Neural Processing Letters **16:** 201–210, 2002.

The Minimum Number of Errors in the *N*-Parity and its Solution with an Incremental Neural Network

J. MANUEL TORRES-MORENO[1⋆], JULIO C. AGUILAR[2], and MIRTA B. GORDON[3]
[1]*École Polytechnique de Montréal, Département de Génie informatique, CP 6079 Succ. Centre-ville, H3C3A7 Montréal (Québec) Canada. e-mail: juan-manuel.torres@polymtl.ca*
[2]*Laboratorio Nacional de Informática Avanzada (LANIA), Rébsamen 80-91090 Xalapa, México*
[3]*Laboratoire Leibniz – IMAG (CNRS), 46, Avenue Félix Viallet, 38031 Grenoble Cedex, France*

Abstract. The N-dimensional parity problem is frequently a difficult classification task for Neural Networks. We found an expression for the minimum number of errors v_f as function of N for this problem, performed by a perceptron. We verified this quantity experimentally for $N = 1, \ldots, 15$ using an optimal train perceptron. With a constructive approach we solved the full N-dimensional parity problem using a minimal feedforward neural network with a single hidden layer of $h = N$ units.

Key words. classification tasks, minimerror, monoplan, parity problem, perceptrons, supervised learning

1. Introduction

The Neural Networks Community has studied the N-dimensional parity problem for a long time. In their celebrated book, Minsky and Papert [1] elegantly demonstrated that perceptrons are unable to solve non linearly separable problems, such as the parity of 2 inputs (or their equivalent one, the or-exclusive problem, XOR). The capacity of a simple perceptron is limited, since it is unable to solve problems that are linearly separable [6, 7]. The problem still becomes difficult in small dimensions: $N \leqslant 5$ and it is increased exponentially in function of the number of available patterns. This problem has been attacked by several methods such as Gradient Backpropagation (BP) and its variations. These methods have difficulties even in small dimensions due to the problem of the local minima in those in which the minimization of the cost function may fall. An alternative approach is to use Incremental Neural Networks methods that add units while learning errors exist, following a suitable heuristic, as we show it in Section 4.

⋆Also at ERMETIS and LANCI – Université du Québec (Canada). Author to whom correspondence should be sent.

The N-dimensional parity problem can be formulated as a supervised learning problem with a learning set of P patterns with N binary inputs $\xi_i = \pm 1$, $i = 1, \ldots, N$, and a binary output $\tau = \pm 1$:

$$\mathcal{L} = \{(\xi^\mu, \tau^\mu), \mu = 1, \ldots, P\} \tag{1}$$

The underlying difficulty for this problem is that, in general the N-parity is difficult to solve in high dimensionality because if the minimization of a cost function is used, such as the one typically used in Backpropagation, it is very complicated for the gradient search algorithms to escape from the multiple local minima. This problem has become a classic benchmark for classification algorithms [1], given that it is a highly not linearly separable problem (non LS). The classifier should learn how to discriminate against it if a given pattern belongs to the positive class, $\tau = +1$, or the negative one, $\tau = -1$. The problem is considered exhaustive learning, because all the $P = 2^N$ different patterns examples must be learned. The N-parity's problem becomes quickly complicated due to the neighboring states in input's space (its Hamming distances are $d_H = 1$) with opposite outputs. A solution with binary poids perceptrons has been should in [22].

The input's space for the N-dimensional parity and the separating hyperplanes $\boldsymbol{w}$ are represented for values of $N = 2, 3, 4$, in Figures 1(a), 1(b) and 2.

2. Finding the Minimum Number of Errors

Following a constructive approach, an incremental Neural Network adds hidden units one to one, until it is capable of eliminating learning errors. In N-dimensional parity, the problem is quite difficult for the first unit because it should find the

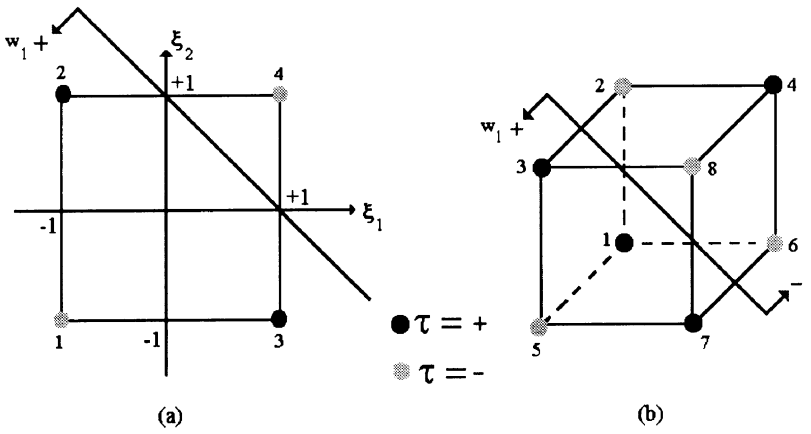

Figure 1. N-dimensional parity: (a) $N = 2$, (b) $N = 3$.

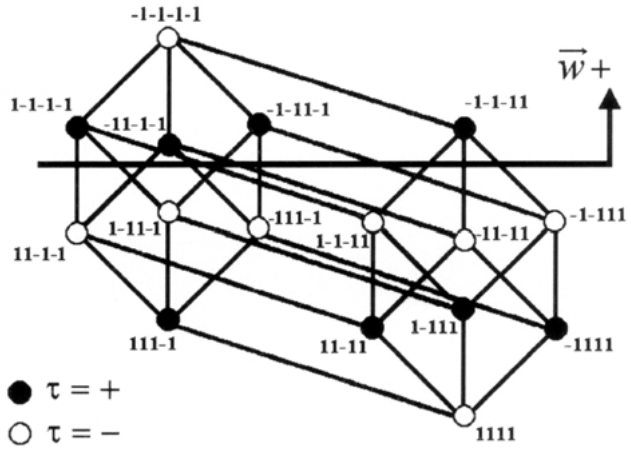

Figure 2. 4-dimensional parity problem with separating hyperplane $\vec{w}$. It misclassifies five patterns.

smaller number of learning errors in a highly intricate *N*-dimensional space. However, the corresponding hyperplane is well located, so this will allow it to find the *minimum number of errors*. Also, thanks to the geometric symmetry of the problem, the rest will be less and less difficult to solve for the subsequent units.

But which is the *minimum number of errors* for the *N*-dimensional parity? To find this number theoretically, first consider Figure 1, which represents the 2-dimensional parity. Vector $\boldsymbol{w}_1$ separates the input's space, where it is observed that patterns $\mu = 2$, 3 and 4 are well classified, whereas the negative class pattern $\mu = 1$ is not well classified. $\boldsymbol{w}_1$ makes an classification error, and it is not possible for any perceptron to make a better classification. For the 3-dimensional parity, consider a 3D input's space. In Figure 1(b), vector $\boldsymbol{w}_1$ classifies the patterns $\mu = \{1, 2, 3, 6, 7, 8\}$ correctly, whereas the patterns $\mu = \{4, 5\}$ are not well classified. Then, two errors are made. Here, it is also observed the symmetrical phenomenon of signs alternation, starting from the position of the separated hyperplane: patterns with $\tau^\mu = -1$ ($\mu = 5$), $\tau^\mu = +1$ ($\mu = 1, 3, 7$), $\tau^\mu = -1$ ($\mu = 2, 6, 8$) and $\tau^\mu = -1$ ($\mu = 4$). A 4-dimensional space is represented in $\mathbb{R}^2$, generating the hypercube shown in Figure 2. There it is possible to separate the patterns of different classes with a hyperplane w to decrease the learning errors, as shown in the same Figure 2. A symmetrical sign's distribution of patterns in the input's space is evident, and it allowed to suspect us a combinatorial behavior.

This observations allow us constructed the Table I, where the class distribution of the hypercube vertices v_k has been defined for the coeficients of the binomial:

$$v_k = \binom{N}{k}; \quad k = 0, 1, \ldots, N \tag{2}$$

This table represents the Pascal's triangle. This alternated class distribution of vertices v_k (pattern class $\tau = -1$ or $\tau = +1$) can be separated for successive hyperplanes.

Table I. Distribution vertices $v_k, k = 0, 1, \ldots, N$, their class τ and the minimum number of errors v_f for the $N = 1, 2, \ldots, 11$-dimensional parity.

N	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_f
2	1	2	1										1
3	1	3	3	1									2
4	1	4	6	4	1								5
5	1	5	10	10	5	1							10
6	1	6	15	20	15	6	1						22
7	1	7	21	35	35	21	7	1					44
8	1	8	28	56	70	56	28	8	1				93
9	1	9	36	84	126	126	84	36	9	1			186
10	1	10	45	120	210	252	210	120	45	10	1		386
11	1	11	55	165	330	462	462	330	165	55	11	1	772
	$\tau = -$	+	−	+	−	+	−	+	−	+	−	+	

If $N = 3$, we have one pattern with class -1 (vertex v_0), three patterns with class $+1$ (vertex v_1), three with class -1 (vertex v_2) and one pattern with class $+1$ (vertex v_3). The separating hyperplane that minimizes the number of errors *should* be located among v_1 and v_2, which generates 2 errors. For $N = 4$, we have one pattern with class -1 (vertex v_0), four with class $+1$ (vertex v_{11}), six patterns with class $-$ (vertex v_2), four with class $+1$ (vertex v_3) and one pattern with class $+1$ (vertex v_4). To minimize the classification errors the separating hyperplane should now be located between v_1 and v_2 or between v_2 and v_3. It generates 5 errors.

The final column v_f in Table I represents the minimum number of errors for the N-dimensional parity made by a perceptron with N inputs. v_f is not a simple addition because the parity of vertices must be considered. From here, a geometrical analysis has shown that:

$$v_f = \begin{cases} v_f(N = 2p) = \sum_{i=1}^{p} \binom{2p}{2p-i+1} & \text{if } N \text{ is even} \\ v_f(N = 2p+1) = 2v_f(2p) & \text{if } N \text{ is odd} \end{cases} \qquad p = 1, 2, 3, \cdots \tag{3}$$

We introduce here the following

THEOREM. *Let* $\mathcal{L} = \{(\boldsymbol{\xi}^\mu, \tau^\mu), \mu = 1, \ldots, P\}$ *a exhaustive learning set of* P *binary patterns* ξ_i^u *with* $P = 2^N$; $i = 1, 2, \ldots, N$; $\tau = \pm 1$ *for the parity problem in a* N*-dimensional input's space. Then, the minimum number of errors* v_f *make by a optimal separating hyperplane is given by:*

$$v_f = 2^{N-1} - \binom{N-1}{m} \tag{4}$$

Proof. Let us consider the class distribution of the N-dimensional parity vertices given for

$$v_k = \binom{N}{k} \tag{5}$$

and let us suppose that the separating hyperplane is placed between v_m and v_{m+1}, guided in such a way that patterns in both v_m and v_{m+1} vertices are well classified. If m is even, we have that the misclassified patterns to the left of the separating hyperplane according to the normal vector are in the vertices $v_1, v_3, \ldots, v_{m-1}$. If m is odd, then the errors will be in $v_0, v_2, \ldots, v_{m-1}$ vertices. We call η_1 *first half of number of errors*, and we have:

$$\eta_1 = \begin{cases} \sum_{k=0}^{\frac{m}{2}-1} \binom{N}{2k+1} & \text{if } m \text{ is even} \\ \sum_{k=0}^{\frac{m-1}{2}} \binom{N}{2k} & \text{if } m \text{ is odd} \end{cases} \tag{6}$$

$$= \sum_{k=0}^{m-1} \binom{N-1}{k} \tag{7}$$

Similarly, we can count the errors η_2 in the right side of the separating hyperplane:

$$\eta_2 = \begin{cases} \sum_{k=2}^{\frac{N-m}{2}} \binom{N}{m+k} & \text{if } N-m \text{ is even} \\ \sum_{k=2}^{\frac{n-m-1}{2}} \binom{N}{m+k} & \text{if } N-m \text{ is odd} \end{cases} \tag{8}$$

$$= \sum_{k=m+1}^{N} \binom{N-1}{k} \tag{9}$$

Now η_2 is the *second half of number of errors*. And being given

$$v_f = \eta_1 + \eta_2$$

we have:

$$\begin{aligned} v_f &= \sum_{k=0}^{m-1} \binom{N-1}{k} + \sum_{k=m+1}^{N} \binom{N-1}{k} \\ &= \sum_{k=0}^{N} \binom{N-1}{k} - \binom{N-1}{m} \end{aligned} \tag{10}$$

then:

$$v_f = 2^{N-1} - \binom{N-1}{m}$$

And therefore, v_f will be smaller when $\binom{N-1}{m}$ is bigger. Also, when $N = 2p$, $m = p$, and if $N = 2p+1$ then $m = p$ or $m = p+1$. □

3. Minimerror's Solution

We have studied the problem of the N-dimensional parity with Minimerror, a learning algorithm [2, 3] for perceptrons. This algorithm makes a gradient search of normalized weights $\boldsymbol{w}$, $\boldsymbol{w} \cdot \boldsymbol{w} = N$, through the minimization of a parameterized cost function,

$$E = \frac{1}{2} \sum_{\mu=1}^{P} V\left(\frac{\tau^{\mu} \boldsymbol{w} \cdot \boldsymbol{\xi}^{\mu}}{2T\sqrt{N}} \right) \tag{11}$$

$$V(x) = 1 - \tanh(x) \cdot \tag{12}$$

where $\boldsymbol{\xi}^{\mu}$ is the input pattern ($\mu = 1, \ldots, P$), and $\tau^{\mu} = \pm 1$ its class. The T parameter, called temperature (for reasons related to the cost function interpretation), defines an effective window width on both sides of the separating hyperplane. The derivative $\frac{\mathrm{d}V(x)}{\mathrm{d}x}$ is vanishingly small outside this window. Therefore, if the minimum cost's (11) is searched through a gradient descent, only the patterns μ at a

$$d^{\mu} \equiv \frac{|\boldsymbol{w} \cdot \boldsymbol{\xi}^{\mu}|}{\sqrt{N}} < 2T \tag{13}$$

distance will contribute significantly to learning [3]. Minimerror algorithm implements this minimization starting at high temperature. The weights are initialized with Hebb's rule, which is the minimum of (11) in the high temperature limit. Then, T is slowly decreased upon the successive iterations of the gradient descent – a procedure called *deterministic annealing* – so that only the patterns within the narrowing window of width 2T are effectively taken into account for calculating the correction

$$\delta \boldsymbol{w} = -\epsilon \frac{\partial E}{\partial \boldsymbol{w}} \tag{14}$$

at each time step, where ϵ is the learning rate. Thus, the search of the hyperplane becomes more and more local as the number of iterations increases. In practical implementations, it was found that convergence is considerably speeded-up if patterns already learned are considered at a lower temperature T_L than not learned ones, $T_L < T$. Minimerror algorithm has three free parameters: the learning rate ϵ of the gradient descent, the temperature ratio T_L/T, and the annealing rate δT at which temperature is decreased. At convergence, a last minimization with $T_L = T$ is performed.

Minimerror performs correctly in problems of high dimensionality, as it was recently shown with the discovering of the classic benchmark of the sonar problem [14]. Several efforts [15–19] have not succeeded in finding if it is linearly separable [9, 20, 21].

Coming back to the N-dimensional parity problem, we decided to verify the expression (4) experimentally. For this purpose we prepared exhaustive learning sets

of the N-dimensional parity for $2 \leqslant N \leqslant 15$ and $P = 2^N$. In all cases the Minimerror's solution corresponded exactly to the number of expected errors for (4).

4. Full Solution for *N*-dimensional Parity with Monoplan

In order to fully solve the N-dimensional parity problem, it is necessary to use a neural network with hidden units. A constructive approach allows a growth control of the network (number of units) in relation to difficulty of the learning set, to the one contrary of the BP and fixed architecture that suppose one defined architecture *a priori*.

In the recently introduced Monoplan algorithm [4], each hidden unit added serves to correct learning errors made by the precedent unit. A summary of this algorithm follows:

- Hidden layer.
  A perceptron trained with Minimerror learns the learning set $\mathcal{L}$. If the number of errors is null, $\varepsilon_t = 0$, then $\mathcal{L}$ is linearly separable and the algorithm stops: the neural network is a simple perceptron. If $\varepsilon_t > 0$, this perceptron becomes the first hidden unit, $h = 1$. A second unit $h + 1$ is added, and the classes to be learned are modified. The patterns classes are replaced with the new classes: $\tau_{h+1} = +1$ for the patterns that are well classified by the precedent unit, and $\tau_{h+1} = -1$ for those that are not adequately learned: $\tau^{\mu}_{h+1} = \sigma^{\mu}_{h}\tau^{\mu}_{h}$. It has been shown that each perceptron is capable to correct at least one learning error made by the previous perceptron. This guarantees the convergence of the algorithm [10, 11]. When the learning of a perceptron h concludes, its weight freezes. The hidden layer is generated until the last unit is able to learn all the outputs correctly.
- Output layer.
  The output unit ζ is connected now to all the units of the hidden layer. This unit learns the desired outputs τ^{μ}. If internal representations are LS, ζ will learn them and the algorithm stops. Otherwise, it returns to the first phase of hidden units aggregation, but this time the outputs to learn the new hidden unit $h + 1$ are: $\tau^{\mu}_{h+1} = \tau^{\mu}\zeta^{\mu}$.

These two phases converge, as shown by [8].

Monoplan begins to generate a parity machine: the outputs are the parity of the internal representations, like shown in [11, 12]. However, contrarily to the *Offset* algorithm that uses a second hidden layer to calculate the parity (if the output neuron detects that the internal representations are not linearly separable) Monoplan increases the dimension of the hidden layer until the internal representations are linearly separable.

In the N-dimensional parity problem, although it is known that the exact number of hidden units that allows to solve this problem with a network using one single hidden layer and no feedback (*feedforward*) is $H = N$, Backpropagation and other

non-constructive algorithms [13] cannot find it. Monoplan is able to find the correct solution, as we checked it experimentally until $N \leqslant 15$. Experimental results to solve N-parity beyond $N > 15$ are very difficult, because the gradient algorithm search through the minimization's cost fall in the multiple local minimal.

4.1. DEGENERATED INTERNAL REPRESENTATIONS

It is possible that several patterns are associated to the same internal representation. In other words, some internal representations are degenerated, since they associate an internal representation $\boldsymbol{\sigma}^{\mu}$ to each pattern $\mu = 1, \ldots, P$. In this way, several patterns may be associated to a single state in the hidden layer. For example, in the XOR problem the four patterns are associated only to different three states σ (figure 1a)*. This is a desirable phenomenon that we will call *contraction of the input's space* [5]. Indeed, for P patterns belonging to $\mathcal{L}^{\alpha}$, only $P_{\ell} \leqslant P$ will have an internal representations $\boldsymbol{\sigma}^{\nu}$; $\nu = 1, \ldots, P_{\ell}$.

From the output perceptron's point of view, it is enough to learn the P_{ℓ} different internal representations and to forget those repeated, that is to say, degenerate. Experimentally, we have found that a great number of repeated internal representations may complicate (and even to impede) the correct positioning of the separating hyperplane at the level of the output neuron. Indeed, if an internal representation has been very degenerated, it contributes to learning with a coefficient multiplied by its degeneration (number of repetitions). For example, in an extreme case where there

Table II. Hidden layers weights for the 10-dimensional parity

i	Bias	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
1	−1.04	−1.10	0.52	1.00	1.03	−1.03	−1.07	−1.02	−1.00	−1.07	−1.00
2	1.44	−0.93	0.88	0.92	0.92	−0.96	−0.93	−0.97	−1.06	−0.93	−0.93
3	2.45	0.68	−0.69	−0.73	− 0.71	0.68	0.72	0.74	0.73	0.71	0.68
4	2.47	−0.70	0.72	0.69	0.70	−0.70	−0.71	−0.69	−0.71	−0.68	−0.69
5	2.87	−0.54	0.54	0.51	0.53	−0.52	−0.52	−0.54	−0.50	−0.54	−0.52
6	2.84	0.49	−0.50	−0.58	−0.53	0.54	0.54	0.57	0.56	0.54	0.55
7	3.03	0.39	−0.37	−0.46	−0.45	0.41	0.42	0.43	0.47	0.42	0.45
8	3.06	−0.45	0.46	0.33	0.39	−0.42	−0.43	−0.40	−0.37	−0.43	−0.39
9	3.12	0.23	−0.17	−0.62	−0.40	0.26	0.19	0.31	0.49	0.25	0.39
10	3.12	−0.49	0.63	0.17	0.22	−0.28	−0.42	−0.33	−0.22	−0.38	−0.15

Table III. Output perceptron's weights for the 10-dimensional parity

Bias	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
1.00	1.00	−1.00	1.00	1.00	−1.00	−1.00	1.00	1.00	−1.00	−1.00

*On figures, for the N-dimensional parity $N + 1$ will be had representations different.

are only two different internal representations: σ^1 and σ^2, with a single example associated to σ^1, and $P-1$ examples associated to σ^2, σ^2 is very degenerated. If P is very big, the contribution of σ^1 to learning will not be very significant. In this case, Minimerror will put the only hyperplane near σ^2, and it will need a great quantity of iterations to put it in the appropriate place. Since two identical internal representations are faithful, it is impossible that they give different outputs. For learning the output, it is enough to keep only internal representations that are different. These representations constitute the non degenerate learning set:

$$\mathcal{L}_\ell = \{(\boldsymbol{\sigma}^\mu, \tau^\nu); \quad \nu = 1, \ldots, P_\ell\} \tag{15}$$

smaller than (1), which we used for output perceptron's training of the neural network. This procedure has the additional advantage of robust learning. We show in the Tables II and III the robust 10-parity full solution.

5. Conclusion

We presented the deduction of an expression in order to fully characterize the minimum number of errors (v_f) using a perceptron for solving the N-dimensional parity problem. We have made verifications experimentally using Minimerror. This efficient algorithm allows to find the more stable separating hyperplane, and to minimize errors in non linearly separable learning sets through an annealing deterministic, as well as a gradient descent of a parametrized cost function. A constructive heuristic solution with Monoplan fully solves the N-dimensional parity problem by finding the solution with the minimal feedforward neural network, with $h = N$ hidden units. Results until $N \leqslant 15$ for the minimum number of errors v_f detection with Minimerror, and $N \leqslant 10$ for the full Parity problem with Monoplan were presented in this paper. This couple of learning and incremental heuristic algorithms constitute a powerful tool for learning using Neural Networks.

Acknowledgements

J. M. Torres would like to thank Dr. Christian Lemaître for his helpful comments on the manuscript and for Diego Luna for his help with english.

References

1. Minsky, M. and Papert, S.: *Perceptrons*. MIT Press, Cambridge 1969.
2. Gordon, M. B. and Berchier, D.: Minimerror: A perceptron learning rule that finds the optimal weights. In: Michel Verleysen, (ed.), *European Symposium on Artificial Neural Networks*, pp. 105–110, Brussels, 1993. D facto.
3. Raffin, B. and Gordon, M. B.: Learning and generalization with minimerror, a temperature dependent learning algorithm. *Neural Comput.* **7**(6) (1995), 1206–1224.
4. Torres Moreno, J.-M. and Gordon, M.: An evolutive architecture coupled with optimal perceptron learning for classification. In: Michel Verleysen, (ed.), *European Symposium on Artificial Neural Networks*, pp. 365–370, Brussels, 1995. D facto.

5. Torres Moreno, J.-M.: Apprentissage et Généralisation par des Réseaux de Neurones: étude de nouveaux algorithmes constructifs. Ph.D. these INPG, Grenoble, France, 1997.
6. Cover, T. M.: Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14 (1965), 326–334.
7. Gardner, E.: Maximum storage capacity in neural networks. *Europhysics Letters*, **4** (1987), 481–485.
8. Torres Moreno, J. M. and Gordon, M. B.: Efficient adaptive learning for classification tasks with binary units. *Neural Comput*. **10**(4) (1997), 1007–1030.
9. Torres Moreno, J. M. and Gordon, M. B.: Characterization complete of the sonar benchmark. *Neural Processing Letters*. **7**(1) (1998), 1–4.
10. Gordon, M. B.: A convergence theorem for incremental learning with real-valued inputs. In *IEEE International Conference on Neural Networks*, pp. 381–386, Washington, 1996.
11. Martinez, D. and Estéve, D.: The offset algorithm: building and learning method for multilayer neural networks. *Europhysics Letters*, **18** (1992), 95–100.
12. Biehl, M. and Opper, M.: Tilinglike learning in the parity machine. *Physical Review A*, **44** (1991), 6888.
13. Peretto, P.: *An introduction to the Modeling of Neural Networks*. Cambridge, University Press, 1992.
14. Gorman, R. P. and Sejnowski, T. J.: Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, **1** (1988), 75–89.
15. Berthold, M. A.: probabilistic extension for the DDA algorithm. In: *IEEE International Conference on Neural Networks*, pp. 341–346, Washington, 1996.
16. Berthold, M. R. and Diamond, J.: Boosting the performance of RBF networks with dynamic decay adjustment. In: G. Tesauro, D. Touretzky, and T. Leen, (eds.), *Advances in Neural Information Processing Systems*, volume 7, pp. 521–528. The MIT Press, 1995.
17. Bruske, J. and Sommer, G.: Dynamic cell structures. In: G. Tesauro, D. Touretzky, and T. Leen, (eds.), *Advances in Neural Information Processing Systems*, volume 7, pp. 497–504. The MIT Press, 1995.
18. Chakraborty, B. and Sawada, Y.: Fractal connection structure: Effect on generalization supervised feed-forward networks. In: *IEEE International Conference on Neural Networks*, pp. 264–269, Washington, 1996.
19. Karouia, M., Lengellé, R. and Denoeux, T.: Performance analysis of a MLP weight initialization algorithm. In: Michel Verleysen, (ed.), *European Symposium on Artificial Neural Networks*, pp. 347–352, Brussels, D facto, 1995.
20. Hasenäger, M. and Ritter, H.: Perceptron Learning Revisited: The Sonar Targets Problem. *Neural Processing Letters*. **10**(1) (1999), 1–8.
21. Perantonis, S. J. and Virvilis, V.: Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis. *Neural Processing Letters*. **10**(3) (1999), 243–252.
22. Kim, J. H. and Kwong-Park, S.: The Geometrical Learning of Binary Neural Networks. *IEEE Transactions on Neural Networks*. **6**(1) (1995), 237–247.