ELSEVIER

Neurocomputing 48 (2002) 477–488

NEUROCOMPUTING

www.elsevier.com/locate/neucom

N-bit parity neural networks: new solutions based on linear programming

Derong Liu[a,*,1], Myron E. Hohil[b], Stanley H. Smith[c]

[a] *Department of Electrical and Computer Engineering, University of Illinois at Chicago, 851 South Morgan Street, Chicago, IL 60607-7053, USA*

[b] *Army Armament Research Development and Engineering, US ARMY TACOM, AMSTA-AR-FSF-RM, Picatinny Arsenal, NJ 07806-5000, USA*

[c] *Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030-5991, USA*

Received 20 January 2001; accepted 24 May 2001

Abstract

In this paper, the N-bit parity problem is solved with a neural network that allows direct connections between the input layer and the output layer. The activation function used in the hidden and output layer neurons is the threshold function. It is shown that this choice of activation function and network structure leads to several solutions for the 3-bit parity problem using linear programming. One of the solutions for the 3-bit parity problem is then generalized to obtain a solution for the N-bit parity problem using $\lfloor N/2 \rfloor$ hidden layer neurons. Compared to other existing solutions in the literature, the present solution is more systematic and simpler. Furthermore, the present solution can be simplified by using a single hidden layer neuron with a "staircase" type activation function instead of $\lfloor N/2 \rfloor$ hidden layer neurons. The present activation function is easier to implement in hardware than those in the literature for N-bit parity networks. We also review similarities and differences between the present results and those obtained in the literature.

Keywords: N-bit parity problem; Exclusive-OR problem; Neural networks

* Corresponding author. Tel.: +312-355-4475; fax: +1-312-413-0024.

E-mail address: dliu@ieee.org (D. Liu).

[1] D. Liu's work was supported by the National Science Foundation under Grant ECS-9732785.

PII: S0925-2312(01)00612-9

1. Introduction

The XOR/parity problem has a long history in the study of neural networks. It is used in [6] as a basis for illustrating the limitations of the computational power of perceptrons. The parity mapping problem has since been recognized as one of the most popular benchmarks in evaluating neural network training algorithms [2]. The N-bit parity function is a mapping defined on 2^N distinct binary vectors that indicates whether the sum of the N components of a binary vector is odd or even. When $N=2$, the parity function is the exclusive-or (XOR) logic function. Let $\beta=[\beta_1,\beta_2,\ldots,\beta_N]^{\mathrm{T}}$ be a vector in B^N where $B^N \triangleq \{\beta\in R^N\colon \beta_k=0 \text{ or } 1 \text{ for } k=1,2,\ldots,N\}$. The mapping $\psi: B^N \to B^1$ is called the N-bit parity function if

$$\psi(\beta)=\begin{cases} 1 & \text{if } \sum_{k=1}^{N}\beta_k \text{ is odd,} \\ 0 & \text{if } \sum_{k=1}^{N}\beta_k \text{ is even.} \end{cases} \tag{1}$$

The parity mapping is considered difficult for neural network learning since changes in a single bit results in changes in the output.

It is previously thought that a standard feedforward neural network model requires N hidden layer neurons to solve the N-bit parity problem [7]. The network in [7] uses the sigmoidal transfer function,

$$f(u)=\frac{1}{1+\mathrm{e}^{-u}},$$

and is trained using the generalized delta rule. Although it is shown in [7] that the XOR function could be realized using a neural network with a single hidden layer neuron and *direct connections* between the input and output layers, no generalization utilizing such a structure is made for the parity function with $N>2$.

In [9], it is shown that standard feedforward neural networks can be used to solve the N-bit parity problem with $\lceil (N+1)/2\rceil$ hidden layer neurons, where $\lceil\cdot\rceil$ stands for rounding towards $+\infty$. Using the same network structure, a solution is proposed in [8] for the N-bit parity problem. The connection weights between the inputs and hidden layer neurons can be obtained trivially, while the connection weights between the hidden layer neurons and the output layer neuron can be obtained by solving a system of linear equations.

The implementation of the N-bit parity problem using a different network structure is proposed in [5]. The implementation leads to a neural network with an output $y=u-2\eta$, where

$$\eta=\sum_{j=1}^{[N/2]}\frac{1}{1+\mathrm{e}^{-40(u-2j+0.15)}}, \tag{2}$$

$u=\sum_{i=1}^{N}x_i$ and $[\,\cdot\,]$ indicates truncation to the nearest integer. The connection weights from the inputs x_i to u and from u to the hidden and output layer neurons

are all equal to 1. It is also noted in [5] that the node u can be replaced by direct connections from the input nodes to the hidden and output nodes.

It is shown in [10] that using a standard feedforward neural network, the N-bit parity problem can be solved with just two hidden layer neurons. The activation function used in both hidden units is

$$f(u)=\frac{1}{N}\left(u-\frac{\cos(\pi u)}{\alpha\pi}\right), \tag{3}$$

where the choice of $\alpha>1$ ensures that the function is monotonically increasing. One of the hidden units has a constant bias of -1 while the other has zero bias. The output unit is a linear threshold unit with a constant bias of $-1/N$. All connection weights are equal to 1 except the weight from the first hidden unit to the output unit which is -1.

When direct connections between the input layer and the output layer are introduced, it is shown in [1], that the problem can be solved with a structure that requires only one hidden layer neuron. The activation function used in the hidden layer neuron is the continuously differentiable "step" function,

$$f(u)=\lfloor u\rfloor+\sin^K\left(\frac{\pi(u-\lfloor u\rfloor)}{2}\right), \tag{4}$$

where $K>1$ and $\lfloor\cdot\rfloor$ stands for rounding towards $-\infty$.

In this paper, the N-bit parity problem is solved with a neural network that allows connections between the input layer and the output layer. The threshold transfer function is used in the hidden and output layer neurons. It is shown that a set of solutions for the 3-bit parity problem can be obtained using linear programming. One of these solutions is then generalized to obtain a solution for the N-bit parity problem using $\lfloor N/2\rfloor$ hidden layer neurons. Furthermore, the present solution can be simplified by using a single hidden layer neuron with a "staircase" type activation function instead of $\lfloor N/2\rfloor$ hidden layer neurons. The present paper makes contributions beyond that of [3] by solving the N-bit parity neural networks systematically using linear programming. We also note that there has been a recent follow-up to our letter [3] published in 1999 (see, e.g., [4]).

2. Solutions to the 3-bit parity problem

It can easily be shown that all mappings $\psi: B^3\to B^1$ are realizable using the architecture in Fig. 1. The 256 distinct mappings of this type can be generated by appropriately selecting the connection weights $w_1, w_2, w_3, k, k_1, k_2, k_3$ and bias values b_1 and b_2. The output of the neural network model shown in Fig. 1 is given by

$$y=f(x_1,x_2,x_3)=\varphi[w_1x_1+w_2x_2+w_3x_3+b_1+k\varphi(k_1x_1+k_2x_2+k_3x_3+b_2)]$$

where the activation function φ is given by

$$\varphi(u)=\begin{cases}1 & \text{if } u>0,\\ 0 & \text{if } u\leqslant 0.\end{cases} \tag{5}$$

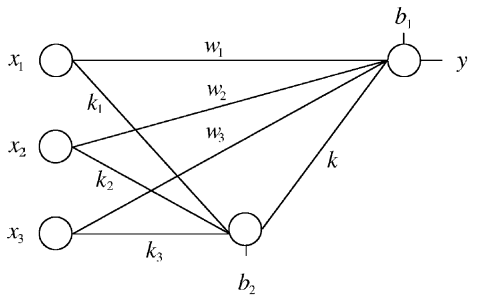

Fig. 1. Neural network structure used to map $\psi:\{0,1\}^3 \to \{0,1\}$.

Table 1
Truth table for the 3-bit parity function

	1	2	3	4	5	6	7	8
x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
y	0	1	1	0	1	0	0	1

In the present section, we will show how to use linear programming to obtain solutions for the 3-bit parity problem using the structure in Fig. 1. The truth table for the 3-bit parity function is shown in Table 1.

Remark 1. The structure in Fig. 1 can be used to obtain a solution for the 3-bit parity mapping as follows. First, connection weights between the inputs and the output are all set to 1, and the bias for the output neuron is set to -0.5. Disregarding the hidden units, this guarantees the realization of columns 1, 2, 3, and 5 in the truth table (Table 1). The connection weights between the input layer and the hidden layer neurons are also set to 1. We now consider columns 4, 6, and 7 in Table 1. In order to inhibit the output neuron when two of the inputs are true, we select a threshold of -1.5 for the hidden layer neuron and choose a weight of -2 between the hidden layer neuron and the output neuron. This choice automatically satisfies the condition in column number 8.

The solution in Remark 1 can also be obtained in a more systematic manner. The 2^3 combinations of x_1, x_2, and x_3 (cf. Table 1) are used to generate the following system of equations:

(1) $f(0,0,0) = \varphi[b_1 + k\varphi(b_2)]$,
(2) $f(0,0,1) = \varphi[w_3 + b_1 + k\varphi(k_3 + b_2)]$,
(3) $f(0,1,0) = \varphi[w_2 + b_1 + k\varphi(k_2 + b_2)]$,
(4) $f(0,1,1) = \varphi[w_2 + w_3 + b_1 + k\varphi(k_2 + k_3 + b_2)]$,
(5) $f(1,0,0) = \varphi[w_1 + b_1 + k\varphi(k_1 + b_2)]$,

(6) $f(1,0,1)=\varphi[w_1+w_3+b_1+k\varphi(k_1+k_3+b_2)]$,
(7) $f(1,1,0)=\varphi[w_1+w_2+b_1+k\varphi(k_1+k_2+b_2)]$,
(8) $f(1,1,1)=\varphi[w_1+w_2+w_3+b_1+k\varphi(k_1+k_2+k_3+b_2)]$.

Depending on the values of the function on the left hand side, the above 8 equations can be equivalently written as 8 inequalities. For example, equation no. (8) can be written as

$$w_1+w_2+w_3+b_1+k\varphi(k_1+k_2+k_3+b_2)>0 \quad \text{if } f(1,1,1)=1$$

or

$$w_1+w_2+w_3+b_1+k\varphi(k_1+k_2+k_3+b_2)\leqslant 0 \quad \text{if } f(1,1,1)=0.$$

The resulting set of equations is a system of linear inequalities for which linear programming provides a solution. The linear programming problem in standard inequality form is given as

$$\begin{aligned} &\text{maximize} \quad c^{\mathrm{T}}\xi \\ &\text{subject to} \quad A\xi \leqslant b, \end{aligned} \tag{6}$$

where A is an $m\times n$ coefficient matrix, b is an $m\times 1$ column vector, c is an $n\times 1$ vector, and the decision variables ξ_j, $j=1,\ldots,n$, are contained in the $n\times 1$ column vector ξ.

It turns out that the design problem for any binary mapping $B^3\rightarrow B$ can be solved using linear programming. We will demonstrate this in the following for the 3-bit parity problem. In this case, the 8 equations above are equivalent to the following inequalities

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & -1 & 1 \\ 0 & -1 & 0 & -1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ -1 & 0 & 0 & -1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ b_1 \\ \varepsilon \end{bmatrix} \leqslant \begin{bmatrix} -k\varphi(b_2) \\ k\varphi(k_3+b_2) \\ k\varphi(k_2+b_2) \\ -k\varphi(k_2+k_3+b_2) \\ k\varphi(k_1+b_2) \\ -k\varphi(k_1+k_3+b_2) \\ -k\varphi(k_1+k_2+b_2) \\ k\varphi(k_1+k_2+k_3+b_2) \end{bmatrix}. \tag{7}$$

For example, for no. (8), we have

$$w_1+w_2+w_3+b_1+k\varphi(k_1+k_2+k_3+b_2)>0$$

since $f(1,1,1)=1$. This is equivalent to

$$-w_1-w_2-w_3-b_1+\varepsilon\leqslant k\varphi(k_1+k_2+k_3+b_2)$$

where $\varepsilon>0$. There are eight φ functions on the right-hand side of (7). Assuming $k=-2$, there are 256 possible choices for the right-hand side of (7). Each choice

gives a set of conditions for k_1, k_2, k_3 and b_2. For example, the following is one of the possible choices:

$$b_2 > 0,$$

$$k_3 + b_2 \leqslant 0,$$

$$k_2 + b_2 \leqslant 0,$$

$$k_2 + k_3 + b_2 > 0,$$

$$k_1 + b_2 \leqslant 0,$$

$$k_1 + k_3 + b_2 \leqslant 0,$$

$$k_1 + k_2 + b_2 > 0,$$

$$k_1 + k_2 + k_3 + b_2 > 0,$$

which can be equivalently written as

$$\begin{bmatrix} 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & -1 & -1 & -1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ -1 & -1 & 0 & -1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ b_2 \\ \varepsilon \end{bmatrix} \leqslant \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{8}$$

where $\varepsilon \geqslant 0$.

Linear programming can be used to solve (7) and (8) with the objective function $c^{\mathrm{T}}\xi = \varepsilon$ subject to the constraints in (7), (8) and $\varepsilon \geqslant 0$. We used MATLAB to solve the above system of inequalities and we obtained 4 valid solutions (from a total of 256 possible cases) for the 3-bit parity problem as shown in Table 2 (assuming $k = -2$). No solution exists when we assume $k \geqslant 0$. Solutions for other $k < 0$ are simply scaled versions of the ones shown in Table 2.

Remark 2. All the 256 possible mappings $\psi : B^3 \to B$ can be solved using linear programming as outlined above. We have verified this claim experimentally.

The first solution in Table 2 is what we have mentioned in Remark 1. We will generalize this solution to the N-bit parity problem in the next section.

Table 2
Solutions for the 3-bit parity problem

	1	2	3	4
w_1	1	1	−1	−1
w_2	1	−1	1	−1
w_3	1	−1	−1	1
b_1	−0.5	1.5	1.5	1.5
k	−2	−2	−2	−2
k_1	1	1	−1	−1
k_2	1	−1	1	−1
k_3	1	−1	−1	1
b_2	−1.5	0.5	0.5	0.5

Remark 3. It is interesting that we also obtained three other solutions to the 3-bit parity network. These three solutions have in common the following:

(1) they have the same threshold values;
(2) the connection weights are either 1 or −1;
(3) connection weights to the hidden neuron and to the output neuron have the same value if they are from the same input neuron; and
(4) there are always two links with weights value −1 and the other link with weight value 1 at each layer.

3. Solutions to the N-bit parity problem

The general solution to the N-bit parity problem can be obtained from the solution for the 3-bit parity problem and the resulting structure of Fig. 1. We make note of the fact that when the input x_3 and all of its corresponding connections are removed, we obtain a well known solution to the XOR problem using only one hidden neuron and direct connections between the inputs and the output. The results of column 1 in Table 2 indicate that the 3-bit parity problem ($N=3$) is realized with the same number of neurons as in the case of XOR problem, by adding connections from the input x_3 to both the output neuron and hidden layer neuron with a connection weight of 1. The truth table for the 3-bit parity function in Table 1 reveals that when $x_3=0$ we obtain the XOR between the $2=N-1$ remaining inputs x_1 and x_2. It is interesting to note that when $x_3=1$, we obtain in Table 1 the complement of XOR for each pair of the same values of x_1 and x_2. This is realized by adding one connection from the input x_3 to the output layer neuron and another connection to the hidden layer neuron, with both weights equal to 1. The hidden layer neuron in the proposed structure inhibits the output whenever exactly two inputs are 1, because the bias value of the hidden neuron is chosen as −1.5 and the weights from the hidden neuron to the output neuron is chosen as −2.

In the case of the 4-bit parity function ($N=4$), when the input corresponding to the most significant bit is 0 (i.e., $x_4=0$), we obtain the $2^{(N-1)}$ output values

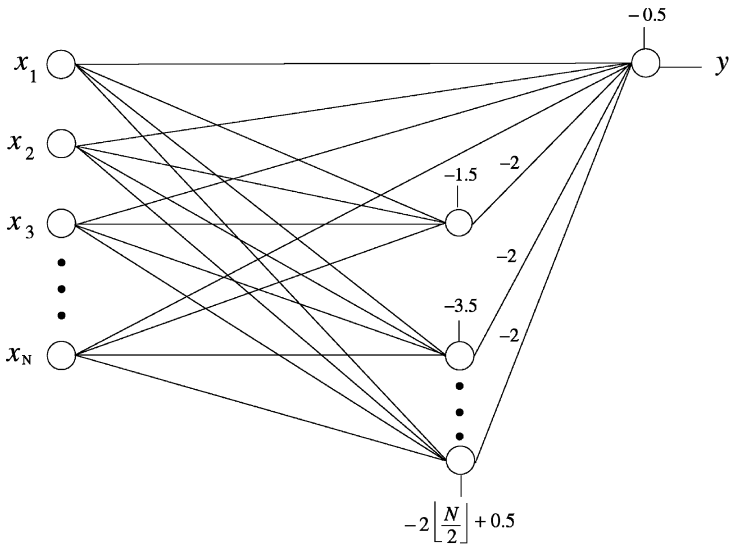

Fig. 2. Structure used to solve the N-bit parity problem.

which are the same as that for the 3-bit parity function. The remaining $2^{(N-1)}$ output values are complements of the previous $2^{(N-1)}$. Following the procedure outlined above, we add a direct link from the input x_4 to the output and hidden layer neurons with a connection weight of 1. This will ensure that when 1 or 3 inputs are 1, the output neuron will be enabled and when only 2 inputs are 1, the output will be zero. To ensure that we obtain an output of zero when all 4 inputs are 1, we add an additional hidden layer neuron with connection weights of 1 from the four inputs, a connection weight of -2 from the hidden layer neuron to the output, and a bias of -3.5 for the hidden layer neuron. To obtain a solution for the 5-bit parity problem, we modify this structure again by adding a connection from the input x_5 to the output neuron as well as to the two hidden layer neurons with connection weights equal to 1. We can continue the same process to obtain a solution for every N-bit parity problem using a similar network structure. The network structure for N-bit parity problem is shown in Fig. 2.

Remark 4. Assume that N is an even number. It is noted that the N-bit parity network and the $(N+1)$-bit parity network will have the same number of hidden layer neurons which is given by $\lfloor N/2 \rfloor$. Therefore, in the present solution, the N-bit parity network and the $(N+1)$-bit parity network will have the identical structure and identical network parameters except with different number of input neurons. Also, the N-bit parity network is built from $(N-1)$-bit parity network by adding a hidden layer neuron with the same connection weights as other hidden layer neurons and with a bias of $-2\lfloor N/2 \rfloor + 0.5$. Note that these statements are true for even number N.

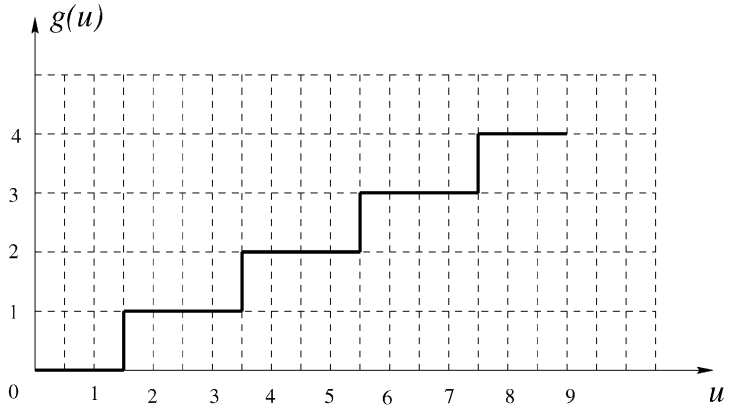

Fig. 3. Transfer function for the single hidden layer neuron when $N = 8, 9$.

The above analysis reveals that by expanding the architecture proposed in Fig. 1 to that of Fig. 2, we can solve the N-bit parity problem using $\lfloor N/2 \rfloor$ hidden layer units. Note that the hidden and output layer neurons use the same activation function. The resulting network may therefore be interpreted as a *two-layer* network with $\lfloor N/2 \rfloor + 1$ output units and no hidden layer units [11]. It is expected that the remaining three solutions to the 3-bit parity problem in Table 2 can also be generalized in a similar manner to obtain solutions to the N-bit parity problem. Currently the method to determine such a solution has not been identified. We attribute the difficulty in the generalization of the remaining three solutions in Table 2 to the observations made in Remark 3.

Careful analysis of the role of the hidden layer neurons in Fig. 2 reveals that they can be further combined into one hidden layer neuron through the use of the transfer function $g(u) = \lfloor (u+0.5)/2 \rfloor$, where $u = \sum_{i=1}^{N} x_i$. For example, when $N = 8$ or 9, the required transfer function $g(\cdot)$ is shown in Fig. 3.

We now briefly review the similarities and differences between our results and those of [1,5,10]. In [5], a structure that is functionally equivalent to Fig. 2 is used to solve the N-bit parity problem. The summing node u can be replaced by direct connections between the input and output neurons. The solution requires similar number of hidden layer units depending on the value of N. The main difference is that [5] uses an activation function of the form (2), shown in Fig. 4. Note that the summation is multiplied by a connection weight of -2 from each hidden layer neuron. It is easily seen that the function in Fig. 4 is functionally equivalent to the staircase function of Fig. 3 since u only takes on integer values.

The transfer function in (2) is similar to the function (3) used for the two hidden layer neurons in [10] and to the function (4) used in [1] in that all three functions are monotonically increasing functions. The activation functions in (2)–(4) all resemble the "staircase" function used in the present paper.

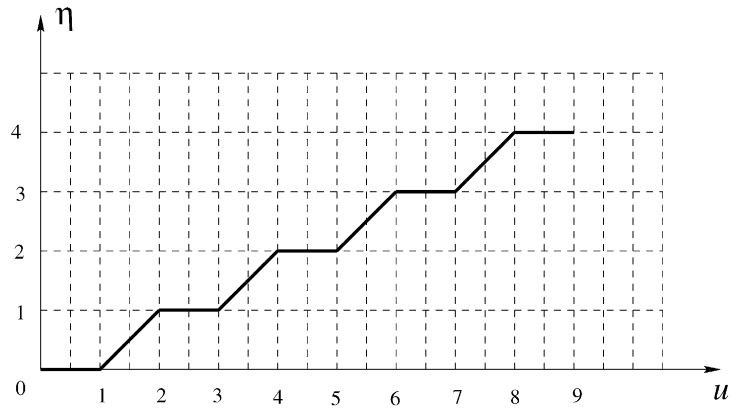

Fig. 4. Transfer function used in (2) when $N = 8, 9$.

In addition to this similarity, structures in [1,5] also allow direct connections between the input layer and the output layer. In cases where practical issues arise concerning hardware realization, we note that the transfer function described in this paper is easier to implement.

4. Conclusion

We have shown that when direct connections are allowed between the input neurons and the output neuron, the N-bit parity mapping problem can be solved using neural networks requiring $\lfloor N/2 \rfloor$ hidden layer neurons. It is shown that through the choice of a "staircase" type activation function, the $\lfloor N/2 \rfloor$ hidden layer neurons can be further combined into a single hidden layer neuron. Our solution to the N-bit parity problem is generalized from a solution obtained for the 3-bit parity problem using linear programming. Finally, we note that the present approach is constructive and it requires no training and adaptation.

References

[1] D.A. Brown, N-Bit parity networks, Neural Networks 6 (1993) 607–608.
[2] S.E. Fahlman, An Empirical Study of Learning Speed in Back-Propagation Networks, Technical Report CMU-CS-88-162, Department of Computer Science, Carnegie Mellon University, September 1988.
[3] M.E. Hohil, D. Liu, S.H. Smith, Solving the N-bit parity problem using neural networks, Neural Networks 12 (1999) 1321–1323.
[4] E. Lavretsky, On the exact solution of the parity-N problem using ordered neural networks, Neural Networks 13 (2000) 643–649.
[5] J.M. Minor, Parity with two layer feedforward nets, Neural Networks 6 (1993) 705–707.

[6] M.L. Minsky, S.A. Papert, Perceptrons, MIT Press, Cambridge, MA, 1969.
[7] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland, the PDP Research Group (Eds.), Parallel Distributed Processing, Vol. 1, The MIT Press, Cambridge, MA, 1986 (Chapter 8).
[8] R. Setiono, On the solution of the parity problem by a single hidden layer feedforward neural network, Neurocomput. 16 (1997) 225–235.
[9] E.D. Sontag, Feedforward nets for interpolation and classification, J. Computer System Sci. 45 (1992) 20–48.
[10] D.G. Stork, J.D. Allen, How to solve the N-bit parity problem with two hidden units, Neural Networks 5 (1992) 923–926.
[11] D.G. Stork, N-bit parity networks: a reply to Brown and Korn, Neural Networks 6 (1993) 609.

Derong Liu received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, Indiana, in 1994; the M.S. degree in electrical engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1987; and the B.S. degree in mechanical engineering from the East China Institute of Technology (now Nanjing University of Science and Technology), Nanjing, China, in 1982. From 1982 to 1984, he worked at China North Industries Corporation, Jilin, China. From 1987 to 1990, he was an instructor at the Graduate School of the Chinese Academy of Sciences, Beijing, China. From 1993 to 1995, he was with General Motors Research and Development Center, Warren, Michigan. From 1995 to 1999, he was Assistant Professor in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, New Jersey. Currently, he is Assistant Professor in the Department of Electrical and Computer Engineering, University of Illinois, Chicago, Illinois. He is coauthor of the book Dynamical Systems with Saturation Nonlinearities: Analysis and Design with A.N. Michel, New York: Springer-Verlag, 1994.

Dr. Liu was a member of the Program Committees of 11th and 14th IEEE International Symposium on Intelligent Control (1996 and 1999), 2000 International Conference on Artificial Intelligence, and 39th IEEE Conference on Decision and Control (2000); he was the Local Arrangements Chair for the 6th IEEE Conference on Control Applications (1997); he was a member of the Conference Editorial Board of the IEEE Control Systems Society (1995–2000); and he served as an Associate Editor for IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications (1997–1999). Dr. Liu is a member of the Program Committee of the 16th IEEE International Symposium on Intelligent Control (2001); he is a member of the Program Committee of the IFIP/IEEE International Conference on Management of Multimedia Networks and Services (2001); he is a member of the Program Committee of the 40th IEEE Conference on Decision and Control (2001); and he serves as an Associate Editor for IEEE Transactions on Signal Processing.

Dr. Liu was recipient of Michael J. Birck Fellowship from the University of Notre Dame (1990); he was recipient of Harvey N. Davis Distinguished Teaching Award from Stevens Institute of Technology (1997); and he was recipient of Faculty Early Career Development (CAREER) award from the National Science Foundation (1999). He is a member of Eta Kappa Nu.

Myron E. Hohil received the B.E. degree in 1991, the M.E. degree in 1994 and the Ph.D. degree in 1998 from Stevens Institute of Technology, Hoboken, NJ, all in Electrical Engineering. From 1996 to 1997 he served as a faculty member of the Electrical and Computer Engineering Department at Stevens Institute of Technology. He is currently a principal consultant to the U.S. Army Armament Research, Development and Engineering Center, Tank-automotive and Armaments Command (TACOM-ARDEC). A member of the IEEE, his research interests include neural networks, array signal processing, sensor fusion and multiple target tracking.

Dr. Stanley H. Smith received the B.A.E., M.A.E., Ph.D. (Engineering Science), and M.E. (honorary) degrees from New York University, Cornell University, New York University and Stevens Institute of Technology, respectively. In 1966, he joined the faculty of the Stevens Institute of Technology in Hoboken, N.J., where he is currently a Professor (Emeritus) of Electrical and Computer Engineering. From 1992 to 1996, he was the Head of the Department of Electrical Engineering and Computer Science.

Dr. Smith has served on technical evaluation and program committees and expert panels for SDIO (now BMDO), BTI, DARPA and Department of the Army programs. The areas that he was involved with were sensors/seekers, electronics, computer architectures, algorithm development and communications. He is a member of the Board of Advisers of the Northeast Region of the Federal Laboratory Consortium (FLC) for Technology Transfer, encompassing all U.S. Government laboratories in New Jersey, New York and New England. He was also a member of the National Advisory Group of the FLC.

His present research interests are concerned with multimedia, interactive video and wireless communications, sensors/seekers, multispectral image processing and pattern recognition (autonomous target recognition), application specific computer architectures (ASCA) for high-speed VLSI and VHSIC realizations including neural networks and embedded computer design. Dr. Smith is the author of more than ninety technical reports and papers. He was a contributing author in books on digital signal processing and on product defects for the legal profession. He is a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), the IEEE Computer Society, the IEEE Communications Society and the Society for Photographic and Instrumentation Engineers (SPIE).