FC Networks for Prediction Applications

Subhash Kak[*]

February 13, 2001

Abstract

We present a generalization of the corner classification approach to training feedforward neural networks that allows rapid learning of non-binary data. These generalized networks, called *FC* networks, are compared against Backpropagation (BP) and Radial Basis Function (RBF) networks and shown to have excellent performance for prediction of time-series and pattern recognition. FC networks do not require iterative training and they can be used in many business applications where fast, nonlinear filtering provides an advantage.

[*]Department of Electrical & Computer Engineering, Louisiana State University, Baton Rouge, LA 70803-5901, Email: `kak@ee.lsu.edu`

1 Introduction

The Backpropagation (BP) and Radial Basis Function (RBF) neural networks, that are used extensively for a variety of signal processing applications, require iterative training [1, 9]. BP networks sometimes don't converge or take too long to train to be useful in real-time applications. BP networks have been proposed as models of biological memory, but given the time it takes these networks to learn, it is clear they could never learn short-term memory, which is instantaneous. It was the motivation to model such memory that led to the development of the corner classification (CC) family of networks [3-7, 14] that learn instantaneously and have very good generalization performance [11]. It has been shown that these networks are hardware friendly and suited for implementation in reconfigurable computing using fine grained parallelism [16]. Although the CC networks have found many applications in engineering and business [8, 12], they suffer from the disadvantage that their input and output must be discrete.

Another disadvantage of the CC networks is that the input is best presented in a unary code, which increases the number of input neurons considerably. Furthermore, the degree of generalization achieved at each trained node is, in the non-adaptive version of the CC network, kept constant. In reality, the data might require that the amount of generalization varies from node to node. These characterisics somewhat limit the applicability of CC networks. An interative version of the CC algorithm that does provide varying degree of generalization has been devised [13] but it is not instantaneous.

In the present paper we present a generalization of the CC network that we call the FC (for *fast* classification) network [15]. This network can operate on real data directly. It reduces to the CC network when the data is binary and the amount of generalization is fixed.

The FC network maps data to the nearest neighbor within whose sphere of generalization the data falls, and if that is not the case it performs an interpolation based on k nearest neighbors. As k becomes large, the FC generalization can be shown to approach the Bayes performance.

2 FC network structure

Figure 1 shows the configuration of a general FC network. It uses the fully-connected feedforward network architecture consisting of a layer of input nodes, a layer of hidden neurons, a rule base, followed by an output layer. The number of output neurons is determined by the problem specifications. For simplicity, only one output neuron has been shown, although the network can be easily extended to multiple outputs. In fact, a network with multiple output neurons can always be separated into multiple single-output networks.

Input data are normalized to the range $[0, 1]$ and presented to the network in the form of an R-element long continuous-valued vector $x = (x_1, x_2, ..., x_R)$, where R is determined by the problem specification. Like the CC network, the number of hidden neurons S in an FC network is equal to the number of training samples the network is required to learn.

Each hidden neuron i ($i = 1, 2, ..., S$) is associated with a weight vector $w_i = (w_{i,1}, w_{i,2}, ..., w_{i,R})$. The convention used in assigning indices to these weights is as follows. The first index indicates the particular hidden neuron destination for that weight. The second index indicates the source of the signal fed to that hidden neuron. Thus, $w_{i,j}$ is the weight from source x_j in the vector to the ith hidden neuron. For the output layer, since there is only one destination neuron, the destination index is dropped.

Each hidden neuron i first computes the normalized Euclidean distance d_i between its weight vector w_i and the test vector x. This distance together with r_i, the radius of generalization for this hidden neuron, constitute the inputs to the activation function F which determines the hidden neuron output h_i. The output of all hidden neurons taken together forms the distance vector $h = (h_1, h_2, ...h_S)$ that gives a measure of similarity between the test vector and each of the training vectors the network has already learned.

The rule base enables the FC network to generalize. It consists of a set of IF-THEN rules that operates on the distance vector h to produce a membership grade vector $\mu = (\mu_1, \mu_2..., \mu_S)$ that indicates to what degree the test vector x belongs to each of the network output classes. The output neuron then computes the dot product between output weight vector $u = (u_1, u_2, ..., u_S)$ and the membership vector μ to produce the generalized network output y corresponding to test vector x.

Training of an FC network involves two distinct steps. The first step determines the input and output weights while the second step finds the radius of generalization for each hidden neuron. It shall be seen that each of these steps requires just one presentation of the training data set. This gives the network its instantaneous learning capability.

The Hidden Neurons

Unlike the neurons in a CC4 network [7, 14], the input vector x, the weight vector w, as well as the scalar output h in the FC neuron are all continuous-valued quantities. Quantity d is the Euclidean distance between x and w. The parameter r is called the radius of generalization, a term borrowed from the CC4 network. As shall be seen later, its purpose is to allocate to this hidden neuron, an area of generalization in the input space with radius r and its center at w. Unlike the CC4 network where the same r applies to the whole network, the radius of generalization in an FC network is individually determined for each hidden neuron during training. Further, the training process used in an FC network ensures that there is no overlapping between the area of generalization for each training sample.

The activation function F takes two scalar inputs, r and d, and produces a scalar output h for the hidden neuron. Its operation can be described mathematically as

$$\begin{aligned} h &= 0 \quad d \leq r \\ h &= d \quad d > r \end{aligned} \tag{1}$$

The effect of this activation function is to replace any distance between x and w less than or equal to r by zero. Equivalently, it may be viewed as a fuzzification of the location occupied by w in the input space from a crisp point with zero radius into a fuzzy area with radius r. From this point of view, any test vector x within a distance r from w will be indistinguishable from w. If w, the weight vector for this hidden neuron, is made identical to a training vector v, then x will also be indistinguishable from v, and will therefore be classified into the same output class as v. As shall be seen later, the training process does make the weight vector for each hidden neuron equal to its training vector. The process of fuzzification of the location of w is one of two mechanisms by which an FC network performs generalization.

The Rule Base

Another mechanism by which an FC network performs generalization is to treat a test vector x as having fuzzy membership grades in output classes of its nearest neighbors. The function of the rule base is to ascertain these membership grades, i.e., the degree to which x belongs to these output classes. In an FC network, as in a CC4 network, each training vector is mapped to an output. However, in a CC4 network, a test vector either belongs to or does not belong to an output class. In other words, membership grade in a CC4 network is crisp, taking on only 0 or 1 as its possible values. If the distance between the test vector and a training sample is less than the radius of generalization of the network, its membership grade in that output class is 1. Otherwise, its membership grade in that output class is 0. By contrast, membership grade in an FC network is fuzzy. It can take on any value between 0 and 1 depending on the distance from the test vector to the training sample. Value 0 indicates non-membership while 1 indicates full membership.

Only the k training samples nearest to the test vector in the Euclidean sense are considered when assigning membership grades. The value of k is typically a small fraction of the size of the training set. Membership grades are normalized, i.e., the sum of all membership grades equals 1. By comparison, in a CC4 network, membership grades are not always normalized. If the test vector falls within the radius of generalization of two or more training samples, it takes on membership grade of 1 in each of their output classes. Its total membership grade thus adds up to more than 1. Similarly, if it does not fall within the radius of generalization of any training sample, its total membership grade equals zero.

The rule base consists of two IF-THEN rules that assign fuzzy membership grades μ_i based on the outputs of the hidden neurons. Let m be the number of hidden neuron outputs h_i that equal 0. As shall be seen later, the training process ensures that the area of generalization for each training sample does not overlap with that for another sample. A test vector can therefore fall within the area of generalization of at most one training sample. This means that m can only be 0 or 1. The two IF-THEN rules within the rule base are as follows:

Rule 1: IF m = 1, THEN assign μ_i using single-nearest-neighbor (INN) heuristic

Rule 2: IF m = 0, THEN assign μ_i using k-nearest-neighbor (kNN) heuristic

Note that, to avoid confusion, the 1NN and kNN rules are referred to as heuristics. The operation of the rule base shall be discussed in the section below.

3 Training of the FC Network

Training of the FC network involves two separate steps. In the first step, input and output weights are prescribed simply by inspection of the training input/output pairs. For the ith training input $(i = 1, 2, ..., S)$ presented to the network, input weight w_{ij} is made equal to x_j $(j = 1, 2, ..., R)$ while the output weight u_i is made equal to the corresponding target output y_i; i.e.,

$$w_{ij} = w_j, u_i = y_i, \ (i = 1, 2, ..., S; j = 1, 2, ..., R) \tag{2}$$

In the second step of the training process, the radius of generalization for each hidden neuron is determined. This is accomplished by another presentation of the training set. When the ith training input is presented, d_i will be zero, since the distance from training vector i to itself is zero. The smallest non-zero distance, d_{min}, is thus the distance to its nearest neighbor, say training vector j. The radius of generalization for the ith hidden neuron, r_i, is then set to $d_{min}/2$. This will ensure that the area of generalization of hidden neuron i will not overlap with another hidden neuron. These two steps complete the training of the FC network. Learning in an FC network is therefore instantaneous, with each training sample presented to the network only twice.

Note that although the method used to determine r ensures no overlapping, it does not guarantee complete coverage of the input space. This is because the nearest neighbor of training vector j is not necessarily training vector i, in which case r_j will be smaller than r_i. However, this is of no concern, as Rule 2 in the rule base is designed to handle its situation.

Generalization by Fuzzy Membership

With the completion of the training process, the FC network is ready to be deployed. When a test vector x is presented to the network, the outputs of the hidden neurons form the distance vector $h = (h_1, h_2, ..., h_s)$ that gives a measure of similarity between the test vector and each of the training vectors the network has already learned. The rule base operates on this distance vector and produces a set of membership grades $\mu = (\mu_1, \mu_2, ..., \mu_s)$ according to the 1NN heuristic or the kNN heuristic. The output neuron then computes the dot product between the output weight vector $u = (u_1, u_2, ...u_s)$ and the membership grade vector to produce the generalized network output y corresponding to test vector x. The network output y can thus be written as

$$y = \sum_{(i=1,S)} \mu_i u_i \tag{3}$$

The operation of the 1NN heuristic and the kNN heuristic shall be described below.

1NN Heuristic The 1NN heuristic is used when exactly one element in the distance vector h is 0, i.e., when the test vector falls within the area of generalization of a training sample. It therefore belongs to the output class of that training sample with a membership grade of 1 and to all other classes with a membership grade of 0. Thus, if h_j is zero, the membership grades are assigned as follows:

$$\mu_i = 1 \ if \ i = j (i = 1, 2, ..., S)$$

$$\mu_i = 0 \ if \ i \neq j \tag{4}$$

kNN Heuristic The kNN heuristic is used when none of the elements in the distance vector h is equal to 0, i.e., when the test vector x does not fall within the area of generalization of any training sample. The test vector is assigned fuzzy memberships of classes whose training samples are its k nearest neighbors. These training samples correspond to the k hidden neurons with the smallest outputs. Membership grades for all other classes whose training samples are not in the set of k nearest neighbors of x are set to zero. The discussion below begins with $k = 2$ and then generalizes k to higher values.

Consider the point X, representing a test vector x, and two points A and B, representing two training samples, a and b, which are the nearest neighbors of x. Let the distances from x to a and b be a and b respectively. The triangular membership functions $\mu_{close\ to\ a}$ and $\mu_{close\ to\ b}$ characterize the change in membership grades of x as its distances from a and b change. For example, if $a = 0$, then x falls within the area of generalization of a. Its membership grade in a's class is 1 and that in b's class is 0. The kNN heuristic then reduces to the 1NN heuristic. A similar situation exists if $b = 0$. In between these two extreme cases, x's membership grades in these two classes, denoted by $\mu(x, a)$ and $\mu(x, b)$ respectively, change linearly as

$$\mu(x, a) = b/(a + b)$$

$$\mu(x, b) = a/(a + b) \tag{5}$$

This is the kNN heuristic for $k = 2$. Note that, for clarity in presentation, the notation μ_i used earlier has been changed to $\mu(x, a)$.

The above results can be generalized to higher values of k. The above equation can be rewritten as

$$\mu(x, a) = (1/a)/(1/a + 1/b)$$

$$\mu(x, b) = (1/b)/(1/a + 1/b) \tag{6}$$

It is easy to see that, for example, for $k = 3$ and c as the third nearest neighbor at a distance c from x, the respective membership grades are

$$\mu(x, a) = (1/a)/(1/a + 1/b + 1/c)$$

$$\mu(x, b) = (1/b)/(1/a + 1/b + 1/c)$$

$$\mu(x, c) = (1/c)/(1/a + 1/b + 1/c) \tag{7}$$

Results for higher values of k can be derived in a similar fashion. It is easy to verify that the membership grades sum up to 1 in each case.

For simplicity, the above discussion is based on the triangular membership function. Other membership functions are of course possible. It is defined mathematically as:

$$\mu(x;p,q) = \begin{cases} 0 & \text{for } x < p \\ 2((x-p)/(q-p))^2 & \text{for } p \leq x < (p+q)/2 \\ 1 - 2((x-q)/(q-p))^2 & \text{for } (p+q)/2 \leq x < q \\ 1 & \text{for } x \geq q \end{cases} \tag{8}$$

The membership grades $\mu(x, a), \mu(x, b)$, etc., would vary with the squares of the respective distances when this quadratic membership function is used. Similarly, other distance metrics such as the city block distance could be used in place of the Euclidean distance metric. Preliminary testing indicates that the performance of the network is not seriously affected by the choice of distance metric and membership function.

To summarize, the network can be trained with just two passes of the training samples. The first pass assigns the synaptic weights for the input and output layers. The second pass determines the radius of generalization r for each training sample. The network exhibits fuzziness in two regards; (i) by fuzzification of the location of each training vector in the input space; and (ii) by assigning fuzzy memberships of output classes to new input vectors. The notion of radius of generalization for each training vector provides a basis for the network to switch between a 1NN classifier and a kNN classifier during generalization. The network behaves like a 1NN classifier when the input vector falls within the area of generalization of a training vector, and a kNN classifier otherwise. This enables the network to benefit from the strength of both classifiers.

Let us view the operation of FC network from various perspectives. It can be shown that the transformation performed by an FC network is consistent with Cover's theorem on separability of patterns. This may be seen with an example featuring the classic Exclusive-OR problem. The FC network meets the specification of traditional curve fitting in its strict sense in that the function implemented by the FC network is constrained to pass through all data points given in the training set. The FC network operating as a kNN classifier can be viewed as kernel regression if $k = S$ and the weighting function is chosen as the membership function. Along a similar line of argument, it may be seen that the FC network operating as a kNN classifier can be made to behave like an RBF network if $k = S$ and the Gaussian distribution function is chosen as the membership function.

The performance of the FC network as a 1NN and a kNN classifier may be cast in probabilistic terms to see its relationship to the Bayes classifier. Unfortunately, while the analysis is relatively straightforward for a two-class problem where classification is done by voting, the more general cases where classification is done by fuzzy membership and where the problem involves multiple classes are not so amenable to a similar kind of treatment. The discussion on the FC network as a kNN classifier provides some insight regarding the choice of the value of k. While theoretical analysis suggests that k should be made large to improve performance, practical consideration resulting form finite sample size dictates that k be kept to a small fraction of the sample size. That the choice of a small k is confirmed in experiments involving time series prediction and pattern classfication.

Example Consider the following input-output training samples:

Sample	Input				Output
1	2	3	4	-1	7
2	1	-1	2	-3	4
3	3	0	1	8	9

The Euclidean distances between the samples are: $d_{12} = 5,\ d_{13} = 10,\ d_{23} = 11.27$. Therefore, the radii of generalization to be associated with the three hidden nodes are 2.5, 2.5, 5, respectively.

Now, consider an input vector $X = (1, 2, 3, 4)$, whose output we wish to compute. This vector is at the Euclidean distance of $d_1 = 5.29$, $d_2 = 7.68$, and $d_3 = 5.29$ from the three stored vectors. Since these distances are larger than the radii of generalization of the three vectors, we compute the interpolation weights $\mu_i = \frac{1/d_i}{1/d_1+1/d_2+1/d_3}$

This gives us $\mu_1 = 0.372,\ \mu_2 = 0.256,\ \mu_3 = 0.372$. So, the output is: $0.372 \times 7 + 0.256 \times 4 + 0.372 \times 9 = 6.976$.

4 Experimental Results

Although we have tested the FC network for a variety of signal processing applications, we confine ourselves here to time-series prediction and pattern recognition.

Time series perdiction

The time-series prediction problem is of interest as in electric load demand forecast, traffic volume forecast, equalization in communications, and prediction of stock prices, currency and interest rates. Here we describe the performance of the FC network using two benchmark chaotic time-series with different characteristics. The first is the Henon map [2] time-series, which is volatile, and the second is the Mackey-Glass time-series, which changes direction more slowly.

In the examples on chaotic time-series prediction, the value of k that gave the best performance for a sample size of 500 is 5. Thus k is just 1% of the sample size. The FC network was compared to the RBF and the BP networks and it was found that the generalization performance was comparable. Further, the scalability of the performance of the FC network with respect to the changes in window size and sample size was found to be superior to the RBF network.

Henon map The one-dimensional Henon map is:

$$x(t) = 1 - ax^2(t-1) + bx(t-2)$$

A chaotic regime occurs for $a = 1.4$ and $b = 0.3$. This equation is used to generate a total of 554 points out of which 500 training samples are derived from the first 504 points while the remaining 50 points are used for out-of-sample testing of the network after training.

In this experiment, the window size is set to 4 and prediction is made for one point ahead. In other words, each training sample is formed by a sliding window consisting of four consecutive points along the time series as input with the fifth point as the target output. For example, the first training sample consists of $x(1), x(2), x(3)$, and $x(4)$ as input while the target output is $x(5)$. The second training sample used $x(2), x(3), x(4)$, and $x(5)$ as input while the target output is $x(6)$, and so on. The last training sample therefore consists of $x(500), x(501), x(502)$ and $x(503)$ as input with $x(504)$ as target output. The FC network architecture needed is thus 4-500-1.

The out-of-sample testing points are similarly organized into test vectors. For example, the first test vector is made up of $x(501), x(502), x(503), x(504)$. The actual ouput corresponding to this test vector is $x(505)$. After the network has been trained using the sample set, the fifty test vectors are presented one at a time and the network is required to predict the output corresponding to each of the test vectors. The predicted output is then compared against the actual output and the difference between the two is used to compute the total prediction error expressed in terms of a sum-of-squared error (SSE) cumulated over the fifty test points. Table 1 shows the error for various values of k.

Table 1: Henon map time series prediction using FC network

k	SSE
1	0.009054
2	0.007567
3	0.003381
4	0.002971
5	0.002481
6	0.002486
7	0.003401

The table shows that the best performance is obtained when $k = 5$. It can be seen that the predicted points match the actual data points very well. In particular, all turning points in the time series have been correctly predicted. Points before $x(505)$ are not shown since the mathc between actual data and network output is exact, i.e., training error is zero in a FC network.

Mackey-Glass time series The Mackey-Glass (MG) equation is a non-linear time delay differential equation originally developed for modeling whie blood cell production. Its discrete time representation can be written as

$$x(t+1) = (1-B)x(t) + Ax(t-D)/[1 + x^C(t-D)]$$

where A, B and C are constants and D is the time delay parameter. Under a suitable choice of these numbers, the resultant time series will exhibit chaotic behavior. The popular case with $A = 0.2, B = 0.1$ and $C = 10$, and the delay parameter D set to 30 is selected here.

In accordance with previously published work, the MG equation is used to generate a continuous sequence of data points. The first 3000 points are discarded to allow initialization transients to decay. The remaining data points are then sampled once every six points to obtain the actual time series used for this experiment. The window size is 6 and prediction is made for one point ahead. A total of 500 samples are used for training. The FC network architecture required to learn this time series is therefore 6-500-1. Here also our experiments showed that the SSE was minimized for $k = 5$.

The performance of FC network is excellent. This is seen most clearly when we consider the question of performance scalability. Here the FC network and RBF network are optimized for the sample size of 500 and window size of 4. Then the window and the sample size are allowed to change without reoptimization of the k parameter for the FC network and the spread constant for the RBF network. The results for the SSE are in Table 2.

Table 2. Henon map time series prediction: performance scalability

Window size	4	6	8	10
Sample size=500				
SSE for FC	0.00248	0.01	0.07	0.17
SSE for RBF	0.00096	0.40	1.84	9.10
Sample size=400				
SSE for FC	0.0077	0.03	0.31	0.70
SSE for RBF	0.0002	0.60	8.15	23.45
Sample size=300				
SSE for FC	0.017	0.054	0.28	0.73
SSE for RBF	0.23	4.01	9.51	34.41
Sample size=200				
SSE for FC	0.035	0.13	0.44	0.97
SSE for RBF	0.041	3.27	8.35	73.52

Table 3 presents the corresponding results for the MG time series prediction performance.

Table 3. Mackey-Glass time series prediction: performance scalability

Window size	4	6	8	10
Sample size=500				
SSE for FC	0.90	0.144	0.07	0.08
SSE for RBF	532.77	0.1563	0,13	0.21
Sample size=400				
SSE for FC	0.98	0.20	0.13	0.20
SSE for RBF	3510.5	4.28	0.19	0.44
Sample size=300				
SSE for FC	1.42	0.50	0.34	0.33
SSE for RBF	33.65	3.39	1.04	2.55
Sample size=200				
SSE for FC	1.88	1.20	0.92	1.17
SSE for RBF	443.55	4.25	4.49	6.08

From these tables it can be seen that the performance of the FC network remains good and reasonably consistent throughout all window and sample sizes while that of the RBF network is adversely affected by changes in the window size or sample size or both. Indeed, the performance of the RBF network can become erratic for certain combinations of these parameters. This implies that an FC network designed for one window size or one sample size is generally applicable to other window sizes and sample sizes. One the other hand, each time the window size or sample size changes a new RBF network may be required. We expect the same drawback to hold for BP networks as well, which were not included in the comparisons above because of the excessive time required to train them. It is clear, therefore, that the FC network is very much easier to use in practice compared to the other networks.

Table 3 also reveals another interesting feature that a value of 6 is not the best window size to use for the FC network in modeling this Mackey-Glass time series. A window size of 8 gives better performance for all sample sizes tested. This is probably due to the low volatility in this time series which is better modeled by a longer window size.

Pattern recognition

Another problem on which experiments were done is that of pattern recognition. In one of these experiments, a spiral pattern in a 32-by-32 area was considered. Input to the network consists of the row and column coordinates of the training samples which are integers ranging from 1 to 32. The networks have two output neurons, one for each class. If the training sample belongs to the white region, the corresponding target output is the two-bit binary vector (1,0). On the other hand, if the training sample belongs to the black region, its target output is the vector (0,1). The FC network architecture required is therefore 2-256-2.

This FC network was compared with BP and RBF networks. For the BP network, it required search for the best number of hidden neurons to minimize the error. We tried hidden neurons varying from 20 to 80 and found that the error was minimized for this number to be 60. Table 4 provides a summary of the results.

Table 4. Two-class spiral pattern classification

Network Type	Network Configuration	Error
FC	2-256-2	47
BP	2-60-2	48
RBF	2-256-2	62

We find the FC network performs better than BP and RBF networks. This superior performance shows up even more clearly in performance scalability tests.

5 Conclusions

We have shown that the FC network possesses generalization characteristics that compare favorably with other neural networks such as the BP and RBF networks. We show that the design of an FC network for any real-world problem is very easy compared to BP and RBF networks. Therefore, given its fast training speed, it offers an attractive alternative to the other networks.

This paper has described applications of the FC network to the tasks of pattern recognition and signal prediction. Other applications such as data compression, equalization in business, nonlinear filtering, and estimation are being researched.

References

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation.* Prentice Hall, Upper Saddle River, 1999.

[2] M.A. Henon, "Two dimensional map with a strange attractor," *Communications in Mathematical Physics*, vol. 50, pp. 69-77, 1976.

[3] S. Kak, On training feedforward neural networks. *Pramana -J. of Physics*, 40, pp. 35-42, 1993.

[4] S. Kak, New algorithms for training feedforward neural networks. *Pattern Recognition Letters*, 15, pp. 295-298, 1994.

[5] S. Kak and J. Pastor, Neural networks and methods for training neural networks. *U.S. Patent No. 5,426,721*, June 20, 1995.

[6] S. Kak, The three languages of the brain: quantum, reorganizational, and associative. In *Learning as Self-Organization*, K. Pribram and J. King (eds.). Lawrence Erlbaum, Mahwah, 1996, pp. 185-219.

[7] S. Kak, "On generalization by neural networks," Information Sciences, Vol. 111, pp. 293-302, 1998.

[8] S. Kak, "Better web searches and prediction with instantaneously trained neural networks," IEEE Intelligent Systems, Vol. 14(6), pp. 78-81, 1999.

[9] W. Light (ed.), *Advances in Numerical Analysis Vol II: Wavelets, Subdivision Algorithms, and Radial Basis Functions.* Oxford Science Publications, Oxford, 1991.

[10] C.T. Lin and C.S.G. Lee, *Fuzzy Neural Systems.* Prentice Hall, Upper Saddle River, 1996.

[11] P. Raina, "Comparison of the learning and generalization capabilities of the Kak and the backpropagation algorithms," Information Sciences, Vol. 81, pp. 261-274, 1994.

[12] B. Shu and S. Kak, "A neural network based intelligent metasearch engine," Information Sciences, vol. 120, pp. 1-11, 1999.

[13] G.A. Souza and S. Kak, "Dynamic radius allocation in corner classification neural networks," *Proceedings of the Third International Conference on Computational Intelligence and Neuroscience.* North Carolina, 1998.

[14] K.W. Tang and S. Kak, "A new corner classification approach to neural network training," Circuits, Systems Signal Processing, Vol. 17, pp. 459-469, 1998.

[15] K.W. Tang, *Instantaneous Learning Neural Networks.* Ph.D. Dissertation, LSU, 1999.

[16] J. Zhu and G. Milne, "Implementing Kak neural networks on a reconfigurable computing platform," In FPL 2000, LNCS 1896, R.W. Hartenstein and H. Gruenbacher (eds.), Springer-Verlag, 2000, pp. 260-269.