:

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

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

However, for an arbitrary value of $\lambda$ there is a small neighborhood $\mathcal{R}(r, \rho)$ with center $r$ and radius $\rho$, with $\rho \propto \lambda^{-1}$, that for any $x \in \mathcal{R}(r, \rho)$ it holds that $F(x) > E(x)$. To be more specific, when the value of $\lambda$ is small (say $\lambda < 1$) the denominator in the above relation becomes one for $w$ "far" from $r$. Thus, the deflection procedure affects a large neighborhood around $r$ in the weight space. On the other hand, when the value of $\lambda$ 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 $\lambda$. We currently investigate techniques in order to find a proper relaxation parameter $\lambda$ for each case.

The above procedure is named DETRA (DEflactive TRajectory Algorithm) and it can be incorporated in any training algorithm.

# 4 The procedure of simulated annealing

*Simulated Annealing* (SA) [6] 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 system.

In the numerical optimization framework, SA 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. In particular, the method is able to discriminate between "gross behavior" of the function and finer "wrinkles". First, it reaches an area in the function domain where a global minimum should be present, following the gross behavior irrespectively of small local minima found on the way. It then develops finer details, finding a good, near–optimal local minimum, if not the global minimum itself.

The performance of the SA [3], 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 [21].

In [2] 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.* [2]. 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.

# 5 Genetic Algorithms

*Genetic Algorithms* (GA) are simple and robust search algorithms based on the mechanics of natural selection and natural genetics. The mathematical framework of GAs was developed in the 1960s and is presented in Holland's pioneering book [5].

GAs have been used primarily in optimization and machine learning problems.

A simple GA processes a finite population of fixed length binary string called *genes*. GAs have three basic operators, namely: *reproduction* of solutions based on their fitness, *crossover* of genes, and *mutation* for random change of genes. Another operator associated with each of these three operators is the *selection* operator, which produces survival of the fittest in the GA. Reproduction directs the search toward the best existing but does not create any new strings, the crossover operator explores different structures by exchanging genes between two strings at a crossover position, and mutation introduces diversity in the population by altering a bit position of the selected string. The mutation operation is used to escape the local minima in the weight space. The combined action of reproduction and crossover is responsible for much of the effectiveness of GA's search, while reproduction and mutation combine to form a parallel, noise–tolerant hill–climbing algorithm.

GAs can be used to train neural networks. The main advantage of these algorithms is that they search the whole weight space. To this approach, instead of GAs, we utilize *Differential Evolution* (DE) strategies [18], since DEs handle non differentiable, nonlinear and multimodal objective functions more efficiently. To fulfill this requirement, DEs have been designed as stochastic parallel direct search methods, which utilize concepts borrowed from the broad class of evolutionary algorithms, but require few easily chosen control parameters. Experimental results have shown that DEs have good convergence properties and outperform other evolutionary algorithms [15, 16].

In order to apply DEs to neural network training we start with a specific number ($NP$) of $N$-dimensional weight vectors, as an initial weight population, and evolve them over time. $NP$ is fixed throughout the training process. The weight population is initialized randomly following a uniform probability distribution.

At each iteration, called *generation*, new weight vectors are generated by the combination of weight vectors randomly chosen from the population. This operation is called *mutation*. The outcoming weight vectors are then mixed with another predetermined weight vector – the *target* weight vector – and this operation is called *crossover*. This operation yields the so-called *trial* weight vector. The trial vector is accepted for the next generation if and only if it reduces the value of the error function $E$. This last operation is called *selection*.

We now briefly review the two basic DE operators used for FNN training. The first DE operator, we consider, is mutation. Specifically, 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 following relations:

$$v_{g+1}^i = w_g^{r_1} + \xi \left( w_g^{r_1} - w_g^{r_2} \right) \quad (1)$$

$$v_{g+1}^i = w_g^{\text{best}} + \xi \left( w_g^{r_1} - w_g^{r_2} \right) \quad (2)$$

$$v_{g+1}^i = w_g^{r_1} + \xi \left( w_g^{r_2} - w_g^{r_3} \right) \quad (3)$$

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

$$v_{g+1}^i = w_g^{\text{best}} + \xi \left( w_g^{r_1} - w_g^{r_2} \right) + \xi \left( w_g^{r_3} - w_g^{r_4} \right) \quad (5)$$

$$v_{g+1}^i = w_g^{r_1} + \xi \left( w_g^{r_2} - w_g^{r_3} \right) + \xi \left( w_g^{r_4} - w_g^{r_5} \right) \quad (6)$$

where $w_g^{\text{best}}$ 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 $i$.

Relation (1) has been introduced as crossover operator for genetic algorithms [11] and is similar to relations (2) and (3). The remaining relations are modifications which can be obtained by the combination of (1), (2) and (3). It is clear that more such relations can be generated using the above ones as building blocks. In recent works [14, 15, 16], we have shown that the above relations can efficiently be used to train FNNs with arbitrary integer weights as well.

To increase further the diversity of the mutant weight vector, the crossover operator is applied. Specifically, 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 $[0, 1]$. Then, we compare this number with $\rho$ (crossover constant), and if $r \leq \rho$, we select as the $j$-th component of the trial vector $u_{g+1}^i$, the corresponding component $j$ of the mutant vector $v_{g+1}^i$. Otherwise, we pick the $j$-th component of the target vector $w_{g+1}^i$.

## 6 Simulation results and discussion

Several experiments have been performed to evaluate the training methods mentioned in the previous sections and compare their performance. Below, we exhibit preliminary results on two notorious for their local minima problems. 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 is obtained. Global convergence has been achieved 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 $E$.

We call DE1 the algorithm that uses relation (1) as mutation operator, DE2 the algorithm that uses relation (2), and so on. We note here that a key feature of the DE algorithms is that *only* error function values are needed. No gradient information is required, so there is no need of backward passes. We made no effort to tune the mutation and crossover parameters, $\xi$ and $\rho$ respectively. We have used the fixed values $\xi = 0.5$ and $\rho = 0.7$, instead. The weight population size $NP$ has been chosen to be twice the dimension of the problem, i.e. $NP = 2N$, for all the simulations considered. Some experimental results have shown that a good choice for $NP$ is $2N \leq NP \leq 4N$. It is obvious that the exploitation of the weight space is more effective for large values of $NP$, but sometimes more error function evaluations are required. On the other hand, small values of $NP$ make the algorithm inefficient and more generations are required in order to converge to the minimum.

Problem 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 [17, 19]. The XOR problem is sensitive to initial weights and presents a multitude of local minima [1]. 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.

Problem 2. *The three bit parity problem* [17]*:* a 3–3–1 FNN receives 8, 3–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 the network's weight space 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 run and the results are summarized in Table 1.

| Training Method | XOR Problem | | | Parity Problem | | |
|---|---|---|---|---|---|---|
| | Succ. | Mean | s.d. | Succ. | Mean | s.d. |
| BP | 42% | 144.1 | 112.6 | 91% | 932.0 | 1320.8 |
| SA1 | 43% | 424.2 | 420.8 | 22% | 805.4 | 2103.1 |
| BPSA | 65% | 1661.9 | 2775.7 | 66% | 2634.0 | 6866.8 |
| 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 |
| DETRA | 100% | 575.1 | 387.3 | 100% | 760.0 | 696.4 |

Table 1: Comparative results

The results of Table 1 suggest that combination of local and global search methods like BPSA and DETRA provide a better probability of success than the BP. Note that the performance of SA1 is not the appropriate one although derivative related information has been used. On the other hand, 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.

The results indicate that the algorithms of the DE class are promising and effective, even when compared with other methods that require the gradient of the error function, in addition to the error function values. For example, DE3 and DE4 have exhibited very good performance for the test problems considered. On the other hand, there have been cases where a discrepancy has been found in DE's behavior; see for example DE5 and DE6. For a discussion on the generalization capabilities of the networks generated by the DE algorithms see [15, 16].

In conclusion, global search methods provide techniques that alleviate the problem of occasional convergence to local minima in feedforward neural

network training. Escaping from local minima is not always possible, however these methods exhibit a better chance in locating appropriate solutions. Preliminary results on two notorious for their local minima problems are promising.

## References


[1] E.K. Blum, "Approximation of Boolean functions by sigmoidal networks: Part I: XOR and other two variable functions", Neural Computation, vol.1, 1989, pp.532–540.

[2] M. Burton Jr., G.J. Mpitsos, "Event dependent control of noise enhances learning in neural networks", Neural Networks, vol.5, 1992, pp.627–637.

[3] A. Corana, M. Marchesi, C. Martini, S. Ridella, "Minimizing multimodal functions of continuous variables with the Simulated Annealing algorithm", ACM Trans. Math. Soft., vol.13, 1987, pp.262–280.

[4] M. Gori, A. Tesi, "On the problem of local minima in backpropagation", IEEE Trans. Pattern Analysis and Machine Intelligence, vol.14, 1992, pp.76–85.

[5] J.H. Holland, "Adaptation in Neural and Artificial Systems", University of Michigan Press, 1975.

[6] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, "Optimization by simulated annealing", Science, vol.220, 1983, pp.671–680.

[7] Y. Lee, S.H. Oh, M. Kim, "An analysis of premature saturation in backpropagation learning", Neural Networks, vol.6, 1993, pp.719–728.

[8] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, "Effective backpropagation training with variable stepsize", Neural Networks, vol.10, No.1, 1997, pp.69–82.

[9] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, "On the alleviation of local minima in backpropagation", Nonlinear Analysis, Theory, Methods and Applications, vol.30, 1997, pp.4545–4550.

[10] G.D. Magoulas, M.N. Vrahatis, T.N. Grapsa, G.S. Androulakis, "Neural network supervised training based on a dimension reducing method", Mathematics of Neural Networks, Models, Algorithms and Applications, S.W. Ellacott, J.C. Mason, I.J. Anderson Eds., Kluwer Academic Publishers, Boston, Chapter 41, 1997, pp.245–249.

[11] Z. Michalewicz, "Genetic algorithms + data structures = evolution programs", Springer, 1996.

[12] M.F. Möller, "A scaled conjugate gradient algorithm for fast supervised learning", Neural Networks, vol.6, 1993, pp.525–533.

[13] J.M. Ortega, W.C. Rheinboldt, "Iterative Solution of Nonlinear Equations in Several Variables", Academic Press, New York, 1970.

[14] V.P. Plagianakos, D.G. Sotiropoulos, M.N. Vrahatis, "Integer weight training by differential evolution algorithms", Recent Advances in circuits and systems, N.E. Mastorakis Ed., World Scientific, 1998, pp.327–331.

[15] V.P. Plagianakos, M.N. Vrahatis, "Training neural networks with 3–bit integer weights", Proceedings of Genetic and Evolutionary Computation Conference, to appear, 1999.

[16] V.P. Plagianakos, M.N. Vrahatis, "Neural network training with constrained integer weights", Proceedings of Congress on Evolutionary Computation, to appear, 1999.

[17] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning internal representations by error propagation", Parallel Distributed Processing: Explorations in the Microstructure of Cognition 1, D.E. Rumelhart, J.L. McClelland Eds., MIT Press, 1986, pp.318–362.

[18] R. Storn, K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", Journal of Global Optimization, vol.11, 1997, pp.341–359.

[19] P.P. Van der Smagt, "Minimisation methods for training feedforward neural networks", Neural Networks, vol.7, 1994, pp.1–11.

[20] T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink, D.L. Alkon, "Accelerating the convergence of the back-propagation method", Biological Cybernetics, vol.59, 1988, pp.257–263.

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

[22] X.-H. Yu, G.-A. Chen, "On the local minima free condition of backpropagation learning", IEEE Trans. Neural Networks, vol.6, 1995, pp.1300–1303.