A Geometrical Representation of McCulloch–Pitts Neural Model and Its Applications

Ling Zhang and Bo Zhang

***Abstract*— In this paper, a geometrical representation of McCulloch-Pitts neural model is presented. From the representation, a clear visual picture and interpretation of the model can be seen. Two interesting applications based on the interpretation are discussed. They are 1) a new design principle of feedforward neural networks and 2) a new proof of mapping abilities of three-layer feedforward neural networks.**

***Index Terms*—Feedforward neural networks, measurable functions, neighborhood covering.**

I. Introduction

IN 1943, McCulloch and Pitts [1] first presented a mathematical model (M-P model) of a neuron. Since then many artificial neural networks have developed from the well-known M-P model [2], [3].

An M-P neuron is an element with n inputs and one output. The general form of its function is

$$y = \operatorname{sgn}(wx - \varphi)$$

where

$x = (x_1, x_2, \cdots, x_n)^T$—an input vector
$w = (w_1, w_2, \cdots, w_n)$—a weight vector
φ—a threshold

$$\operatorname{sgn}(v) = \begin{cases} 1, & v > 0 \\ -1, & v \le 0 \end{cases} \tag{1}$$

Rumelhart *et al.* [12] presented the concept of feedforward neural networks and their corresponding learning algorithm, back propagation (BP), which provided the means for neural networks to be practicable. In essence, the BP is a gradient descent approach. In BP, the node function is replaced by a class of sigmoid functions, i.e., infinitely differentiable functions, in order to use mathematics with ease. Although BP is a widely used learning algorithm it is still limited by some disadvantages such as low convergence speed, poor performance of the network, etc.

In order to overcome the learning complexity of BP and other well-known algorithms, several improvements have been presented. Since the node function $f(wx-\varphi)$ of the M-P model can be regarded as a function of two functions: a linear function $wx-\varphi$ and a sign (or characteristic) function $f(\cdot)$, generally, there are two ways to reduce the learning complexity. One is to replace the linear function by a quadratic function or a distance function. For example, the polynomial-time-trained hyperspherical classifier presented in [7] and [8] and the restricted Coulomb energy algorithm presented in [6] and [14] are neural networks using some distance function as their node functions. A variety of classifiers such as those based on radial basis functions (RBF) [9]–[13] and fuzzy classifiers with hyperboxes [15] and ellipsoidal regions based on Gaussian basis function [16], etc., use the same idea, i.e., replacing the linear function by a quadratic function. Although the learning capacity of a neural network can be improved by making the node functions more complicated, the improvement of the learning complexity would be limited due to the complexity of functions. Another way to enhance the learning capacity is by changing the topological structure of the network. For example, in [17] and [18], the number of hidden layers and/or the number of connections between layers are increased. Similarly, the learning capacity is improved at the price of increasing the complexity of the network. A detailed description of the above methods can be found in [19].

Now the problem is whether we can reduce the learning complexity of a neural network and still maintain the simplicity of the M-P model and its corresponding network. Some researchers tried to do so by directly using the geometrical interpretation of the M-P model, however, they have been unsuccessful. Note that $wx-\varphi=0$ can be interpreted as a hyperplane P in an n-dimensional space. When $(wx-\varphi)>0$, input vector x falls into the positive half-space of the hyperplane P. Meanwhile, $y = \operatorname{sgn}(wx-\varphi) = 1$. When $(wx-\varphi)<0$, input vector x falls into the negative half-space of P, and $y=-1$. In summary, the function of an M-P neuron can geometrically be regarded as a spatial discriminator of an n-dimensional space divided by the hyperplane P. Rujan *et al.* [4], [5] intended to use such a geometrical interpretation to analyze the behavior of neural networks. Unfortunately, when the dimension n of the space and the number m of the hyperplanes P (i.e., the number of neurons) increase, the mutual intersection among these hyperplanes in n-dimensional space will become too complex to analyze. Therefore, so far the geometrical representation has still rarely been used to improve the learning capacity of complex neural networks.

In order to overcome this difficulty, a new representation is presented as follows. First, assume that each input vector x has an equal length (norm). Thus all input vectors will be restricted to an n-dimensional sphere S^n. (In general cases, the

Manuscript received March 11, 1999; revised January 30, 1998, December 15, 1998, and March 11, 1999. This work was supported by the National Nature Science Foundation of China and the National Key Basic Research Program of China.

L. Zhang is with The State Key Lab of Intelligent Technology and Systems, Artificial Intelligence Institute, Anhui University, Anhui, China.

B. Zhang is with the Department of Computer Science, Tsinghua University, Beijing, China.

Publisher Item Identifier S 1045-9227(99)05482-X.

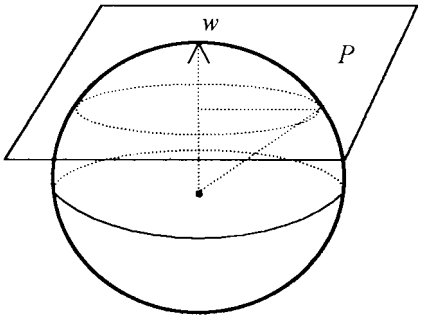

Fig. 1. A sphere neighborhood.

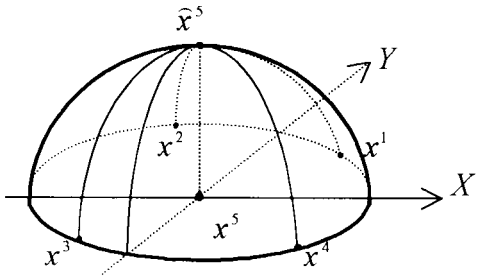

Fig. 2. Input vectors and their projections.

definition of S^n will be given later on.) Then, $(wx-\varphi)>0$ represents the positive half-space partitioned by the hyperplane P and the intersection between the positive half-space and S^n is called a "sphere neighborhood" as shown in Fig. 1. When input x falls into the region, output $y=1$, otherwise, $y=-1$.

If the weight vector w has the same length as input x, then w becomes the center of the sphere neighborhood, and $r(\varphi)$, a monotonically decreasing function of φ, becomes its radius. If the node function is a characteristic function $\sigma(v)$

$$\sigma(v)=\begin{cases}1, & v>0\\ 0, & \text{otherwise}\end{cases} \tag{2}$$

then the function $\sigma(wx-\varphi)$ of a neuron is a characteristic function of the sphere neighborhood on the sphere, i.e., an M-P neuron corresponds to a sphere neighborhood in an n-dimensional sphere. The preceding clear visual picture of an M-P neuron is a great help to the analysis of neural networks. In the preceding discussion, input vectors are assumed to have equal length. This is not the case in general. Although normalization could turn any input vector into a unit vector (a vector with a unit length), it will reduce the original n-dimensional space to an $(n-1)$-dimensional space. Some input information will be lost, but a way to circumvent the limitation is presented as follows.

Assume that the domain of input vectors is a bounded set D of an n-dimensional space. S^n is an n-dimensional sphere of an $(n+1)$-dimensional space.

Define a transformation $T: D\to S^n, x\in D$, such that

$$T(x)=(x,\sqrt{d^2-|x|^2})$$

where $d\geq \max\{|x||x\in D\}$.

Thus all points of D are projected upward on the S^n by transformation T (Fig. 2). Obviously, a neuron (w,φ) corresponds to a characteristic function of a "sphere neighborhood" on S^n with w as its center and $r(\varphi)$ as its radius.

Example 1: Five input vectors on a plane are given, i.e.,

$$\begin{aligned}D&=\{(1,1),(1,-1),(-1,1),(-1,-1),(0,0)\}\\&=\{x^1,x^2,x^3,x^4,x^5\},\end{aligned}$$

as shown in Fig. 2.
Construct a neural network to classify D into two classes

$$D^1=\{x^1,x^2,x^3,x^4\}\quad\text{and}\quad D^2=\{x^5\}.$$

Solution: First, D is projected on sphere S^2 so that each vector x^i in D is transformed into a three-dimensional vector y^i having the equal length $\sqrt{2}$. Then, we have

$$\begin{aligned}\tilde{D}=\{&y^1=(1,1,0),y^2=(1,-1,0),y^3=(-1,1,0),y^4\\&=(-1,-1,0),y^5=(0,0,\sqrt{2})\}.\end{aligned}$$

From the geometrical representation (Fig. 2), it is known that only one neuron (w,φ) is needed to partition $\tilde{D}$ into two classes, i.e., $\tilde{D}^1$ and $\tilde{D}^2$, where $w=(0,0,\sqrt{2})$ and $\varphi=\frac{\sqrt{2}}{2}$.

Note that in the original space D, at least two lines (neurons) are needed to partition D into D^1 and D^2. It is shown that the partition based on sphere neighborhoods is more powerful than the "half-space" partition. A hyperplane, i.e., the same linear form $(wx-\varphi)$, in the transformed curved (quadratic) space S^n is equivalent to a quadratic curved surface in the original flat space. Thus the partition by planes on a sphere space is equivalent to the partition by quadratic curved surfaces in the original flat space. For any given m points on S^n, m sphere neighborhoods can always be used to partition them such that each point falls into just one neighborhood. It is not always the case in the flat-space partition.

II. A Constructive Algorithm of Neural Networks

A framework of the proposed constructive algorithm of neural classifiers can be stated as follows. A set $K=\{l^t=(x^t,y^t),t=0,1,2,\cdots,s\}$ of training samples is given. Assume that output y^t in K has only k different values, i.e., the training samples will be classified into k classes. There is no loss of generality in assuming that the first k outputs have mutually different values. Let $I(t)$ be a set of indices of samples with the same output y^t, $t=0,1,\cdots,k-1$, i.e., $I(t)=\{i\mid y^i=y^t\}$, the input set corresponding to $I(t)$ be $p(t)=\{x^i\mid i\in I(t)\},t=0,1,\cdots,k-1$.

The design of a neural classifier can be divided into two stages: the input covering stage and the forward propagation algorithm stage.

A. Input Covering Stage

As a classifier (or an associative memory), the function of a neural network can be stated as follows. Given a set $K=\{l^t=(x^t,y^t),t=0,1,\cdots,s\}$ of training samples, after learning, the network should store the input pairs (x^t,y^t). Moreover, when the input becomes $x^t+\Delta^t$, the output y^t still remains the same, where Δ^t is regarded as a noise or an error.

For an M-P neuron (w, φ), when $w = x^t$ and $r(\varphi) = \Delta^t$, as mentioned before, it geometrically corresponds to a sphere neighborhood on S^n with x^t as its center and Δ^t as its radius. When an input falls into the neighborhood, the neuron always has outputs 1, otherwise, -1. An M-P neuron acts as a discriminator of an input class. Therefore, a network consisting of these neurons has the same function as a classifier.

The first design stage of a neural classifier can be transformed to an input vectors (a point set of an n-dimensional space) covering problem. Namely, a set of sphere neighborhoods is chosen to cover the inputs having the same output, i.e., belonging to the same class. Different classes of inputs are covered by different sets of sphere neighborhoods. Namely, an M-P neuron also corresponds to a covering on S^n.

It means that a set $\{C_j(t), j = 1, 2, \cdots, k_t\}$ of sphere neighborhoods (coverings) is chosen such that $C(t) = \bigcup c_j(t)$ covers every input $x^i \in p(t)$ and does not cover any input $x^i \notin p(t)$, and $C(t)'s, t = 0, 1, \cdots, k-1$ are mutually disjoint.

One possible covering approach is as follows.

Fix t, for any $x^i \in p(t)$
let

$$\left.\begin{aligned} d^1(i) &= \max_{j \notin I(t)} \{\langle x^i, x^j \rangle\} \\ d^2(i) &= \min_{j \in I(t)} \{\langle x^i, x^j \rangle > d^1(i)\} \\ d(i) &= \frac{d^1(i) + d^2(i)}{2} \end{aligned}\right\} \tag{3}$$

where $\langle x, y \rangle$ denotes the inner-product of x and y, and

$d^1(i)$ minimum "distance" between x^i and $x^j \notin p(t)$;
$d^2(i)$ maximum "distance" between x^i and x^j belonging to $p(t)$ and at a distance less than $d^1(i)$ from x^i.

Choose

$$w^i = x^i, \quad \varphi_i = d(i), \qquad i = 1, 2, \cdots, k_t. \tag{4}$$

Let

$$\begin{aligned} C_i(t) &= \{x \mid \langle x, x^i \rangle > \varphi_i\} \\ C(t) &= \bigcup_{i \in I(t)} C_i(t), \qquad t = 0, 1, \cdots, k-1. \end{aligned} \tag{5}$$

An M-P neuron (w^i, φ_i) described by (4) corresponds to a covering $C_i(t)$, a neighborhood on S^n with x^i as its center. The number of coverings $C_i(t)$ determines the number of neurons needed in the first layer of a neural classifier. The inputs are classified into different classes by the first layer (hidden layer). The second layer is used to form the given outputs y^t. From $C_i'(t)$, if a subcovering $C_i'(t)$ of $p(t)$ can be found, the amount of first-layer neurons will be reduced.

Therefore, the first design stage of a neural classifier is reduced to a minimum coverings problem. Since the minimum coverings problem is a classical one in many fields, e.g., Combinatorics [23], there are many well-known algorithms to deal with it. Neural network research can draw lessons from these algorithms. Therefore, a variety of covering approaches can be adopted. For example, it is not necessary to take the training samples as the centers of every coverings, etc. This can be seen later on.

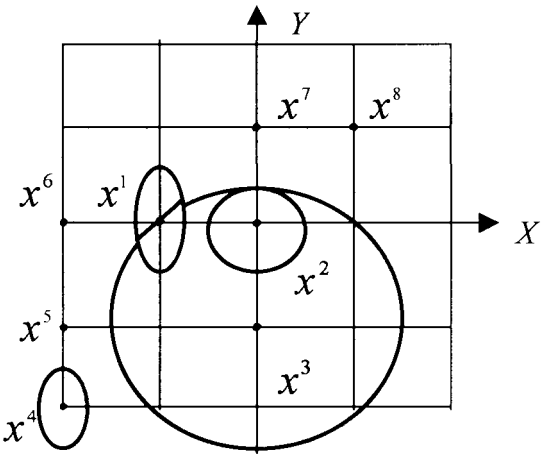

Fig. 3. Input vector and their coverings.

Example 2: Eight training samples (points) K are given in a plane (Fig. 3). Design a three-layer neural network to classify K into K_1 and K_2, where

$$\begin{aligned} K_1 = \{&x^1 = (-1, 0), x^2 = (0, 0), x^3 = (0, -1), \\ &x^4 = (-2, -2)\} \\ K_2 = \{&x^5 = (-2, -1), x^6 = (-2, 0), x^7 = (0, 1), \\ &x^8 = (1, 1)\}. \end{aligned}$$

Solution: Project the input vectors on the sphere, we have

$$\begin{aligned} K_1 = \{&y^1 = (-1, 0, \sqrt{7}), y^2 = (0, 0, \sqrt{8}), y^3 = (0, -1, \sqrt{7}), \\ &y^4 = (-2, -2, 0)\} \\ K_2' = \{&y^5 = (-2, -1, \sqrt{3}), y^6 = (-2, 0, 2), y^7 = (0, 1, \sqrt{7}), \\ &y^8 = (1, 1, \sqrt{6})\}. \end{aligned}$$

According to (3) and (5), we can find a set $C(1) = \{C_1(1), C_2(1), C_3(1), C_4(1)\}$ of coverings which cover every point in K_1' and do not cover any point in K_2', where $C_1(1), C_2(1), C_3(1)$, and $C_4(1)$ cover y^1, y^2, y^3, and y^4, respectively. The minimum coverings of K_1' are $C_3(1)$ and $C_4(1)$. The projection of the coverings on the XY plane is shown in Fig. 3. Two neurons are needed in the first layer, specifically,

$$\begin{aligned} w^1 = y^3 = (0, -1, \sqrt{7}), &\quad \varphi_1 = 6.5 \\ w^2 = y^4 = (-2, -2, 0), &\quad \varphi_2 = 7. \end{aligned}$$

The neuron in the second layer is an OR element. The network is shown in Fig. 4.

B. Forward Propagation Algorithm

Given a set of training samples

$$K = \{l^{(t)} = (x^t, y^t), t = 0, 1, \cdots, p-1\}$$

by a covering approach, the input vectors have been covered by a set of coverings based on their classification. Assume that a set C of $p-1$ coverings is obtained as follows:

$$C = \{C(i), i = 1, 2, \cdots, p-1\}.$$

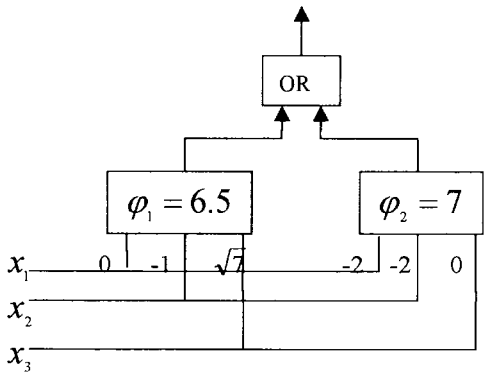

Fig. 4. A neural network example.

Assuming that x^i is the center of covering $C(i)$ and y^t is its corresponding output, $i = 1, 2, \cdots, p-1$. For simplicity, y^t's are assumed to be mutually different, i.e., one class corresponds to only one covering. (When one class corresponds to a set of coverings, the constructive approach is almost the same.) The domain of each component of x^i (y^t) is assumed to be $\{1, -1\}$. Then a three-layer neural network, i.e., two layer elements, can be constructed by the Forward Propagation (FP) algorithm presented in [20].

In contrast to the well-known BP algorithm, the FP algorithm starts from the first layer to the last one, so it is called forward propagation. When the components of input and output vectors are real-valued rather than $\{-1, 1\}$, by making small changes, the FP algorithm is still available.

III. The Comparison of Results

Since the minimum coverings problems are known to be NP-hard, the key point in the design of the proposed neural network is to find a feasible covering approach such that a satisfactory result can generally be obtained. A covering method called covering-deletion algorithm and its computer simulation results were presented in [20]. In order to show the performance of the proposed neural network, one of the problems, two-spiral discrimination, and its input covering result are shown in Fig. 5.

Example 3 (Two-Spiral Discrimination Problem): Two spirals K_1 and K_2 (in polar coordinates)

$$\begin{aligned} K_1&: \quad \rho = \theta \\ K_2&: \quad \rho = \theta + \pi \\ &\quad \frac{\pi}{2} \le \rho \le 6\pi. \end{aligned}$$

The projection of ten coverings on the XY plane is shown in Fig. 5. The correct classification rate for 20 000 training samples is 100%. The correct rate for 10 000 testing samples is 99.95%. Compared to the results in [17], [21], and [22], we can see that the classification of two spirals based on BP fails [21], and it takes 3000 iterations and only has 89.6% correct classification rate by using the generating-shrinking algorithm presented in [22]. Fahlman [17] uses a cascade network with more than ten layers to treat the same problem, and part of his results is presented in [18]. (It can be seen that the result presented in Fig. 5 is better than that presented in [18, Fig. 6.3.4]).

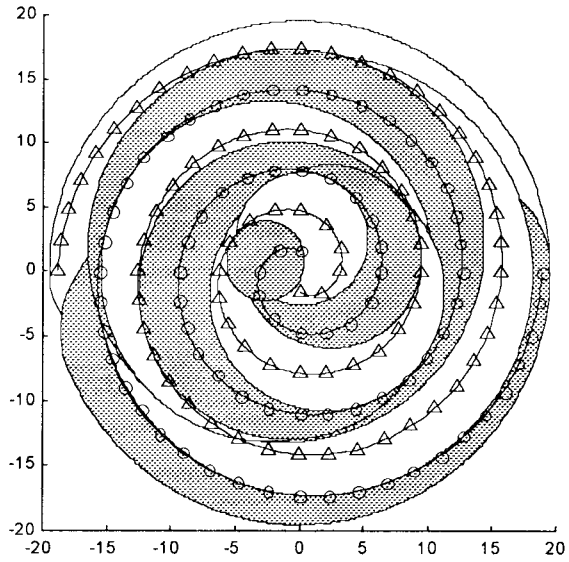

Fig. 5. Two-spirals problem and its input coverings.

IV. Mapping Abilities of Three-Layer Neural Networks

A well-known theorem about the mapping abilities of feedforward neural networks is shown as follows.

Theorem: Any function in $L^2(D^n)$ can be approximated to any desired degree of accuracy by a three-layer feedforward neural network, where D^n is a unit hypercube in an n-dimensional Euclidean space.

Many researchers have proved some similar theorems [8]–[10] from various approaches. A new visual proof of the theorem will be given below by using the geometrical interpretation of M-P neurons.

Definition 1: Assume that D is a bounded measurable set on S^n, where S^n is an n-dimensional sphere in an $(n+1)$-dimensional space. Function $f(x) : D \to R$ and $f(x) = \sum_{i=1}^{m} a_i d(D_i)$, where $d(D_i)$ is a characteristic function in D_i, and D_i is a measurable set. Function $f(x)$ is called a step function. If each D_i is a sphere neighborhood, then $f(x)$ is called a step function on sphere neighborhoods.

Definition 2: Assume that D is a bounded measurable set in R^n, and function $f(x) : D \to R$ is a bounded measurable function. A function $g(x) : D \to R$, if there exists (δ, ε) such that $\mu(\{x \mid |f(x) - g(x)| > \varepsilon\}) < \delta$ is called a (δ, ε)-measure approximation of $f(x)$, where μ is a measure in R^n.

In this paper, the "measurable" always means "Lebesque measurable."

Since D_i is a sphere neighborhood, each characteristic function $d(D_i)$ can be represented by a neuron A_i based on the above geometrical interpretation of the M-P neuron. If the first layer of a feedforward network is composed of these neurons A_i, $i = 1, 2, \cdots, m$, and the second layer is made by a linear element, then the network can realize the step function $f(x)$. From any textbook on Theory of Real Variable Functions [27],

it is known that any function on L^2(L^2-integrable function) can be approximated by a step function. Therefore, to prove the theorem, all we need to do is to confirm that any measurable set can be approximated by a set of sphere neighborhoods. This is also a well-known result in Theory of Real Variable Functions. So the theorem can be proven by slightly changing the well-known results. We have

Basic Theorem: D is a bounded measurable set on S^n and $f(x): D \rightarrow R$ is an almost bounded measurable function. For any δ, $\varepsilon > 0$, there exists a three-layer feedforward network such that its function $g(x)$ is a (δ, ε)-measure approximation of $f(x)$.

Corollary 1: D is a bounded measurable set in R^n, and $f(x): D \rightarrow R$ a L^2-integrable function. For any $\varepsilon > 0$, there exists a three-layer neural network such that $d_{L^2}(f(x), g(x)) < \varepsilon$, where $g(x)$ is the mapping function of the network.

This is the theorem presented in Hecht-Nielsen [26].

V. Conclusion

The key points of our work are the following. The original input space is transferred into a quadratic space, and the well-known (point set) covering method can be applied to perform partition of data in the transformed space. At the same time, the simplicity of the M-P model and its corresponding network still holds true. There is no need either to increase the complexity of node functions or the complexity of network's structure. A feasible covering-deletion design algorithm is presented as well, and computer simulation results show that the proposed neural network is quite efficient.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in neurons activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.

[2] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, vols. 1, 2. Cambridge, MA: MIT Press, 1986.

[3] R. J. Mammone and Y. Y. Zeevi, *Neural Networks: Theory and Applications*. San Diego, CA: Academic, 1991.

[4] P. Rujan and M. Marchand, "A geometric approach to learning in neural networks," in *Proc. IJCNN'89*, Washington, DC, vol. II, pp. 105–110.

[5] U. Ramacher and M. Wesseling, "A geometric approach to neural network design," in *Proc. IJCNN'89*, Washington, DC, vol. II, pp. 147–154.

[6] P. W. Cooper, "A note on adaptive hypersphere decision boundary," *IEEE Trans. Comput.*, pp. 948–949, 1996.

[7] A. Roy and S. Mukhopadhyay, "Pattern classification using linear programming," *ORSA J. Computer*, vol. 3, no. 1, pp. 66–80, 1991.

[8] S. Mukhopadhyay, A. Roy, L. S. Kim, and S. Govil, "A polynomial time algorithm for generating neural networks for pattern classification: Its stability properties and some test results," *Neural Comput.*, vol. 5, no. 2, pp. 317–330, 1993.

[9] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," AI Memo 1140, MIT, Cambridge, MA, 1989.

[10] M. T. M. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, "On the training of radial basis function classifiers," *Neural Networks*, vol. 5, pp. 593–603, 1992.

[11] D. S. Broomhead and D. Lowe, "Multivariate functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.

[12] S. Lee and R. Kil, "Multilayer feedforward potential function network," in *Proc. IEEE 2nd Int. Conf. Neural Networks*, San Diego. New York: IEEE, 1988, vol. I, pp. 161–171.

[13] J. Moody and D. Darken, "Fast learning in networks of locally-tuned processing unit," *Neural Comput.*, vol. I, no. 2, pp. 281–294.

[14] D. L. Reilly and L. N. Cooper, "An overview of neural networks: Early models to real world systems," in *An Introduction to Neural and Electronic Networks*, S. F. Zornetzer, J. L. Davis, and C. Lau, Eds. San Diego, CA: Academic, 1990, pp. 227–246.

[15] S. Abe and M.-S. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 1, pp. 18–28, 1995.

[16] S. Abe and R. Thawonmas, "A fuzzy classifier with ellipsoidal regions," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 3, pp. 358–368, 1997.

[17] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems*, vol. 2, D. S. Tourdtzhy, Ed. San Mateo, CA: Kaufmann, 1990, pp. 524–532.

[18] K. J. Lang and M. L. Witbrock, "Learning to tell two spirals apart," in *Proc. 1988 Connectionists Models Summer Schools* Pittsburgh, PA, 1988, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. San Mateo, CA: Kaufmann, pp. 52–59.

[19] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.

[20] L. Zhang and B. Zhang, "Neural network based classifiers for a vast amount of data," in *Proc. 3rd Pacific-Asia Conf. PAKDD-99, Methodologies for Knowledge Discovery and Data Mining*, Beijing, China, Apr. 26–28, 1999, pp. 238–246.

[21] E. B. Baum and K. J. Lang, "Constructing hidden units using examples and queries," in *Neural Information Processing Systems*, vol. 3, R. P. Lippman *et al.*, Eds. San Mateo, CA: Kaufmann, 1991, pp. 904–910.

[22] Q. C. Chen *et al.*, "Generating-shrinking algorithm for learning arbitrary classification," *Neural Networks*, vol. 7, pp. 1477–1489, 1994.

[23] F. S. Roberts, *Applied Combinatorics*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[24] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[25] M. Arai, "Mapping abilities of three-layer networks," in *Proc. IJCNN'89*, Washington, DC, vol. 1, pp. 419–423.

[26] R. Hecht-Nielsen, "Theory of the back propagation neural networks," in *Proc. IJCNN'89*, Washington, DC, vol. 1, pp. 593–605.

[27] N. Bourbaki, *Fonctions d'une Variable*. Paris, France: Hermann, 1951.