Xing Wu and Bogdan M. Wilamowski, "Advantage analysis of Sigmoid based RBF Networks", *17-th IEEE Intelligent Engineering Systems Conference, INES 2013,* Costa Rica, June 19-21., 2009, pp. 243-248.

Advantage analysis of Sigmoid based RBF Networks

Xing Wu and Bogdan M. Wilamowski, *Fellow, IEEE*
Department of Electrical and Computer Engineering, Auburn University, AL, USA
E-mail: xzw0015@tigermail.auburn.edu, wilam@ieee.org

***Abstract*—By introducing an extra dimension to the inputs, sigmoid function can simulate the behavior of traditional RBF units. This paper introduces a sigmoid based RBF neuron and compares it with traditional RBF neuron. Neural networks composed of these neurons are trained with ErrCor algorithm on two classic experiments. Comparison results are presented to show advantages of the sigmoid based RBF model.**

Keywords—RBF, sigmoid, neural network, ErrCor algorithm

I. INTRODUCTION

Inspired by biological neural networks, artificial neural networks (ANN) are widely applied in function approximation, classification, data processing and pattern recognition. As a special type of ANN that uses radial basis function as activation function, radial basis function (RBF) networks are found more convenient and more powerful than conventional sigmoid neural network[1][2]. Recent research [3] compared traditional sigmoid neural networks with RBF networks and concluded that RBF networks worked better in regular function approximation, RBF networks performed more robustly and tolerantly than traditional neural networks while dealing with noised input data set.

Though RBF networks are powerful and easy to use, they are usually criticized for not reassembling biological neurons. On the other hand, biological neurons are not able to compute a distance in multidimensional input space. In fact, for some special problems, like two-spiral problem, traditional RBF networks don't work quite efficient because of its locally tuning mechanism and flat limit property. To improve these drawbacks of traditional RBF networks, many attempts were made. A method to simulate traditional RBF unit with sigmoid function by projecting inputs on a hyper sphere [4] solves this limit of conventional RBF and improves the computation significantly. This paper presents the implementation of this proposed model and compares it with traditional RBF neural networks through several experiments.

The paper is organized as following. In the section II, the fundamentals of traditional RBF networks and RBF unit are presented briefly. Section III introduces the sigmoid based RBF model. Section IV gives a brief introduction of Error Correction (ErrCor) algorithm for RBF networks training. In Section V, several classic experiments are taken to compare these models.

II. FUNDAMENTALS OF RBF NETWORKS AND UNITS

A. RBF networks

RBF networks have the fixed structure with three layers: an input layer, a hidden layer with RBF activation function and a linear output layer.

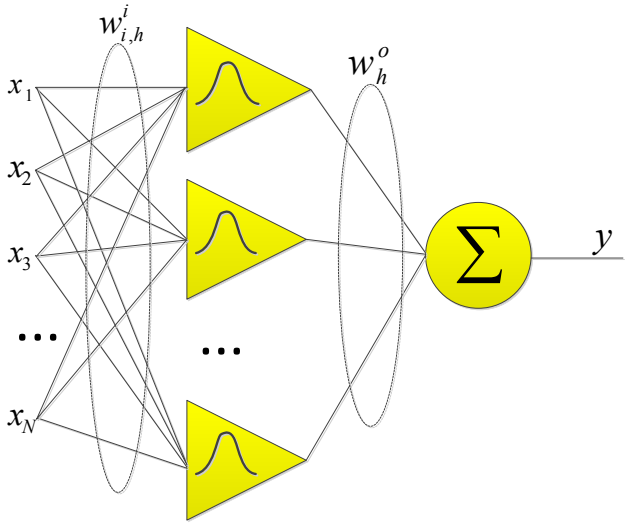

Fig. 1. RBF network structure

B. Radial basis function

A radial basis function depends only on the distance between input(**X**) and center(**C**).

$$\Phi(X,C)=\Phi(\|X-C\|) \tag{1}$$

In which $\|\bullet\|$ represents Euclidean distance.

There are different types of RB function like multiquadric, inverse quadratic, etc. Most popular used is Gaussian function:

$$\phi(X)=\exp\left(-\frac{\|X-C\|^2}{\sigma}\right) \tag{2}$$

In which, X is input vector, C is center vector and σ is width. $\phi(\bullet)$ represents output of the RBF unit. Figure 2 shows a 2-dimension example of RBF with center (0,0) and width(1).

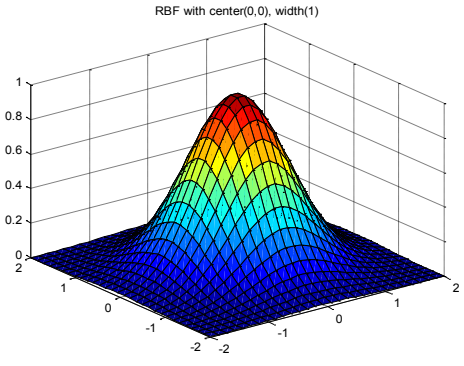

Fig. 2. RBF (C=(0,0), σ =1)

C. Network computation

Because of the fixed three layer structure, it's very easy to compute RBF networks. Consider the network in Figure 1, it has N inputs (x_1 , x_2 … x_N), H hidden neurons and one output(y). Assume the weight connecting the i_{th} input to the h_{th} hidden neuron is $w_{i,h}^{in}$, and the weight connecting the h_{th} hidden neuron to output neuron is w_h^o . The computation is processing in following three steps:

1) Multiply all the inputs with corresponding weights of each hidden neuron. Results will be inputs of each RBF neuron.

$$IN_h = (x_1 * w_{1,h}^{in}, x_2 * w_{2,h}^{in}, ... x_N * w_{N,h}^{in}) \qquad (3)$$

2) Compute output of each hidden RBF neuron (O_h).

$$O_h = \exp(-\frac{\|IN_h - C_h\|^2}{\sigma_h}) \qquad (4)$$

3) Multiply all the hidden neuron outputs with output weights.

$$y = O_1 * w_1^o + O_2 * w_2^o + ... + O_H * w_H^o \qquad (5)$$

III. Sigmoid Based RBF Unit and Network

From above introduction to RBF networks, the advantages of RBF networks over conventional neural networks are obvious. Conventional sigmoid neuron can only linearly divide space into two categories. Therefore, in order to separate one cluster in 2-dimension as shown in Fig. 3, at least 3 sigmoid neurons are needed; likewise, at least N+1 sigmoid neurons are needed to separate the cluster in N-dimension. While the same problem in any dimension can always be solved with only one RBF neuron. The 2-dimension situation is shown in Fig. 4.

On the other hand, as the property shown in Fig. 2, RBF unit only has output value around its center and all the other parts will be zero. The single layer structure of RBF networks make all the RBF units independent to each other. As a result, every unit only cares one part of the function to be approximated and doesn't affect other parts.

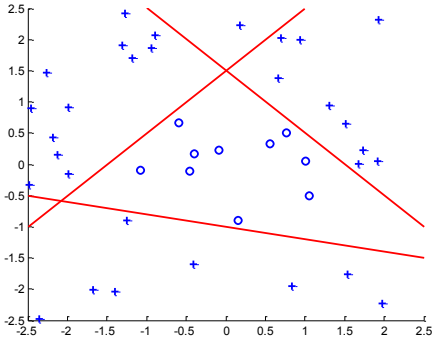

Fig. 3. Contour of sigmoid neurons to separate cluster

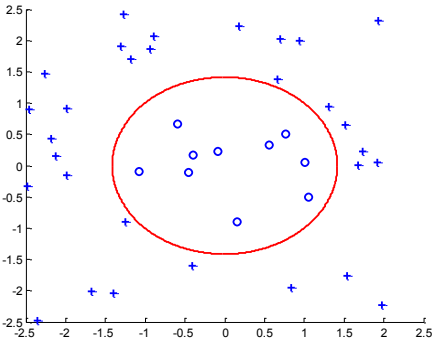

Fig. 4. Contour of RBF neuron to separate cluster

Except these advantages over traditional neural networks, RBF networks are criticized that they behave differently than biological neurons since actual neurons cannot compute a distance between an input and a stored pattern (center). To solve this dilemma, a simple way is to increase the input dimension by nonlinear transformation. It can be done in several ways [5-8]. But it seems that the most efficient way is to project input space into a sphere [4]. Assume inputs are (X_1 , X_2 ,… X_N), an extra input can be calculated as:

$$X_{N+1} = \sqrt{R^2 - X_1^2 - X_2^2 - ... - X_N^2} \qquad (6)$$

In which, R represents predefined radius.

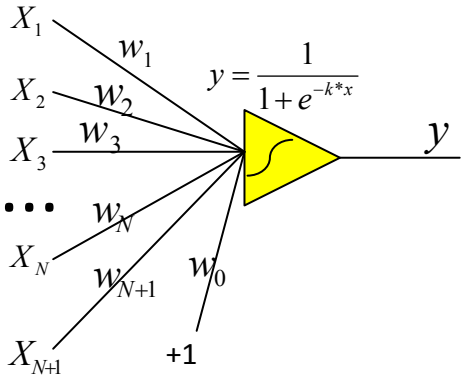

Fig. 5. Sigmoid neuron

In this way, the problem dimension increases from N to N+1. All the input points are mapped onto a half-hyper sphere with radius R in the N+1 dimension space. The sigmoid function is cutting the half-hyper sphere with a hyper plane whose equation is

$$w_0 + X_1 * w_1 + X_2 * w_2 + ... + X_{N+1} * w_{N+1} = 0 \quad (7)$$

The points above this hyper plane tend to be convex while those points below it tend to be flat. This property is obviously the advantage of RBF unit.

In this new "RBF" unit, the center would be proportional to the weights ($w_1, w_2, \ldots w_{N+1}$) and its Euclidean norm is R. Gain(k) of the sigmoid function will function like the width of traditional RBF unit. Fig. 6 shows an example transform from 2D to 3D. In which, $gain = 10$ and

$$(w_0, w_1, w_2, w_3) = (-7.84, -0.8966, -0.2069, 2.6746)$$

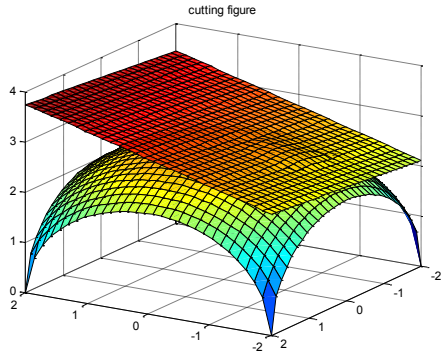

Fig. 6. Cutting hypersphere

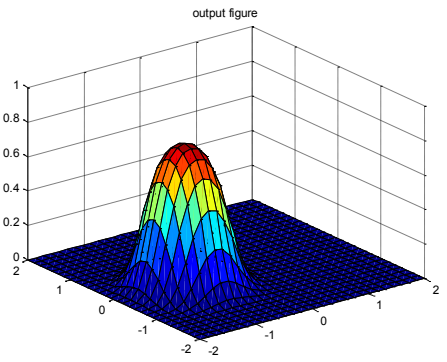

Fig. 7. Output figure

Fig. 7 shows the final output of the sigmoid neuron. Comparing this figure with Fig. 2, all advantages of traditional RBF unit can be realized in the sigmoid based "RBF" unit.

In fact, the sigmoid based RBF neuron is more flexible than traditional RBF neuron. Not only the output shown in Fig. 7 can be achieved, with the proposed sigmoid based RBF neuron, we can also get output figure with a flat top when adjusting those input weights, as shown in Fig. 8.

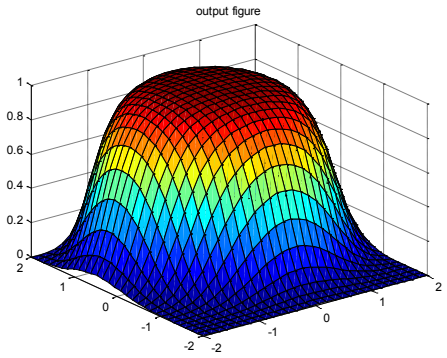

Fig. 8. Output figure with flat top

Another advantage of sigmoid based RBF network compared to traditional RBF network is the training cost. Review section II, traditional RBF networks training requires tuning of input weights, hidden neurons' centers and widths, and output weights. While sigmoid based RBF networks only need to train input weights (including bias weights) and output weights. Assume for an N-input network, consider the traditional RBF network shown in Fig. 1, every time add a new neuron, (2*N+2) more parameters (N input weights, N-dimension center, width and 1 output weight) will be added for the network. While for sigmoid RBF network, as Fig. 9 shows, every increased neuron only needs to add (N+3) more parameters (N+2 input weights, including added input and bias, 1 output weight) for training. As a result, sigmoid based RBF networks training is simpler and more efficient compared to traditional RBF networks, especially for multiple-dimension problems.

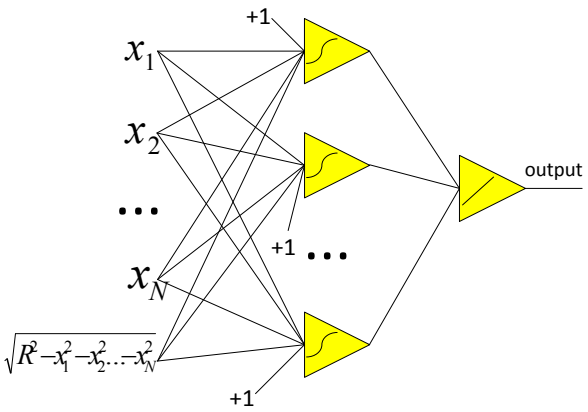

Fig. 9. Sigmoid based RBF network

IV. TRAINING ALGORITHM

One of the core problems to train RBF networks is to determine the structure and centers for the network. Random selection of centers is obviously not an effective option. Many attempts were made to find an optimal way. Moody and Darken [9] proposed a method to determine the centers and widths of receptive fields with self-organized selection. S.

Chen, C. F. N. Cowan, and P. M. Grant [10] applied an orthogonal least square (LS) algorithm to evaluate the optimal number of hidden units. Wu and Chow [11] presented an extended self-organizing map to optimize the number of hidden units. Orr [12] combined forward subset selection and zero-order regularization to select the centers of RBF networks.

In this paper, we use the ErrCor algorithm [13] for network construction and use Levenberg-Marquardt (LM) algorithm [14] to train the RBF networks.

ErrCor algorithm is efficient and easy to process. It adds the hidden RBF neuron one by one. Every time before adding the new neuron, train the whole network and evaluate the error of each pattern, then select the pattern with biggest error as center of the new neuron. Repeat this process iteratively until the desire error arrives. Following shows pseudo code of this algorithm:

```
// pseudo code of ErrCor algorithm
while 1
    Evaluate error of each pattern;
    Calculate SSE;   // Sum of Squared Errors
    if SSE<desired SSE
        break;
    end;
    C=pattern with biggest error;
    Add a new neuron with center=C;
    Train the whole network;
end:
```

In the process, we use LM algorithm to train the whole network. LM algorithm provides a numerical solution to the problem minimizing a function, especially for nonlinear, over a space of parameters of the function. It interpolates between the Gauss-Newton algorithm (GNA) and the method of gradient descent and it's more robust than GNA and much faster than gradient descent. The pseudo code of LM algorithm is shown below:

```
// pseudo code of LM algorithm
mu=0.1; mu_max=10e20; mu_min=10e-20;
Rearrange all the parameters(weights) in a vector(W);
for iter=1:maxiter
    count=0;
    calculate Jacobian matrix(J); // derivatives
   gradient=Jacobian'*error;
   hessian=Jacobian'*Jacobian;
   del=inv(hessian+mu*I)*gradient;// change of weights
   W_back=W;
   while 1
       W=W_back+del;
       Evaluate SSE(iter);
       if SSE(iter)<=SSE(iter-1)
           if mu>mu_min
               mu=mu/10;
           end;
          break;
        end;
       if mu<mu_max
           mu=mu*10;
       end;
       count=count+1;
       if count>5
           break;
       end;
```

Using ErrCor algorithm to construct network and LM algorithm to train, RBF networks can work efficient on function approximation, classification problems.

V. EXPERIMENTS

In this section, parity-N problem and a highly nonlinear 2D bench peak function are presented to test traditional RBF networks and sigmoid based RBF networks. In both experiments, ErrCor algorithm is used to train these two types of networks and experiments results are given to show differences between them.

The testing environment used for proposed experiments is: Windows 7 Professional 64-bit operating system; Intel Core2 Quad CPU Q8400 @2.66GHz; with 4.00GB RAM.

A. Peaks function approximation

Peaks function is a highly nonlinear function of two variables obtained by translating and scaling Gaussian distributions. As a result, RBF network is the best option to approximate the peaks function. In this paper, peaks function with following formula is used to test the sigmoid based RBF network. Fig. 10 shows the output of peaks function with 50X50 resolution.

$$z(x,y) = (0.3 - 1.8x + 2.7x^2)\exp(-1 - 6y - 9x^2 - 9y^2)$$
$$-\frac{1}{30}\exp(-1 - 6x - 9x^2 - 9y^2)$$
$$-(0.6x - 27x^3 - 243y^5)\exp(-9x^2 - 9y^2)$$

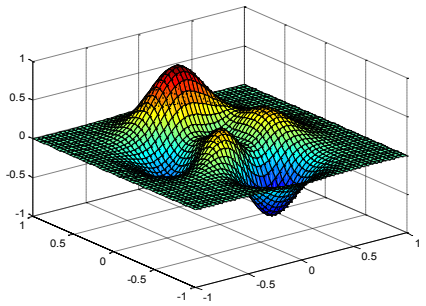

Fig. 10. Peaks function

To approximate peaks function, 15X15 points are used to train the RBF networks and 80X80 points are used as test data set for validation. With ErrCor algorithm, both traditional RBF networks and sigmoid based RBF networks increase hidden neurons from 1 to 20. Fig. 11 shows the root mean square errors(RMSE) observations of training and validation while adding each hidden neuron.

From the training observation, we can see both types of RBF networks work efficient and quite small RMSE (smaller than 1e-3) arrived while approximating peaks function. This also proves that proposed sigmoid based networks can simulate behavior of traditional RBF networks well. Due to the advantages in local tuning, traditional RBF networks.

converge faster as number of hidden neurons increases and errors decreases

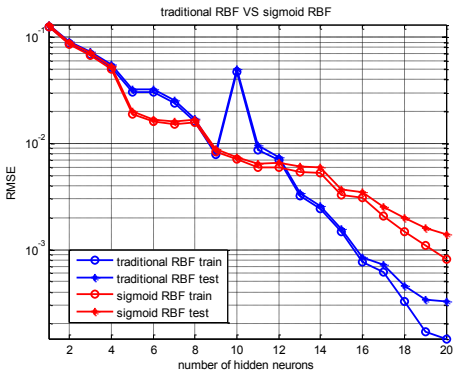

Fig. 11. Training observation

B. *Parity-N Problem*

Parity-N problems have been studied deeply in many literatures [15-16]. The N-bit parity function can be interpreted as a mapping (defined by 2^N binary vectors) that indicates whether the sum of the N elements of every binary vector is odd or even. Fig. 12 shows the problem when N is 2.

input	output
00	0
01	1
10	1
11	0

Fig. 12. Parity-2 problem

It is shown that threshold networks with one hidden layer require N hidden threshold units to solve the parity-N problem [17-18]. In this paper, considering RBF networks are more powerful, we increase number of hidden neurons to N while training parity-N problem. For 2^N patterns in parity-N problem, 70% of them are randomly selected as training data set, and the remaining 30% are used to test the trained neural network. Table I shows the training time and errors comparisons between traditional RBF networks and sigmoid based RBF networks for training parity-N problem while N ranges from 5 to 10.

TABLE I. TRAIN COMPARISON

Parity-N Problem	Traditional RBF			Sigmoid based RBF		
	Training Time(s)	Train RMSE	Test RMSE	Training Time(s)	Train RMSE	Test RMSE
N=5	2.06	0.00	13.94	0.62	0.00	2.09
N=6	5.24	0.28	11.12	1.44	0.00	0.58
N=7	13.48	0.40	4.98	6.42	0.00	0.02
N=8	37.86	0.47	4.83	15.23	0.01	0.25
N=9	79.27	0.47	4.96	40.71	0.15	0.38
N=10	228.54	0.49	3.50	146.91	0.00	0.15

[a] For parity-N problem, increase hidden neurons from 1 to N

[b] Every time adding a new neuron, 200 iterations are trained

As mentioned in section III, sigmoid based RBF networks tune less parameters during training compared to traditional RBF networks. As a result, as the training time comparison shown in Table I, it costs less time to train sigmoid based RBF networks than traditional RBF networks. From the training and testing root mean square error (RMSE) comparison, it can be concluded that sigmoid RBF networks converge and generalize much better than traditional RBF networks. Fig. 13 shows a detailed comparison of training process while training parity-10 data set.

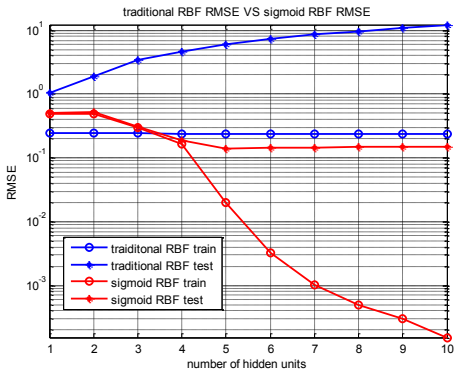

Fig. 13. Comparison for parity-10

VI. CONCLUSION

This paper implements a sigmoid based RBF neuron, which derives from traditional sigmoid neuron but can behave similar to RBF neuron. The proposed sigmoid based RBF neuron and neural networks composed of it are compared with traditional RBF unit and networks in detail. And it is shown that the proposed sigmoid based RBF neuron is more flexible than traditional RBF neuron. Finally these two types of networks are trained on a highly nonlinear peaks function approximation and a classic parity-N problem. The experiments results show that the sigmoid based RBF networks work more efficient than traditional RBF network, especially for multi-dimension classification and approximation. Traditional RBF networks work better on local error tuning in final stage of training.

REFERENCES

[1] J. Moody and C. J. Darken, "Fast learning networks of locally-tuned processing units," *Neural Computation* vol. 1, no. 2, pp. 281-294, 1989.

[2] E. Hartman, J. D. Keeler, and J. M. Kowalski, "Layered neural networks with Gaussian hidden units as universal approximations." *Neural Computationi,* vol.2, no. 2, pp.210-215, 1990.

[3] T. T. Xie, H. Yu and Bogdan Wilamowski, "Comparison between Traditional Neural Networks and Radial Basis Function Networks," in *Proc. 20th IEEE Int. Symp. Ind. Electron.-ISIE'11,* Gdansk, Poland, Jun. 27-30, 2011, pp. 1194-1199.

[4] B. M. Wilamowski and R. C. Jaeger, "Implementation of RBF Type Networks by MLP Networks," *IEEE International Conference on Neural Networks,* Washington, DC, June 3-6, 1996, pp. 1670-1675.

[5] Pao, Y. H. *Adaptive Pattern Recognition and Neural Networks,* Reading, Mass. Addison-Wesley Publishing Co. 1989.

[6] Wilensky G. and N. Manukian, (1992) The Projection Neural Networks, *International Joint Conference on Neural Networks,* II, 358-367.

[7] Wilamowski B., (1994) Fast Algorithms for Neural Network Design and Training, *Proc. of the Third Intelligent Information Systems Workshop,* Wigry, Poland, pp. 443-458, June 6-10.

[8] Ota Y. and B. Wilamowski, (1994) "Input data transformation for better pattern classification with less neurons," *Proc. of World Congress on Neural Networks,* San Diego, California, vol. 3, pp 667-672.

[9] J. Moody and C. J. Darken, "Learning with localized receptive fields," in *Proc. Connectionist Models Summer School,* D. Touretzky, G. Hinton, and T. Sejnowski, Eds., 1988, pp. 133-142.

[10] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.,* vol. 2, no. 2, pp. 302-309, Mar. 1991.

[11] S. Wu and T. W. S. Chow, "Induction machine fault detection using SOM-based RBF neural networks," *IEEE Trans. Ind. Electron.,* vol. 51, no. 1, pp. 183-194, Feb. 2004.

[12] M. J. L. Orr, "Regularization in the selection of radial basis function centers," *Neural Comput.,* vol. 7, no. 3, pp. 606-623, May 1995.

[13] H. Yu, T. T. Xie, S. Paszczynski, and B. M. Wilamowski, "Advantages of radial basis function networks for dynamic system design," *IEEE Trans. Ind. Electron.,* vol. 58, no. 12, pp. 5438-5450, Dec. 2011.

[14] K. Levenberg, "A method for the solution of certain problems in least squares," *Quart. Appl. March.,* vol. 2, pp. 164-168, 1944.

[15] B. M. Wilamowski and D. Hunter, Solving parity-N problems with feedforward neural network, *Proceedings of the IJCNN'03 International Joint Conference on Neural Networks,* pp. 2546-2551, Portland, OR, July 20-23, 2003.

[16] M. E. Hohil, D. Liu, and S. H. Smith, Solving the N-bit parity problem using neural networks, *Neural Networks,* 12, 1321-1323, 1999.

[17] R. C. Minnick, Linear-input logic, *IRE Transactions on Electronic Computers,* EC-10, 6-16, March 1961.

[18] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation,* Addison-Wesley, Reading, MA, 1991.