Fall 1-31-1996

Configuring the radial basis function neural network

Insoo Sohn

New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

CONFIGURING THE RADIAL BASIS FUNCTION NEURAL NETWORK

by
Insoo Sohn

The most important factor in configuring an optimum radial basis function (RBF) network is the training of neural units in the hidden layer. Many algorithms have been proposed, e.g., competitive learning (CL), to train the hidden units. CL suffers from producing "dead-units." The other major factor which was ignored in the past is the appropriate selection of the number of neural units in the hidden layer. The frequency sensitive competitive learning (FSCL) algorithm was proposed to alleviate the problem of dead-units, but it does not alleviate the latter problem. The rival penalized competitive learning (RPCL) algorithm is an improved version of the FSCL algorithm, which does solve the latter problem provided that a larger number of initial neural units are assigned. It is, however, very sensitive to the learning rate. This thesis proposes a new algorithm called the scattering-based clustering (SBC) algorithm, in which the FSCL algorithm is first applied to let the neural units converge. Then scatter matrices of the clustered data are used to compute the *sphericity* for each k, where k is the number of clusters. The optimum number of neural units to be used in the hidden layer is then obtained. The properties of the scatter matrices and sphericity are analytically discussed. A comparative study is done among different learning algorithms on training the RBF network. The result shows that the SBC algorithm outperforms the others.

CONFIGURING THE RADIAL BASIS FUNCTION NEURAL NETWORK

by
Insoo Sohn

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

January 1996

APPROVAL PAGE

CONFIGURING THE RADIAL BASIS FUNCTION NEURAL NETWORK

Insoo Sohn

Dr. Nirwan Ansari, Thesis Advisor Associate Professor of Electrical and Computer Engineering, NJIT	Date
Dr. Yun-Qing Shi , Committee Member Associate Professor of Electrical and Computer Engineering, NJIT	Date
Dr. Edwin Hou, Committee Member Assistant Professor of Electrical and Computer Engineering, NJIT	Date

BIOGRAPHICAL SKETCH

Author: Insoo Sohn

Degree: Master of Science in Electrical Engineering

Date: January 1996

Date of Birth:

Place of Birth:

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1996
- Bachelor of Science in Electrical Engineering,
  Rensselaer Polytechnic Institute, Troy, NY, 1994

Major: Electrical Engineering

This work is dedicated to
my parents

ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his advisor, Dr. Nirwan Ansari, for his guidance and friendship throughout this research.

The author also extends his gratitude to Dr. Yun-Qing Shi and Dr. Edwin Hou for serving as members of the thesis committee.

Special thanks to Amit for help in starting the research, Chen with the help in the analysis of the research, and Murali for the moral support.

Also, the author is grateful to Lisa for her great help in editing of this thesis.

Additionally, the author expresses deep appreciation for his parents, brother, and sister for all their constant support.

Finally, the author wishes to acknowledge his colleagues at the Center of Communications and Signal Processing Research at New Jersey Institute of Technology for their friendship and support throughout this research.

TABLE OF CONTENTS

Chapter		Page
1	INTRODUCTION	1
2	THE RADIAL BASIS FUNCTION NETWORK	3
3	COMPETITIVE LEARNING	5
4	FREQUENCY SENSITIVE COMPETITIVE LEARNING	11
5	RIVAL PENALIZED COMPETITIVE LEARNING	18
6	SCATTERING-BASED CLUSTERING ALGORITHM	29
	6.1 Analysis on Scatter Matrices and Sphericity	29
	6.2 The algorithm	33
	6.3 Simulation Results	34
7	SUPERVISED CLASSIFICATION THROUGH RBF NETWORKS	43
8	CONCLUSION	46
REFERENCES		47

LIST OF TABLES

Table Page

6.1 Angles for the sphericity obtained for four clusters of data by using SBC algorithm 36

6.2 Angles for the sphericity obtained for five clusters of data by using SBC algorithm 36

6.3 Angles for the sphericity obtained for six clusters of data by using SBC algorithm 36

7.1 Classification by using RPCL when r=0.005 and $k = 5$ 45

7.2 Classification by using RPCL when r=0.002 and $k = 5$ 45

7.3 Classification by using RPCL when r=0.002 and $k = 3$ 45

7.4 Classification by using SBC 45

LIST OF FIGURES

Figure		Page
2.1	Radial basis function network.	4
3.1	Two clusters of data that were used in this simulation centered at (-1.0,0.0) and (1.0,0.0) each with 100 patterns Gaussian distributed with deviation of 0.2.	7
3.2	The learning trace obtained by using CL with initial centers at (2.0,2.0) and (2.5,2.5).	7
3.3	Trace of the x-coordinate of the cluster centers for the first 1000 iterations.	8
3.4	Trace of the y-coordinate of the cluster centers for the first 1000 iterations.	8
3.5	Three clusters of data that were used in this simulation centered at (-1.0,1.0), (1.0,1.0), and (0.0,0.0) each with 100 patterns Gaussian distributed with deviation of 0.2.	9
3.6	The learning trace obtained by using CL with initial centers at (2.0,2.0), (2.5,2.5), and (2.0,2.5).	9
3.7	Trace of the x-coordinate of the cluster centers for the first 1000 iterations.	10
3.8	Trace of the y-coordinate of the cluster centers for the first 1000 iterations.	10
4.1	Five clusters of data that were used in this simulation centered at (-1.0,-1.0), (1.0,-1.0), (-1.0,1.0), (1.0,1.0), and (0.0,0.0) each with 100 patterns Gaussian distributed with deviation of 0.2.	13
4.2	The learning trace obtained by using FSCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), and (3.1,3.1).	13
4.3	Trace of the x-coordinate of the cluster centers for the first 1000 iterations.	14
4.4	Trace of the y-coordinate of the cluster centers for the first 1000 iterations.	14
4.5	The learning trace obtained by using FSCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), and (3.3,3.1).	15
4.6	Trace of the x-coordinate of the cluster centers for the first 1000 iterations.	15
4.7	Trace of the y-coordinate of the cluster centers for the first 1000 iterations.	16
4.8	The learning trace obtained by using FSCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), (3.3,3.1), and (3.5,3.1).	16
4.9	Trace of the x-coordinate of the cluster centers for the first 1000 iterations.	17

Figure Page

4.10 Trace of the y-coordinate of the cluster centers for the first 1000 iterations. 17

5.1 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), and (3.1,3.1). 21

5.2 Trace of the x-coordinate of the cluster centers for the first 1000 iterations. 21

5.3 Trace of the y-coordinate of the cluster centers for the first 1000 iterations. 22

5.4 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), and (3.3,3.1). 22

5.5 Trace of the x-coordinate of the cluster centers for the first 1000 iterations. 23

5.6 Trace of the y-coordinate of the cluster centers for the first 1000 iterations. 23

5.7 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), (3.3,3.1), and (3.5,3.1) and r=0.0001. 24

5.8 Trace of the x-coordinate of the cluster centers for the first 1000 iterations. 24

5.9 Trace of the y-coordinate of the cluster centers for the first 1000 iterations. 25

5.10 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), (3.3,3.1), and (3.5,3.1) and r=0.005. 25

5.11 Trace of the x-coordinate of the cluster centers for the first 1000 iterations. 26

5.12 Trace of the y-coordinate of the cluster centers for the first 1000 iterations. 26

5.13 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), (3.3,3.1), and (3.5,3.1) and r=0.001. 27

5.14 Trace of the x-coordinate of the cluster centers for the first 1000 iterations. 27

5.15 Trace of the y-coordinate of the cluster centers for the first 1000 iterations. 28

6.1 Four clusters of data that were used in this simulation centered at (-1.0,0.0), (1.0,0.0), (0.0,1.0), and (0.0,-1.0) each with 100 patterns Gaussian distributed with deviation of 0.2. 37

6.2 The learning trace for $k = 4$ with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), and (3.1,3.1). 37

6.3 $Tr(S)$, $Tr(S_W)$, and $Tr(S_B)$ obtained for four clusters by using SBC for $k = 1 \ldots 10$. 38

Figure Page

6.4 Sphericity obtained for four clusters by using SBC for $k = 1 \ldots 10$. 38

6.5 Five clusters of data that were used in this simulation centered at (-1.0,-1.0), (1.0,-1.0), (-1.0,1.0), (1.0,1.0), and (0.0,0.0) each with 100 patterns Gaussian distributed with deviation of 0.2. 39

6.6 The learning trace for $k = 5$ with initial centers at (3.6,3.2), (3.2,3.9), (3.4,3.1), (3.5,3.8), and (3.4,3.7). 39

6.7 $Tr(S)$, $Tr(S_W)$, and $Tr(S_B)$ obtained for five clusters by using SBC for $k = 1 \ldots 10$. 40

6.8 Sphericity obtained for five clusters by using SBC for $k = 1 \ldots 10$. 40

6.9 Six clusters of data that were used in this simulation centered at (-1.0,-1.0), (1.0,-1.0), (-1.0,1.0), (1.0,1.0), (0.0,1.0), and (0.0,-1.0) each with 100 patterns Gaussian distributed with deviation of 0.2. . . . 41

6.10 The learning trace for $k = 6$ with initial centers at (3.2,3.8), (3.3,3.9), (3.5,3.9), (3.5,3.1), (3.9,3.7), and (3.9,3.7). 41

6.11 $Tr(S)$, $Tr(S_W)$, and $Tr(S_B)$ obtained for six clusters by using SBC for $k = 1 \ldots 10$. 42

6.12 Sphericity obtained for six clusters by using SBC for $k = 1 \ldots 10$. 42

CHAPTER 1

INTRODUCTION

Radial basis function (RBF) networks have drawn attention as an alternative to the multilayered perceptron, such as the back-propagation network for functional approximation. An RBF network consists of three basic layers: the input layer, the hidden layer, and the output layer. The input layer is made up of source nodes. The hidden layer has a high enough dimension, which serves a different purpose from that in a multilayer perceptron. The output layer supplies the response of the network to the activation patterns applied to the input layer. The transformation from the input space to the hidden-unit space is nonlinear, whereas the transformation from the hidden-unit space to the output space is linear. RBF networks have been applied in different areas such as channel equalization in digital communications [14] and signal detection [4].

The most important consideration in configuring an RBF network is the determination of the number and centers of the hidden units. An obvious trivial choice is to have each of the data correspond to a center, but this is not practical for a large amount of data. Much research has been done on the training of RBFs. Broomhead and Lowe [5] were among the first, using the k - means algorithm to minimize the number of centers. Other learning methods include the genetic algorithm [2], [21], the orthogonal least squares algorithm [15], and the competitive learning (CL) [3], [17], [16], [6], [7], [8], [19], [20] which is an adaptive version of k - means algorithm, have been proposed.

CL suffers from producing "dead-units," and the frequency sensitive competitive learning (FSCL) [13] was proposed to alleviate this problem. The key idea in FSCL is to keep count of the frequency of each neural unit winning and use this information to give all the neural units the same opportunity to be modified. With this

improvement on CL, FSCL does alleviate the problem, but FSCL is inadequate when the number of initial neural units are either larger or smaller than the number of clusters.

An improvement over the FSCL method is rival penalized competitive learning (RPCL) [9]. All the procedures are the same, except that another key idea is added. The key idea is that in addition to the center unit, which is the winner modified to adapt to the input, the center unit of its rival (the second winner) is also modified. Utilizing this additional component, RPCL is able to address the problem of having initial neural units that are larger than the number of clusters. Therefore, when the number of neural units are smaller than the number of data clusters, RPCL fails because the neural units do not stablize at the center of the clusters, and even if they do, the representation of the data is lost since the center units would draw data samples from different clusters. To determine the optimum number of neural units, a new algorithm, scattering-based clustering (SBC), is proposed in this thesis. FSCL is adopted to address the problem of under-utilized units and the characteristics of scatter matrices are derived to adaptively determine the optimal number of neural units. A brief review of RBF networks is covered in chapter 2; an overview and simulation results on CL is covered in chapter 3; an overview and simulation results on FSCL is covered in chapter 4; an overview and simulation results on RPCL is covered in chapter 5; an analysis and a description of SBC with simulation results are reported in chapter 6; the application of SBC and RPCL to RBF networks for supervised classification is shown in chapter 7, and the conclusion appears in chapter 8.

CHAPTER 2

THE RADIAL BASIS FUNCTION NETWORK

Fig. 2.1 shows a general RBF network with a mapping $F : R^n \rightarrow R$ according to

$$F(\mathbf{x}) = w_0 + \sum_{i=1}^{K} w_i \varphi(\|\mathbf{x} - \mathbf{c}_i\|), \tag{2.1}$$

where K is the total number of RBFs, w_i are the weights of the output layer, $\varphi(\cdot)$ is the basis function, and c_i are the centers of RBFs.

The weights of the output layer can be easily obtained by using either the pseudo-inverse method or the least mean square (LMS) algorithm if the training set of input $\mathbf{x}$ and the corresponding desired output $\mathbf{d}$ are provided. Different basis functions $\varphi(\cdot)$ can be adopted [18]. The most frequently used basis is the Gaussian function

$$\varphi(x) = \exp(-x^2/2\sigma^2). \tag{2.2}$$

Other basis include the thin-plate-spline function:

$$\varphi(x) = x^2 \log(x), \tag{2.3}$$

the multiquadric function:

$$\varphi(x) = (x^2 + \sigma^2)^{1/2}, \tag{2.4}$$

and the inverse multiquadric function:

$$\varphi(x) = (x^2 + \sigma^2)^{-1/2}. \tag{2.5}$$

Theoretical investigations and practical results, however, seem to show that the type of $\varphi(\cdot)$ is not crucial for the performance of RBF networks [10].

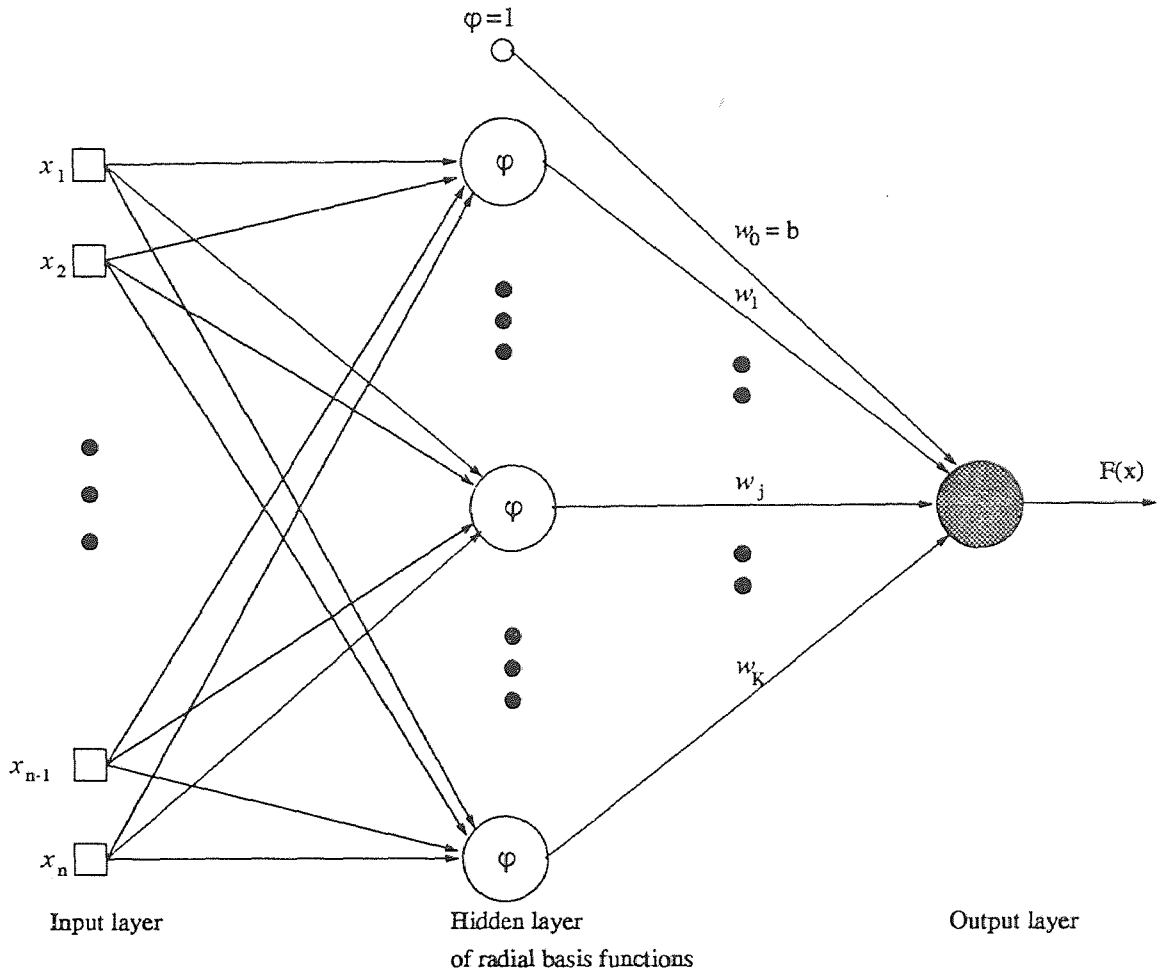

Figure 2.1 Radial basis function network.

CHAPTER 3

COMPETITIVE LEARNING

The essential idea of the CL is to let all the center vectors of the hidden neural units compete with each other. When the input vector $\mathbf{x}$ is presented to all the center vectors $\mathbf{c}$, the difference between $\mathbf{x}$ and $\mathbf{c}$ is computed. The center vector $\mathbf{c}$ that has the minimum variance is considered as the winner and is shifted toward the input vector $\mathbf{x}$ by the amount of the difference. Therefore, the center vector, which is the frequent winner, becomes more sensitive in each iteration of CL. The CL algorithm can be summarized as follows:

Step 1: Randomly choose a sample input vector $\mathbf{x}$ among input data points, and for i=1,...,k, where k is the number of clusters. Determine the winner:

$$h_i = \begin{cases} 1 & \text{if } \|\mathbf{x} - \mathbf{c}_i(n)\|^2 \leq \|\mathbf{x} - \mathbf{c}_j(n)\|^2 \qquad \forall j \neq i \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Step 2: The center vector $\mathbf{c}_i(n)$ of the winner is updated according to:

$$\mathbf{c}_i(n+1) = \mathbf{c}_i(n) + \varepsilon(\mathbf{x} - \mathbf{c}_i(n))h_i, \tag{3.2}$$

where $0 \leq \varepsilon \leq 1$ is the learning rate, which can also be dynamically reduced to zero.

The k-means algorithm works similar to the CL algorithm. Instead of choosing an input vector randomly in every iteration, all the input vectors are compared to each center vector. Input vectors are assigned to center vectors that have the minimum variance, thus essentially clustering data points like step (1) of CL. The mean of data points in each of the k clusters becomes the new center vector. Thus CL can be considered the adaptive version of the k-means algorithm.

The desired result of using the CL algorithm on the input data points is to have the neural units converge toward the center of each data cluster. The actual result, however, highly depends on the initial value of center vectors. Thus, even for

data with only two clusters, the center vectors frequently do not converge toward the center of two data clusters. Fig. 3.2, Fig. 3.3, and Fig. 3.4 show simulation results applying the CL algorithm on two clusters of data, with ϵ=0.05. It is illustrated in Fig. 3.2 that the center units do not converge even for two clusters. Observe from Fig. 3.2, Fig. 3.3, and Fig. 3.4 that the initial center vector initialized at (2.5, 2.5) is the dead-unit that stays in the same position throughout the learning steps. The other initial center vector initialized at (2.0, 2.0) does move toward the clusters, but oscillates between the two clusters, as shown in Fig. 3.2. Fig. 3.6, Fig. 3.7, and Fig. 3.8 show another simulation of the CL algorithm with three clusters; similar results are observed. The initial center unit initialized at (2.5, 2.5) is again the dead unit. The center unit initialized at (2.0, 2.0) is the disturbing unit oscillating between two clusters once again. The center unit initialized at (2.0, 2.5), however, does converge toward the center (1.0, 1.0). These simulation results suggest that the CL algorithm is not the optimum method to train the hidden neural nets of the RBF network. To alleviate the problem of dead-units, FSCL [13] was proposed.

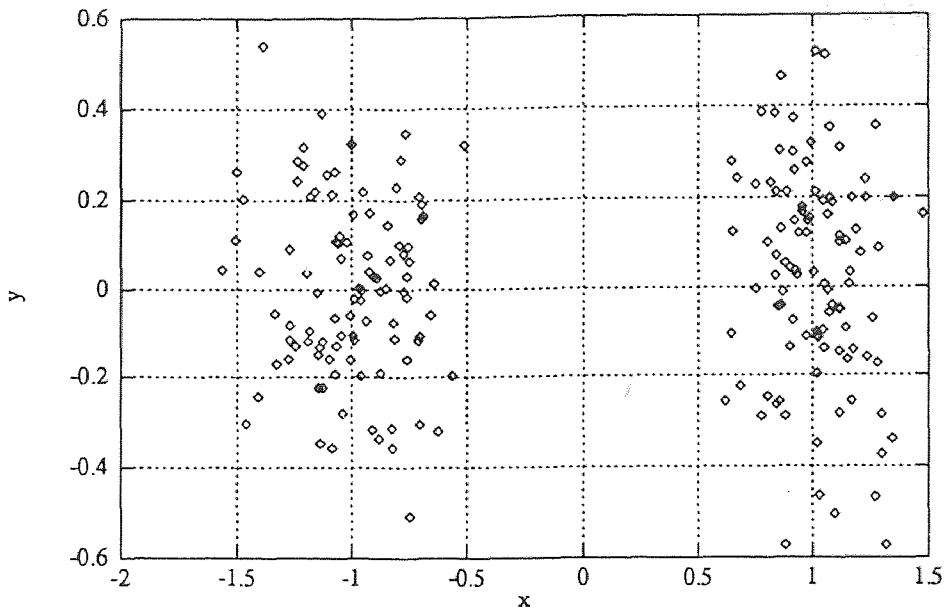

Figure 3.1 Two clusters of data that were used in this simulation centered at (-1.0,0.0) and (1.0,0.0) each with 100 patterns Gaussian distributed with deviation of 0.2.

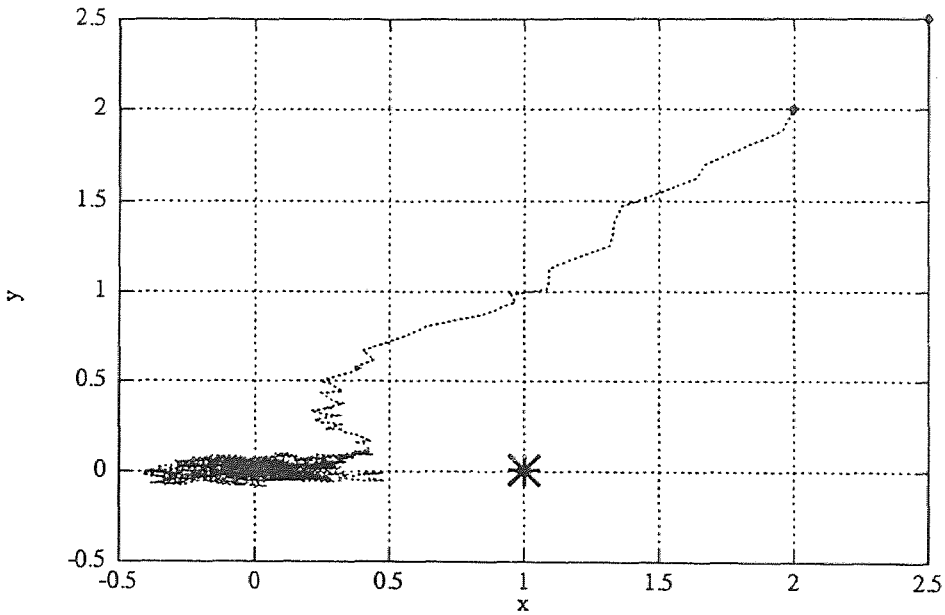

Figure 3.2 The learning trace obtained by using CL with initial centers at (2.0,2.0) and (2.5,2.5).

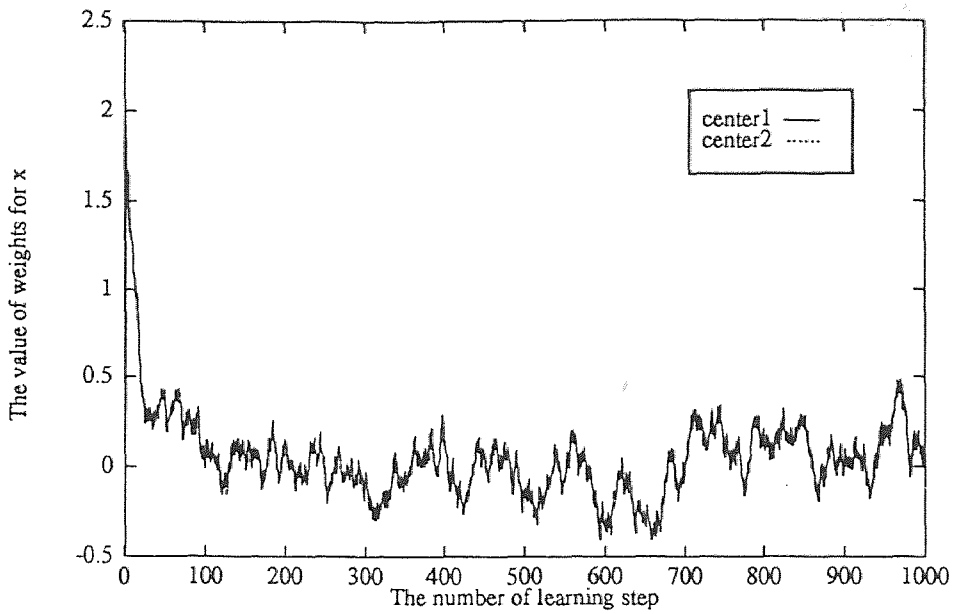

Figure 3.3 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

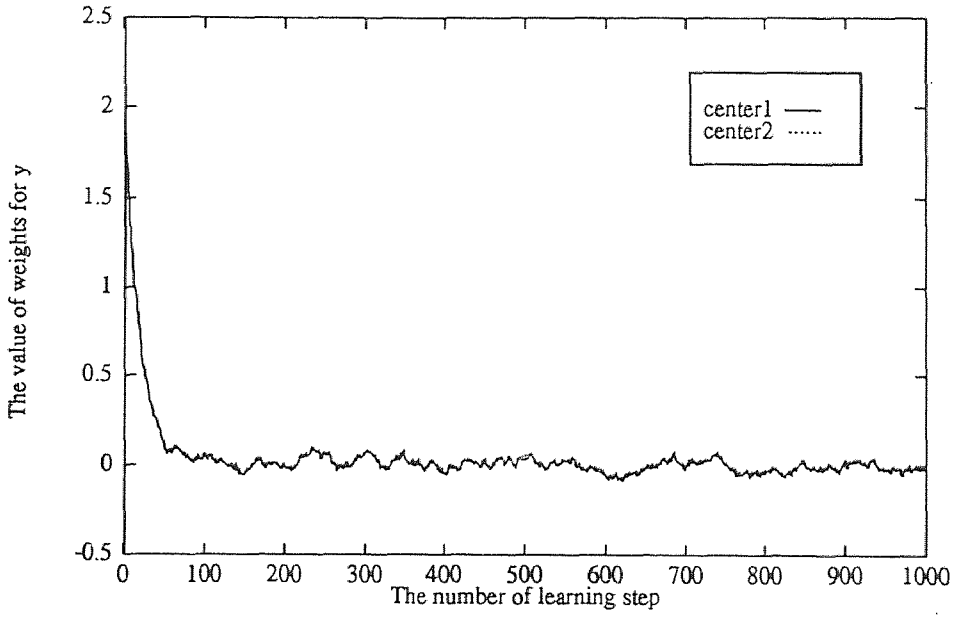

Figure 3.4 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

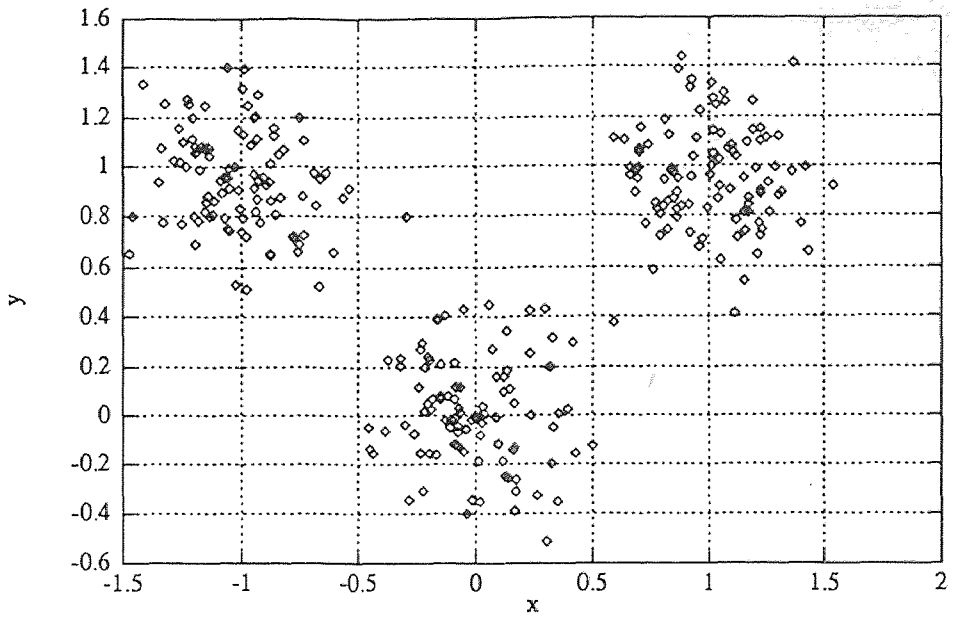

Figure 3.5 Three clusters of data that were used in this simulation centered at (-1.0,1.0), (1.0,1.0), and (0.0,0.0) each with 100 patterns Gaussian distributed with deviation of 0.2.

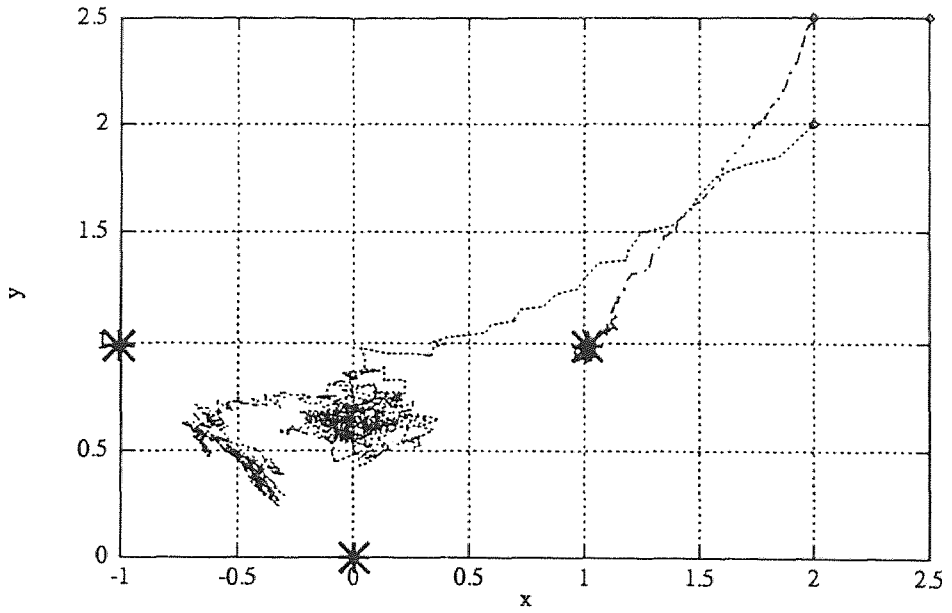

Figure 3.6 The learning trace obtained by using CL with initial centers at (2.0,2.0), (2.5,2.5), and (2.0,2.5).

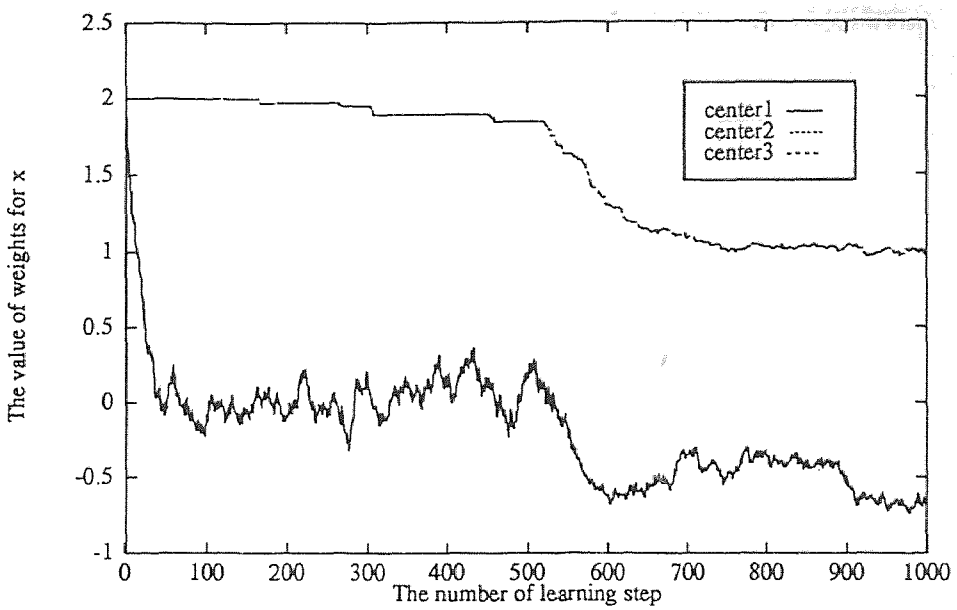

Figure 3.7 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

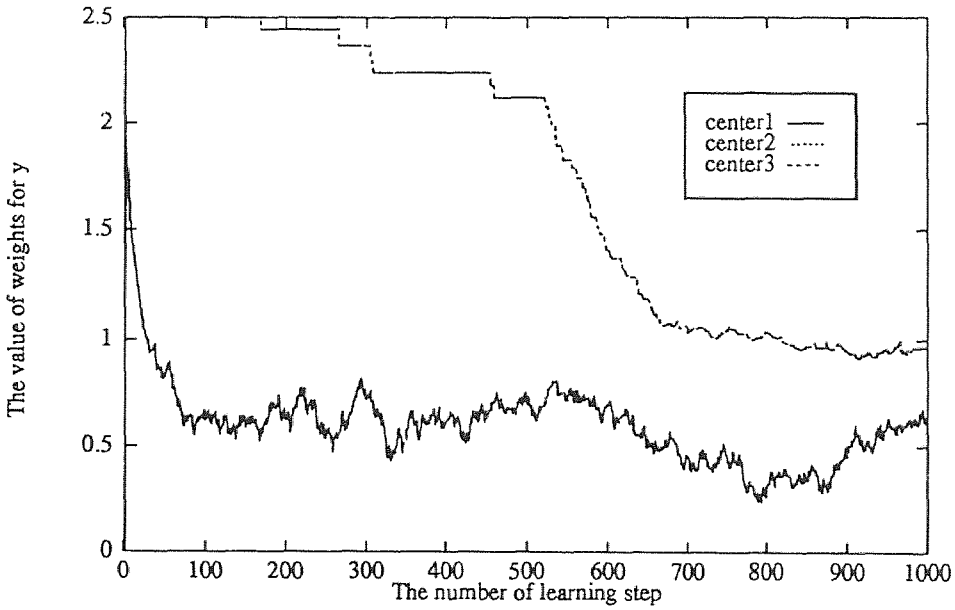

Figure 3.8 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

CHAPTER 4

FREQUENCY SENSITIVE COMPETITIVE LEARNING

Improvement over CL is realized by FSCL through equalization of the average rates of winning for each region. This improvement is achieved by incorporating a count of winnings of each center unit. The FSCL algorithm can be summarized as follows:
Step 1: Randomly choose a sample input vector x among input data points, and for i=1,...,k, where k is the number of clusters. Determine the winner:

$$h_i = \begin{cases} 1 & \text{if } \alpha_i \|\mathbf{x} - \mathbf{c}_i(n)\|^2 \leq \alpha_j \|\mathbf{x} - \mathbf{c}_j(n)\|^2 \qquad \forall j \neq i, \\ 0 & \text{otherwise,} \end{cases} \tag{4.1}$$

where α_i is the total number of times $\mathbf{c}_i(n)$ has been the first winner.
Step 2: The center vector $\mathbf{c}_i(n)$ of the winner is updated according to

$$\mathbf{c}_i(n+1) = \mathbf{c}_i(n) + \varepsilon(\mathbf{x} - \mathbf{c}_i(n))h_i, \tag{4.2}$$

where $0 \leq \varepsilon \leq 1$ is the learning rate which can also be dynamically reduced to zero.

To investigate the performance of FSCL, it is applied to five clusters of data in these simulations. The same number of clusters will be used for the simulation using RPCL, with ε=0.05 and the deviation of the data patterns equal to 0.2 shown in Fig. 4.1. In order to simulate the incapability of obtaining exact k in reality, three cases have been evaluated: the case where the initial number of neural units are the same as the actual number of clusters, the case where the initial number of neural units are larger than the actual number of clusters, and the case where initial number of neural units are smaller than the actual number of clusters. Fig. 4.2, Fig. 4.3, and Fig. 4.4 illustrate case 1, where all five center units converge toward the center of clusters (-1, -1), (-1, 1), (0, 0), (1, -1), and (1, 1). These simulation results confirm that the FSCL algorithm is capable of alleviating the problem of dead-units when the number of initial center units are equal to the actual number of data clusters.

Fig. 4.8, Fig. 4.9, and Fig. 4.10 illustrate case 2 where the number of initial neural units are larger than the actual number of clusters with initial $k = 6$. In order to tackle case 2, it is desirable to have extra center units pushed away from the cluster centers so that they may be eliminated or be used as extra center units. Fig. 4.8, Fig. 4.9, and Fig. 4.10 show the inadequacy of FSCL to function in this manner. The center units initialized at (3.1, 3.5), (3.3, 3.5), (3.5, 3.5), (3.1, 3.1), and (3.5,3.1) do converge to the center of the clusters, but the center unit which started from (3.3,3.1) becomes a disturbing unit converging around (1.3, 1.3). This disturbing unit will not only increase the difficulty of learning for neural units in the output layer, but it will also reduce the recognition rate considerably since the linear output units may be unable to separate the samples distracted by the disturbing unit. Fig. 4.2, Fig. 4.3, and Fig. 4.4 illustrate case 3, that of having a smaller number of initial neural units than the actual number of clusters, where initial $k = 4$. Fig. 4.2, Fig. 4.3, and Fig. 4.4 show that the center units initialized at (3.1, 3.5), (3.3, 3.5), and (3.5, 3.5) do converge to the cluster centers, but the center that was initialized at (3.1, 3.1) becomes the disturbing unit oscillating between the two clusters because there are not enough center units to represent all five data clusters. Apparently, the FSCL algorithm is capable of solving the problem of dead-units, but it is ineffective when the number of initial centers are larger or smaller than the actual number of data clusters. To alleviate these problems, RPCL was introduced with improvement over FSCL [9].

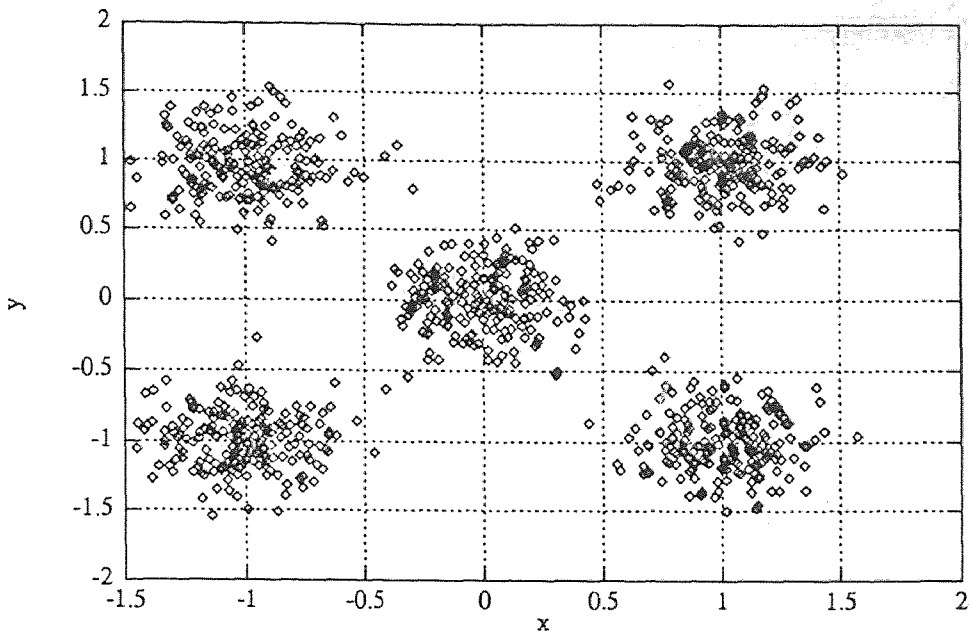

Figure 4.1 Five clusters of data that were used in this simulation centered at (-1.0,-1.0), (1.0,-1.0), (-1.0,1.0), (1.0,1.0), and (0.0,0.0) each with 100 patterns Gaussian distributed with deviation of 0.2.

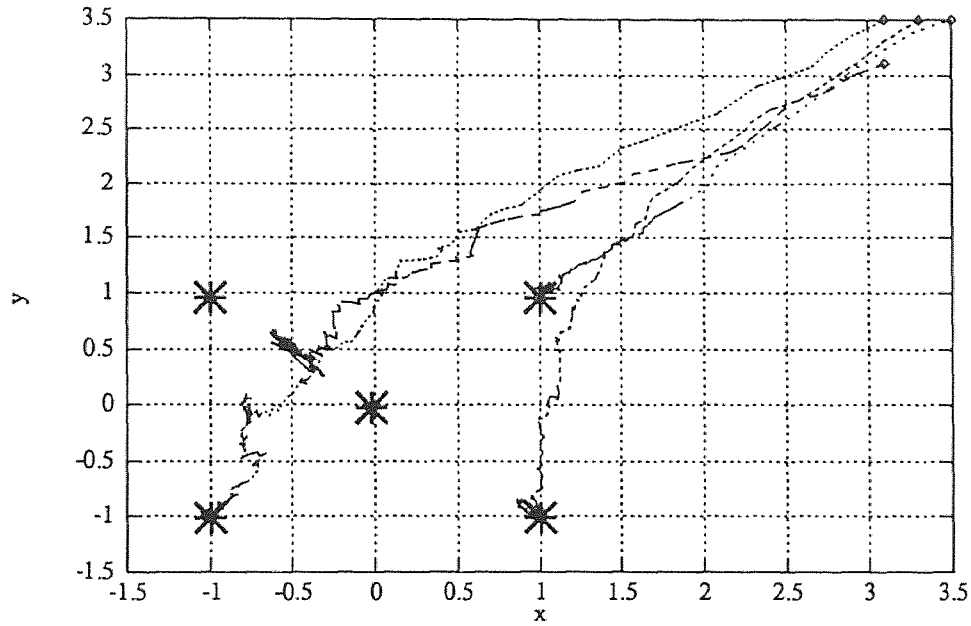

Figure 4.2 The learning trace obtained by using FSCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), and (3.1,3.1).

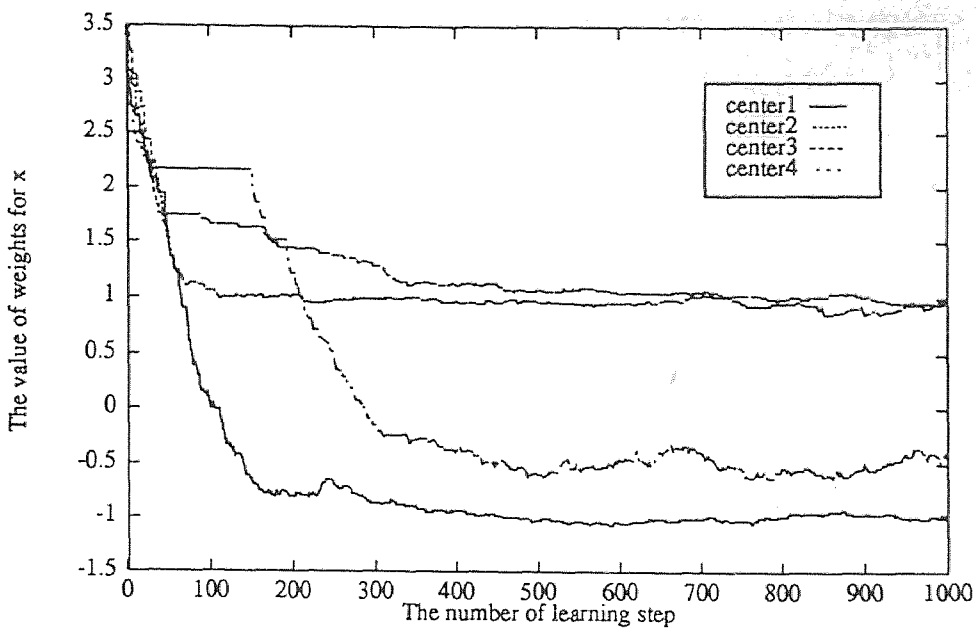

Figure 4.3 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

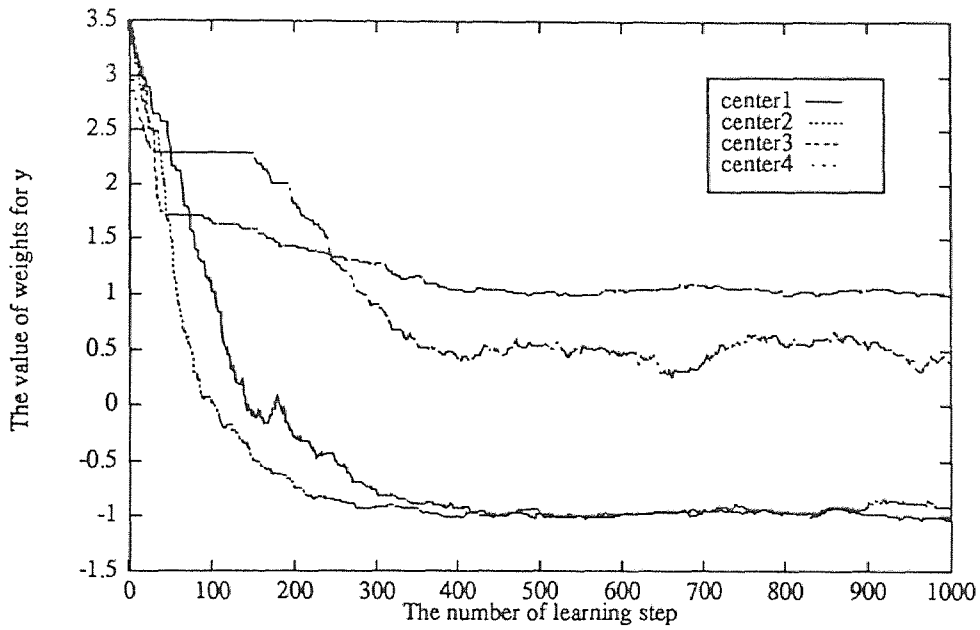

Figure 4.4 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

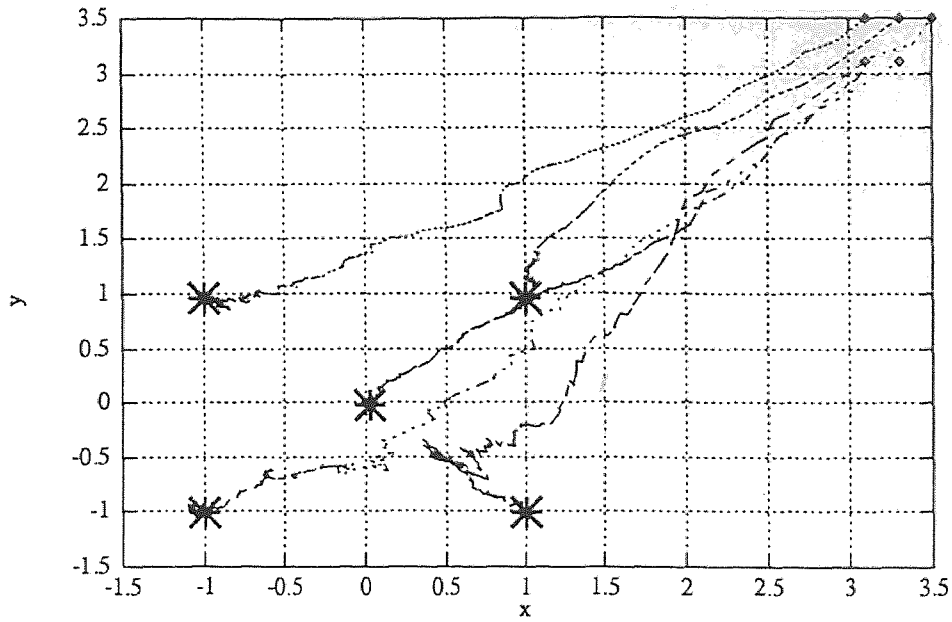

Figure 4.5 The learning trace obtained by using FSCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), and (3.3,3.1).

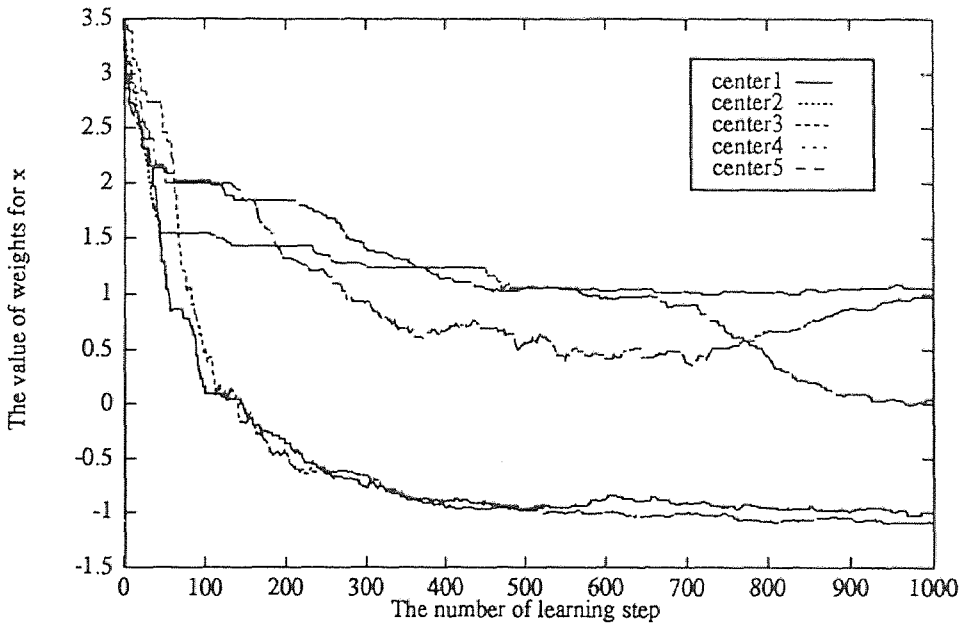

Figure 4.6 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

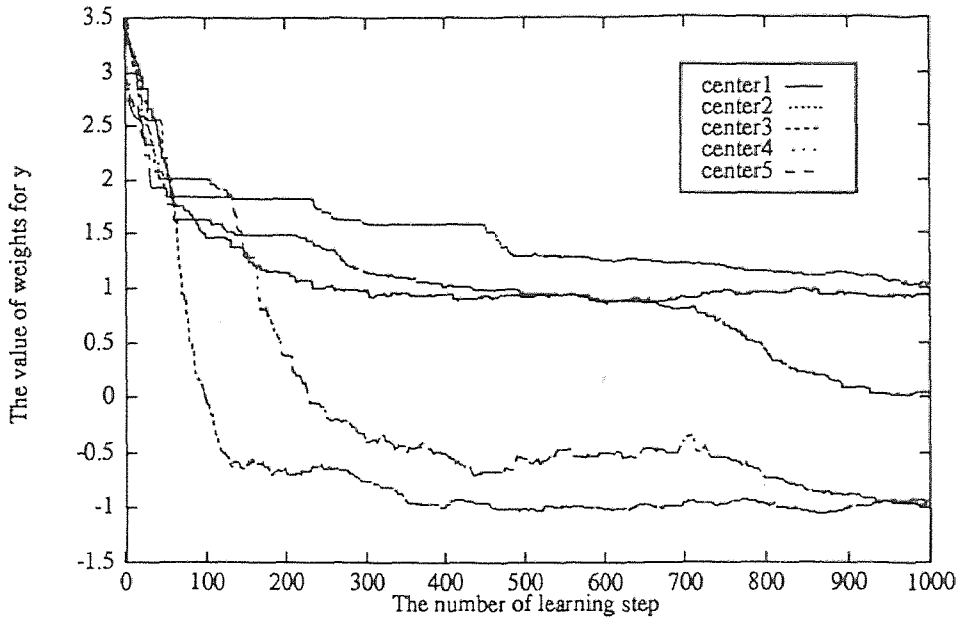

Figure 4.7 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

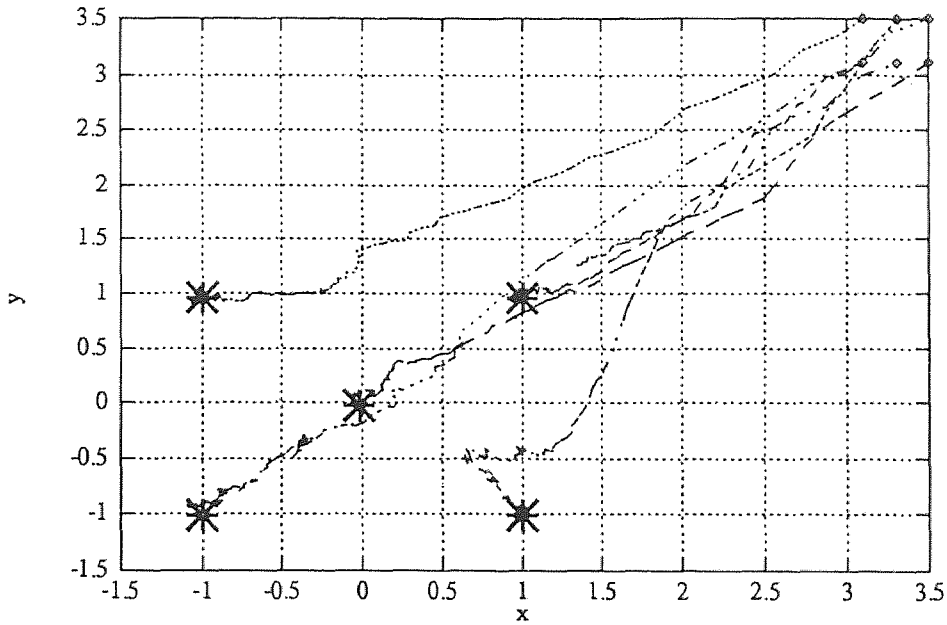

Figure 4.8 The learning trace obtained by using FSCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), (3.3,3.1), and (3.5,3.1).

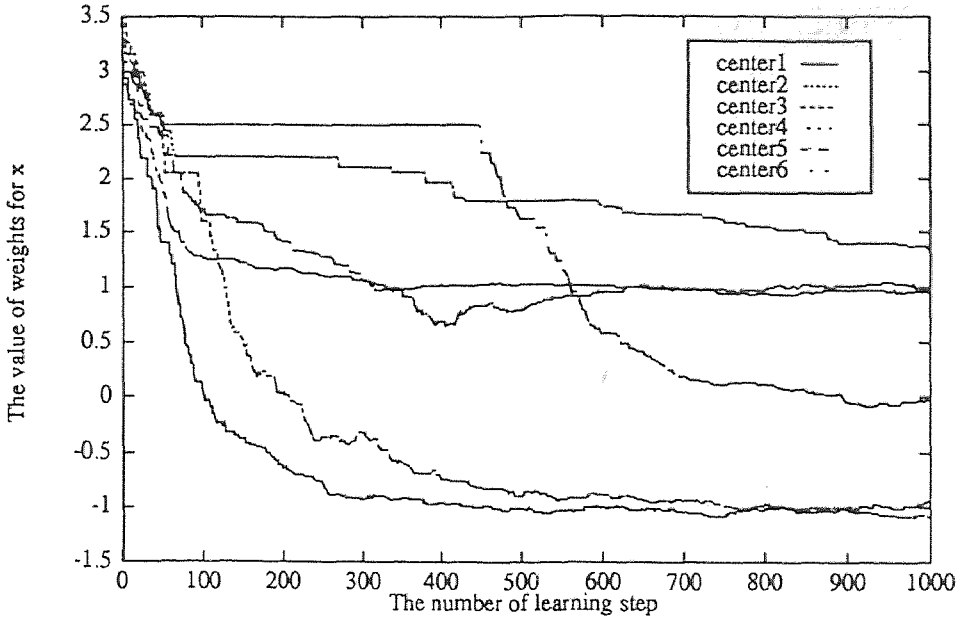

Figure 4.9 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

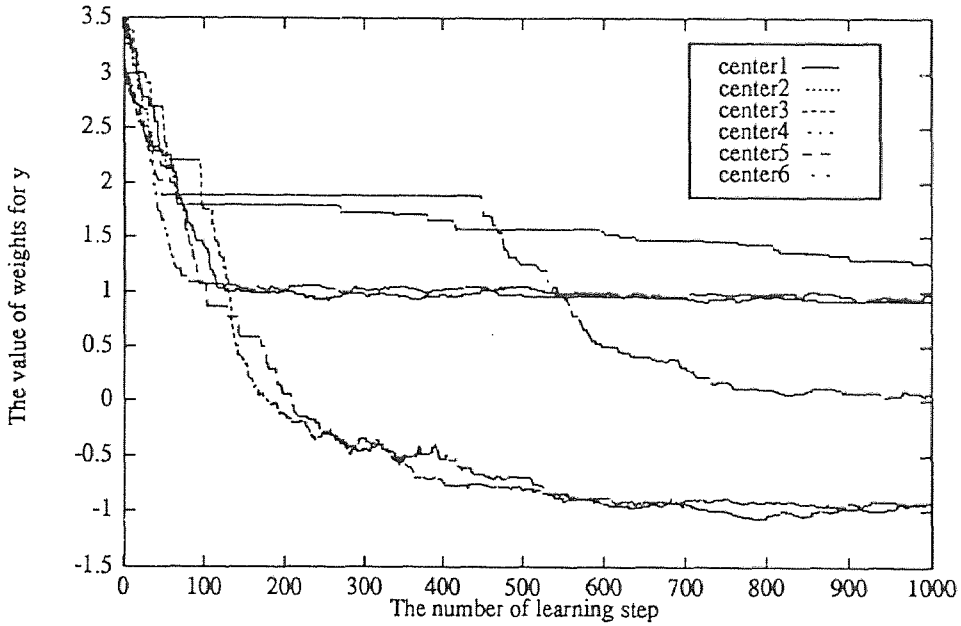

Figure 4.10 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

CHAPTER 5

RIVAL PENALIZED COMPETITIVE LEARNING

The essential idea behind the RPCL algorithm is to equalize the average rate of winning for each region, and it is implemented by letting the second winner of the competition respond to the input vectors in addition to the first winner. The second winner is unlearned by a smaller learning rate, creating a rival penalizing force. The RPCL algorithm can be summarized as follows:

Step 1: Randomly choose a sample input vector $\mathbf{x}$ among input data points, and for i=1,...,k, where k is the number of clusters. Determine the winner:

$$h_i = \begin{cases} 1 & \text{if } \alpha_i \|\mathbf{x} - \mathbf{c}_i(n)\|^2 \leq \alpha_j \|\mathbf{x} - \mathbf{c}_j(n)\|^2 \qquad \forall j \neq i \\ -1 & \text{if } \alpha_l \|\mathbf{x} - \mathbf{c}_l(n)\|^2 \leq \alpha_j \|\mathbf{x} - \mathbf{c}_j(n)\|^2 \qquad \forall j \neq i, l \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

where α_i is the total number of times that the current first winner $\mathbf{c}_i(n)$ has been the first winner, and α_l is the total number of times the current second winner $\mathbf{c}_l(n)$ has been the first winner.

Step 2: The first winner center vector $\mathbf{c}_i(n)$ and the second winner center vector $\mathbf{c}_l(n)$ is updated according to

$$\mathbf{c}_i(n+1) = \mathbf{c}_i(n) + \varepsilon(\mathbf{x} - \mathbf{c}_i(n))h_i \tag{5.2}$$

$$\mathbf{c}_l(n+1) = \mathbf{c}_l(n) + r(\mathbf{x} - \mathbf{c}_l(n))h_i, \tag{5.3}$$

where $0 \leq \varepsilon \leq 1$ is the learning rate, which can also be dynamically reduced to zero.

By unlearning the second winner using equation (12), a rival penalizing force is created, which pushes away the second winner, thus guaranteeing the first winner's convergence. To investigate the performance of RPCL, the same data patterns of five clusters and the same three cases as in FSCL's simulation are applied, with ε=0.05, r=0.0001 and the deviation of the data patterns equal 0.2. Fig. 5.4, Fig. 5.5, and

Fig. 5.6 are illustrations of case 1, where all five center units converge toward the center of data clusters (-1, -1), (-1, 1), (0, 0), (1, -1), and (1, 1). Therefore, it is seen that RPCL is capable of alleviating the problem of dead-units when the number of initial center units are equal to the actual number of clusters. Fig. 5.7 through 5.15 illustrate case 2, that of having a larger number of initial neural units than the actual number of clusters, where initial $k = 6$. There is more than one simulation with six initial center units, because the RPCL algorithm has an extra learning rate r. According to the original paper [9] that introduced RPCL, $r \ll \epsilon$ is suggested. Thus, initially, r is set to 0.0001 and ϵ=0.05. The simulation result shown in Fig. 5.7 shows a similar result compared to Fig. 4.8, with the disturbing unit at (1.3, 1.3). This similar behavior is due to r=0.0001 being too small, thus creating almost no rival penalizing force, which causes the RPCL algorithm to act as if it were an FSCL algorithm. In order to increase the rival penalizing force, r is increased to 0.005 and ϵ is the same. Fig. 5.10, Fig. 5.11, and Fig. 5.12 show two center units that are pushed away by the rival penalizing force. Although extra units being pushed away is desired, the desired number of extra units pushed is one, since the number of clusters is five. This simulation result reveals that r is too big, creating too much of a rival penalizing force, as opposed to too little in the previous case. This conjecture can not be realized in reality, since one does not know the actual number of clusters. By solely observing results obtained in Fig. 5.10, Fig. 5.11, and Fig. 5.12, one would probably think that the optimal number of hidden units is four, since theoretically the rival penalizing force only pushes away the extra center units. Thus, using this invalid result would produce sub-optimum RBF network. After numerous trials and errors, a viable learning rate of r=0.001 is obtained. Fig. 5.13, Fig. 5.14, and Fig. 5.15 show that the extra center unit initialized at (3.1, 3.3) is pushed away from data patterns converging around (2.0, 2.1). All other initial center units converge toward the center of data clusters near (-1.0, -1.0), (1.0, -1.0), (-1.0, 1.0),(1.0, 1.0),

and (0.0, 0.0). Thus RPCL does work with a number of initial centers larger than the actual number of clusters, but it is too sensitive to the value of the learning rate r and results with the wrong optimum number of hidden units for the RBF network, as illustrated in the simulation. Fig. 5.1, Fig. 5.2, and Fig. 5.3 illustrate case 3, that of having a smaller number of initial neural units than the actual number of clusters, where initial $k = 4$. Fig. 5.1 shows a result similar to Fig. 4.2. The only difference is that the center unit initialized at (3.1, 3.1) oscillates between two different clusters of data. Thus RPCL acts similar to FSCL with the number of center units smaller than the actual number of data clusters, because there are not enough center units to represent all five clusters of data.

As illustrated in the simulations, the RPCL algorithm does work with initial center units larger than the actual number of data clusters by pushing away extra center units with a rival penalizing force, but it is very sensitive to the value of r, thus failing to be robust. Moreover, RPCL does not work for a number of center units smaller than the actual number of data clusters. Thus RPCL is superior to FSCL, but it is also not the optimum algorithm for the RBF network training. To solve the problem of obtaining an optimum number of center units for the training of more robust RBF networks, the SBC algorithm is introduced in the next chapter.

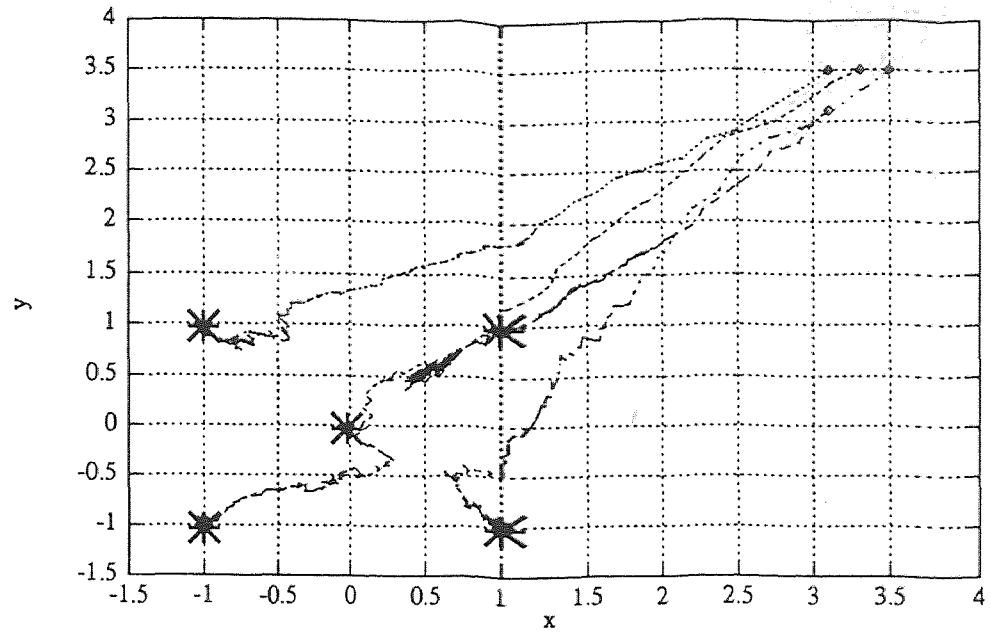

Figure 5.1 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), and (3.1,3.1).

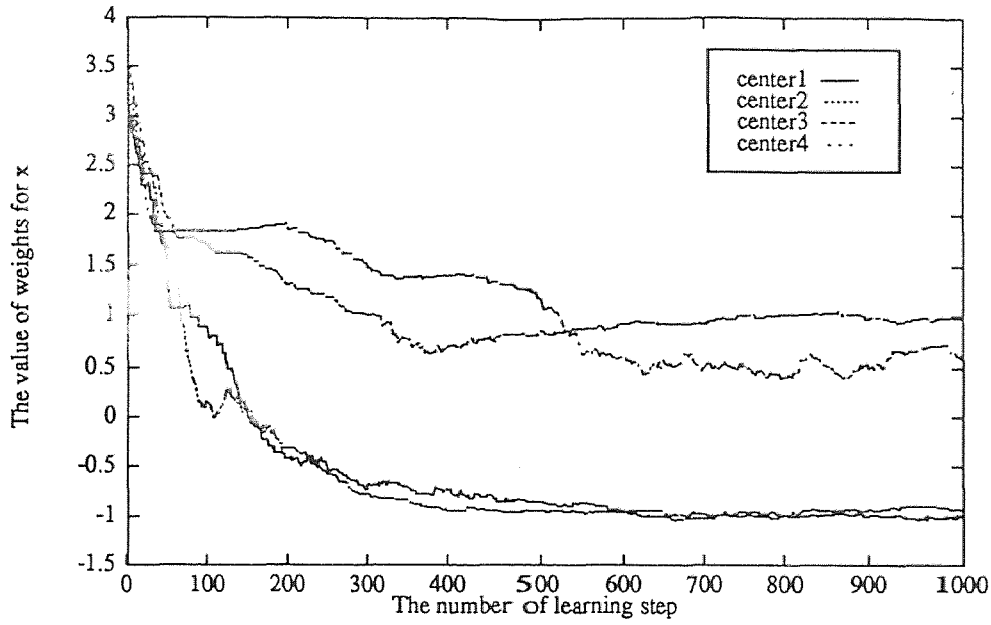

Figure 5.2 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

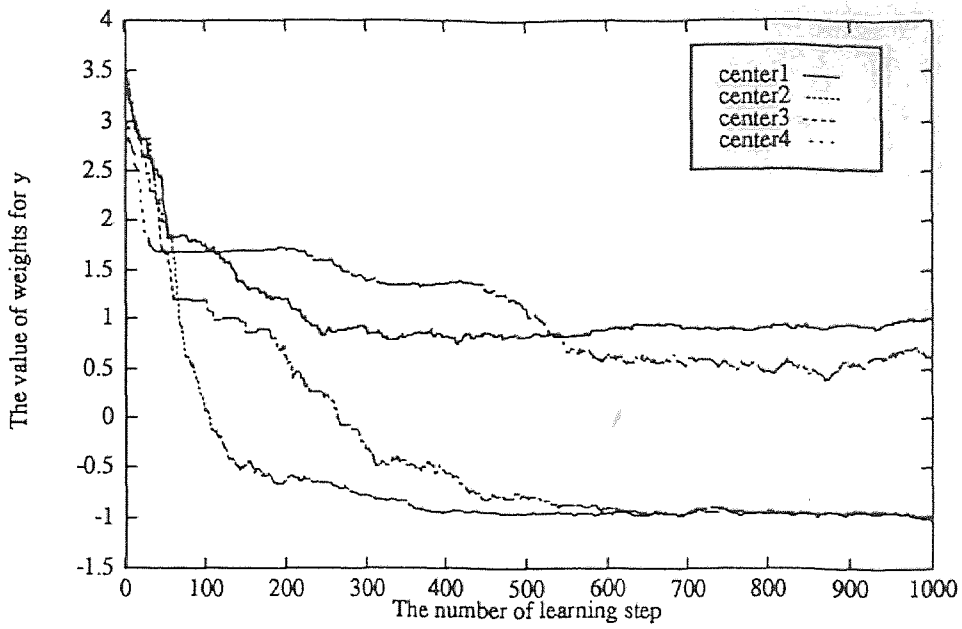

Figure 5.3 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

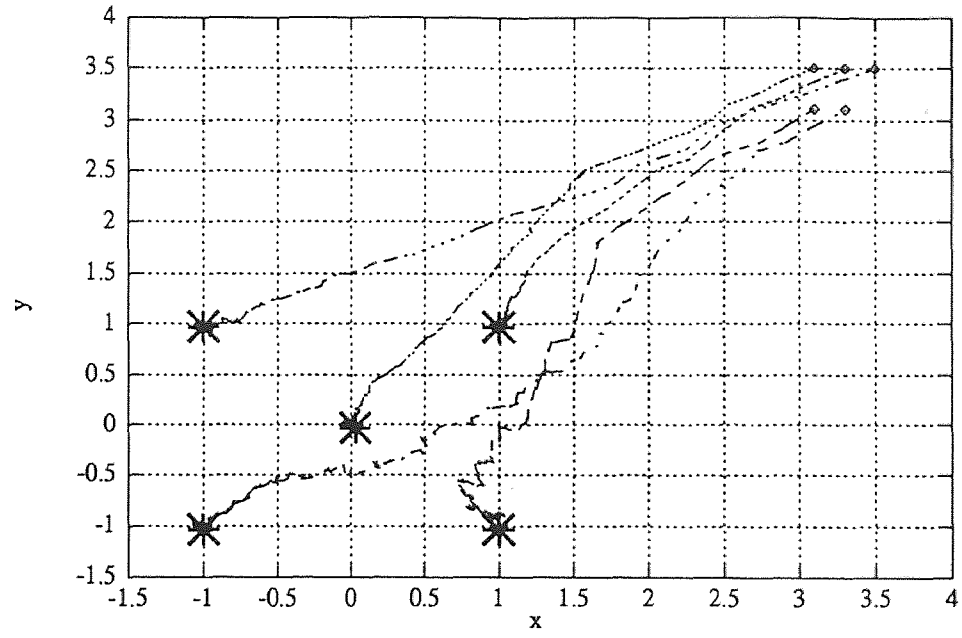

Figure 5.4 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), and (3.3,3.1).

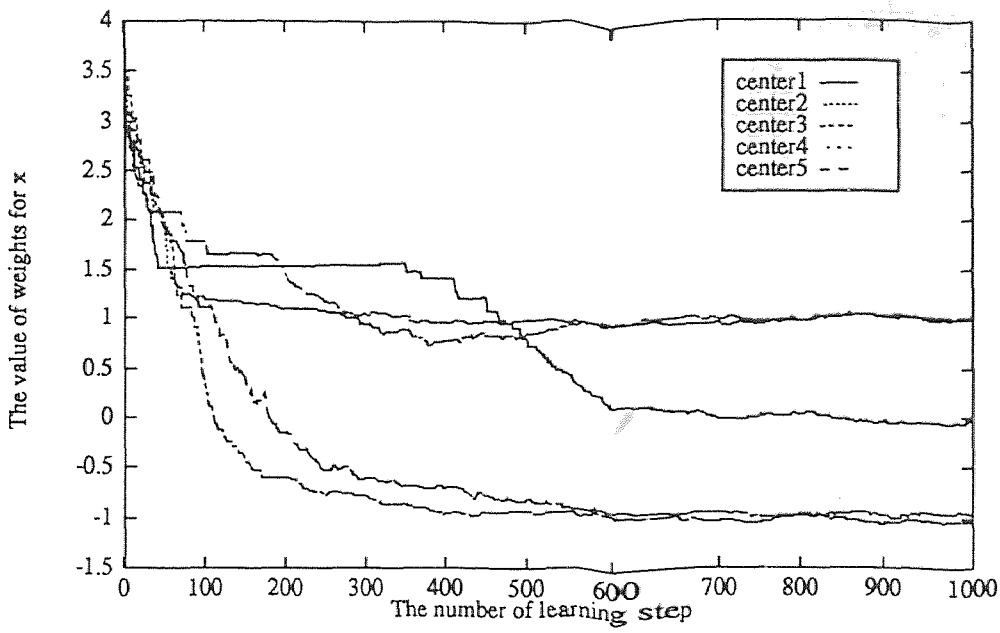

Figure 5.5 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

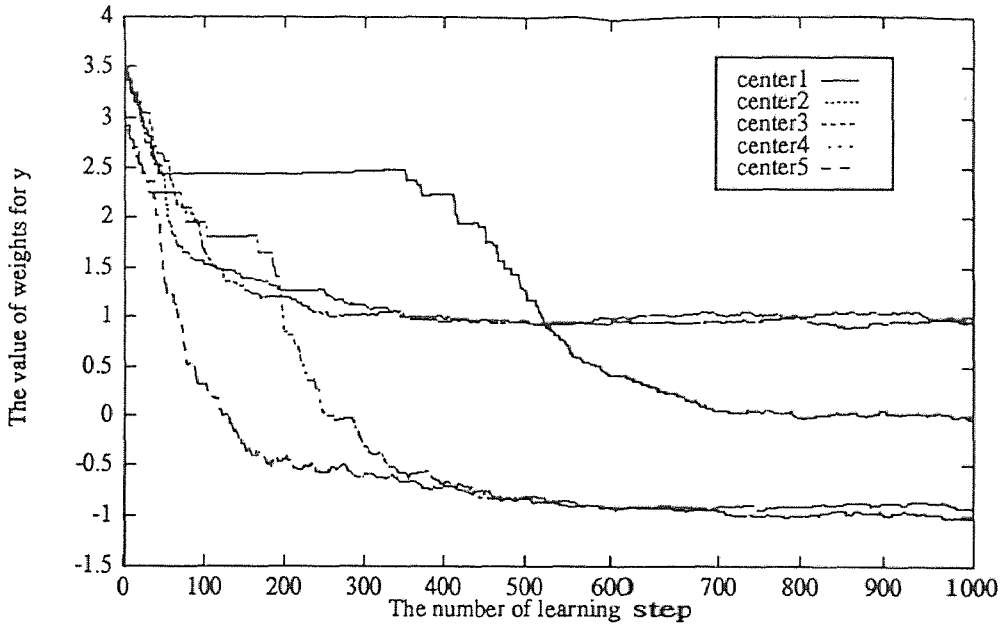

Figure 5.6 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

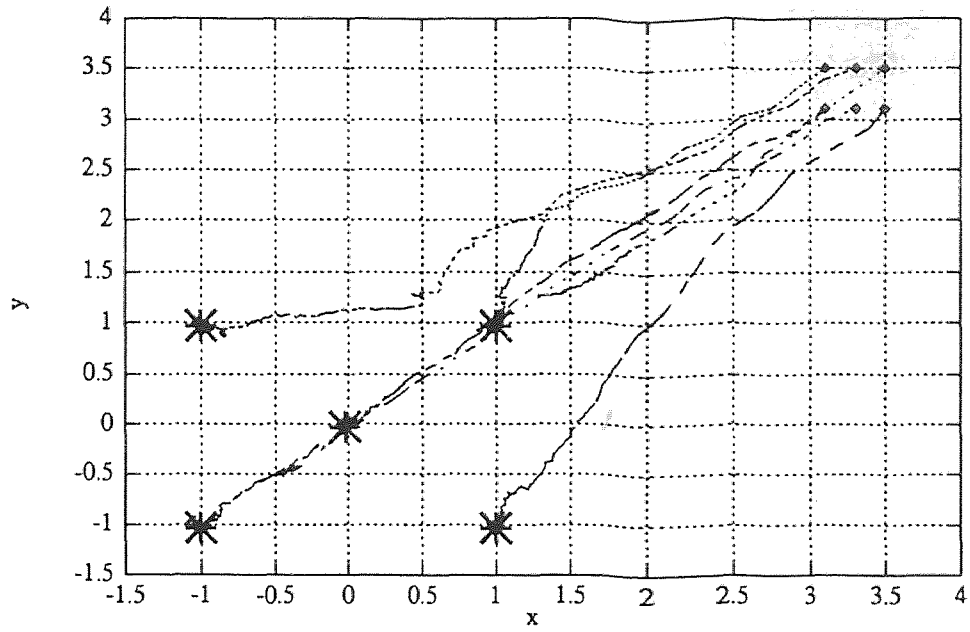

Figure 5.7 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), (3.3,3.1), and (3.5,3.1) and r=0.0001.

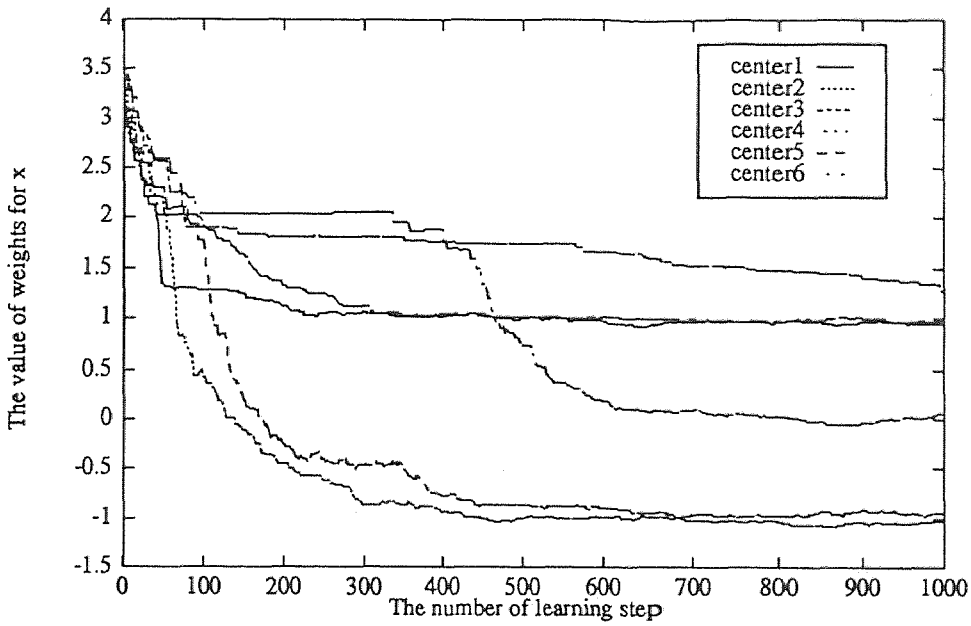

Figure 5.8 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

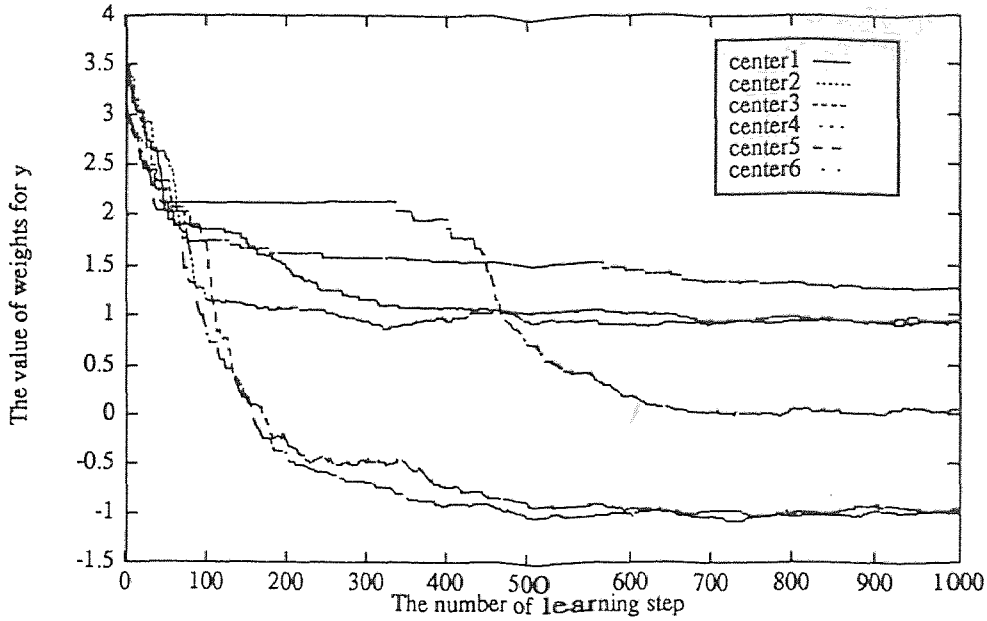

Figure 5.9 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

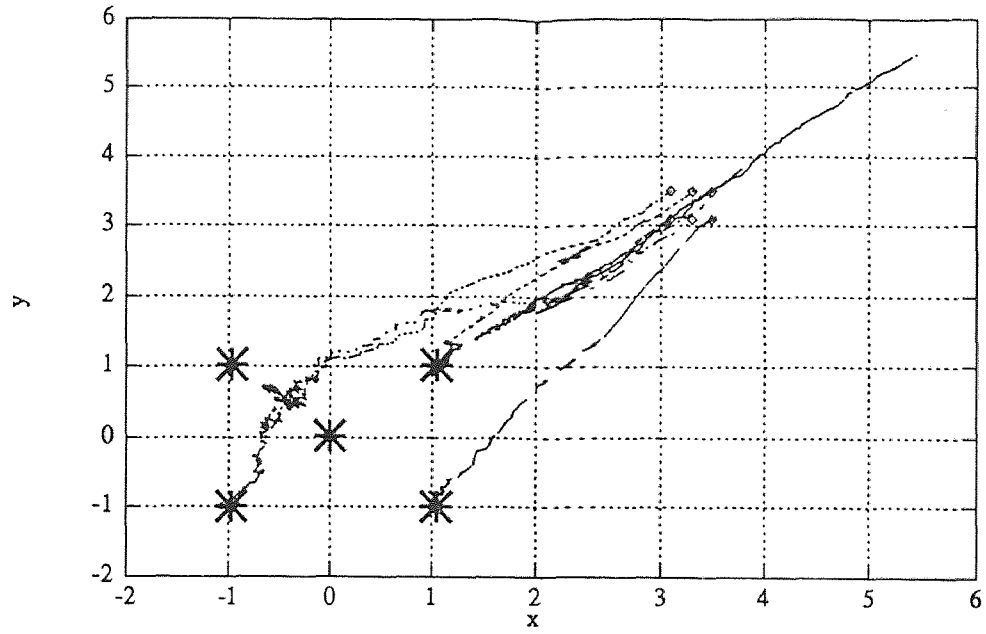

Figure 5.10 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), (3.3,3.1), and (3.5,3.1) and r=0.005.

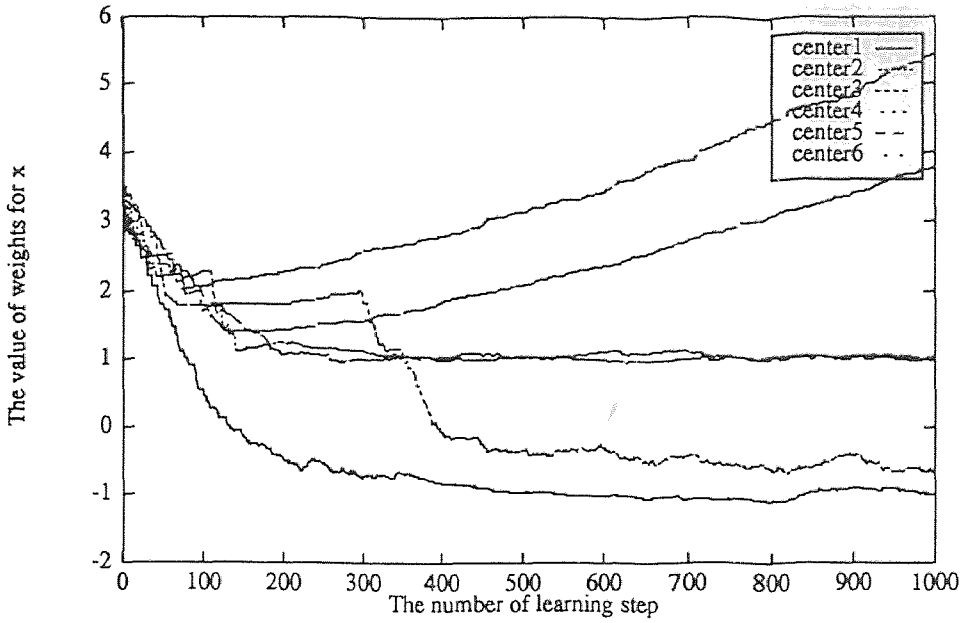

Figure 5.11 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

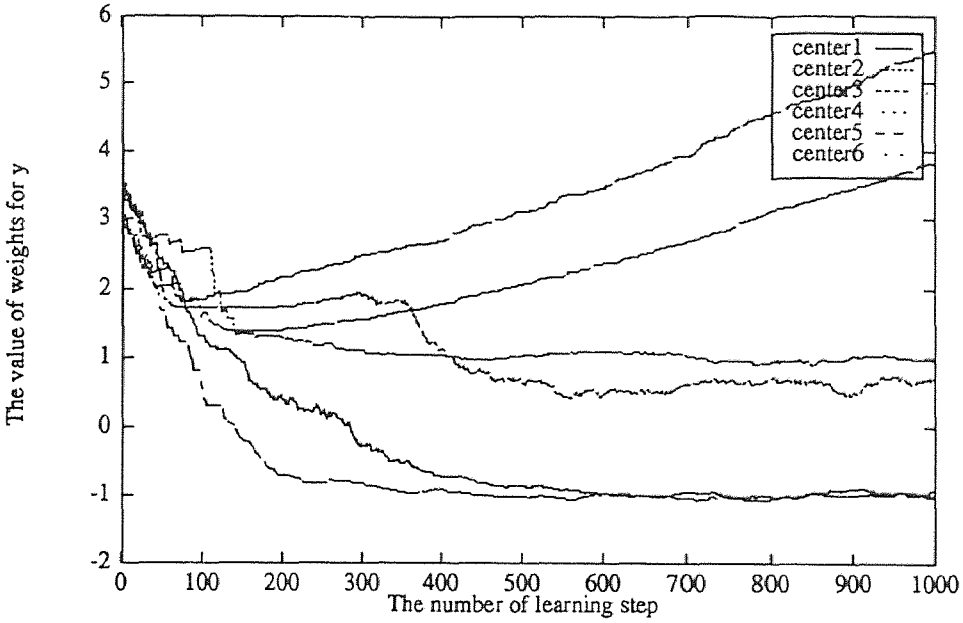

Figure 5.12 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

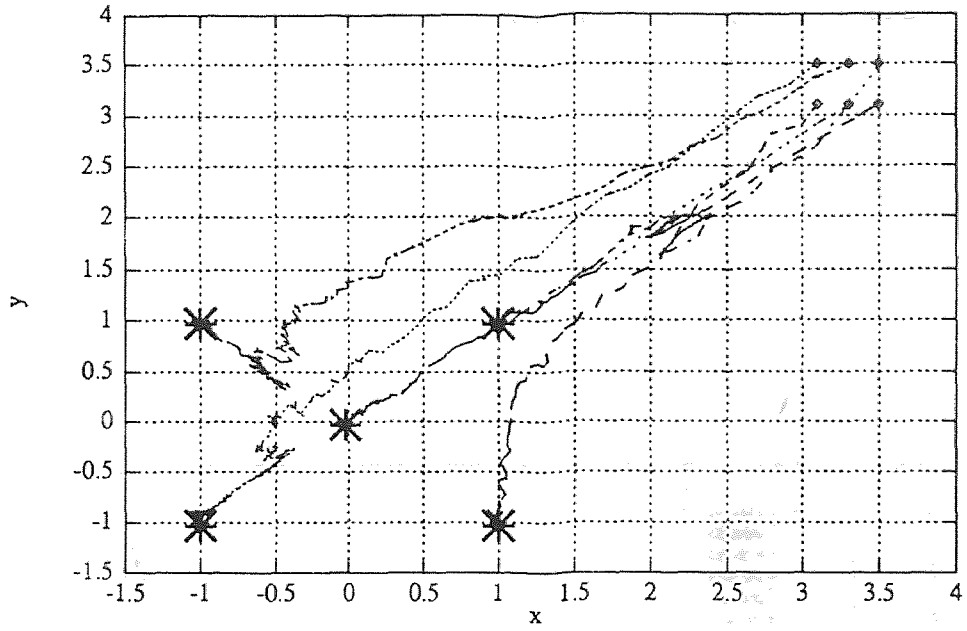

Figure 5.13 The learning trace obtained by using RPCL with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), (3.1,3.1), (3.3,3.1), and (3.5,3.1) and r=0.001.

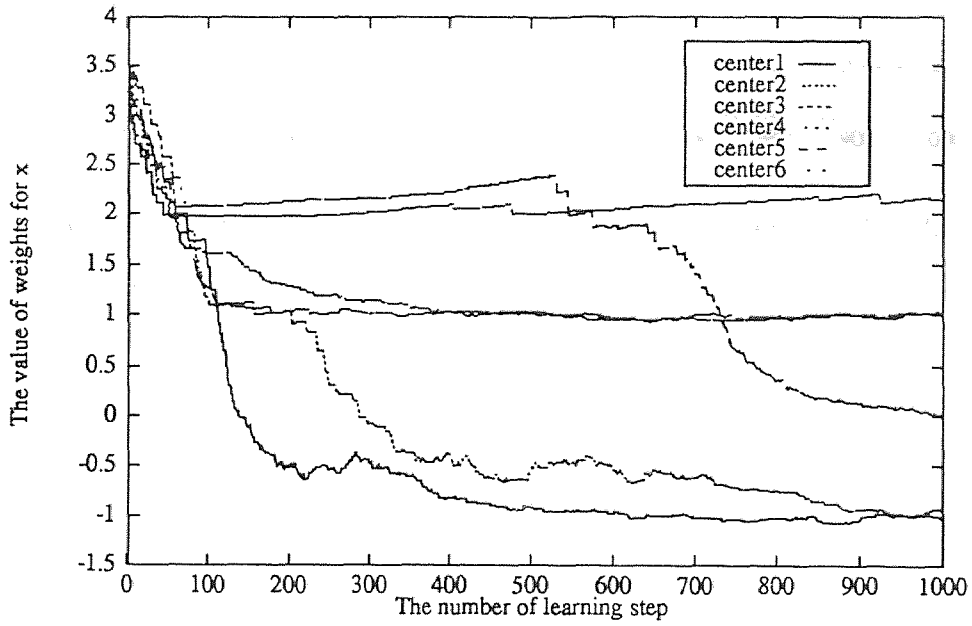

Figure 5.14 Trace of the x-coordinate of the cluster centers for the first 1000 iterations.

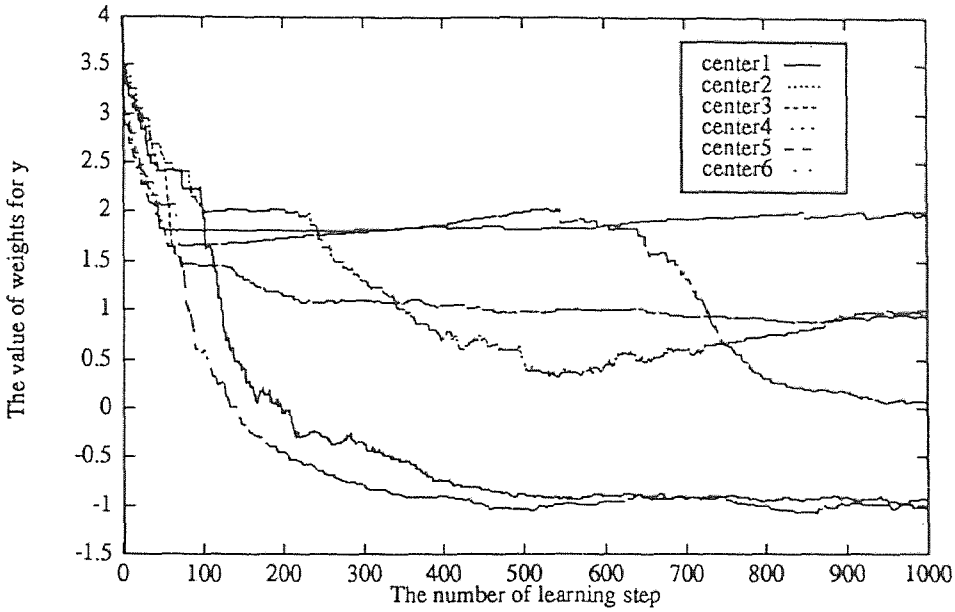

Figure 5.15 Trace of the y-coordinate of the cluster centers for the first 1000 iterations.

CHAPTER 6

SCATTERING-BASED CLUSTERING ALGORITHM

In the following sections, an analysis and a description of the scatter matrices and sphericity will be done, followed by simulation results for a different number of clusters of data patterns.

6.1 Analysis on Scatter Matrices and Sphericity

In the partitional clustering method, different clustering criteria functions are used, such as a *squared error* criterion, a *related minimum variance* criterion, and a *scattering* criterion [1],[12]. The scattering criterion uses scatter matrices used in multiple discriminant analysis. The following equations are the definitions for the scatter matrices:

Definition 1 - jth d-dimensional pattern vector in K th cluster:

$$\mathbf{x}_j^{(k)} = [x_{j1}^{(k)} \dots x_{jd}^{(k)}]^T. \tag{6.1}$$

Definition 2 - d-dimensional mean vector in Kth cluster:

$$\mathbf{m}^{(k)} = [m_1^{(k)} \dots m_d^{(k)}]^T, \tag{6.2}$$

where

$$m_i^{(k)} = (\frac{1}{n_K}) \sum_{j=1}^{n_K} x_{ji}^{(k)}. \tag{6.3}$$

n_K = the number of patterns in the K th cluster.

Definition 3 - Total mean vector:

$$\mathbf{M} = (\frac{1}{n}) \sum_{k=1}^{K} \sum_{j=1}^{n_K} \mathbf{x}_j^{(k)}, \tag{6.4}$$

where

$$n = \sum_{k=1}^{K} n_K. \tag{6.5}$$

Definition 4 - Total scatter matrix:

$$S = \sum_{k=1}^{K} \sum_{j=1}^{n_K} (\mathrm{x}_j^{(k)} - \mathrm{M})(\mathrm{x}_j^{(k)} - \mathrm{M})^T. \tag{6.6}$$

Definition 5 - Total scatter matrix:

$$S = S_W + S_B. \tag{6.7}$$

Definition 6 - Total within scatter matrix:

$$S_W = \sum_{k=1}^{K} \sum_{j=1}^{n_K} (\mathrm{x}_j^{(k)} - \mathrm{m}^{(k)})(\mathrm{x}_j^{(k)} - \mathrm{m}^{(k)})^T. \tag{6.8}$$

Definition 7 - Total between scatter matrix:

$$S_B = \sum_{k=1}^{K} n_K (\mathrm{m}^{(k)} - \mathrm{M})(\mathrm{m}^{(k)} - \mathrm{M})^T. \tag{6.9}$$

Therefore, the trace of the scatter matrices would produce a scalar measure of the scatter matrix. This trace of the scatter matrices measures the square of the scattering radius, since it is proportional to the sum of the variances in the coordinate directions. The following equations are the trace of the scatter matrices:

Definition 8 - Trace of total scatter matrix:

$$Tr(S) = \sum_{k=1}^{K} \sum_{j=1}^{n_K} (\mathrm{x}_j^{(k)} - \mathrm{M})^T (\mathrm{x}_j^{(k)} - \mathrm{M}). \tag{6.10}$$

For the 2-D case:

$$Tr(S) = \sum_{k=1}^{K} \sum_{j=1}^{n_K} (x_j^{(k)} - M_x)^2 + (y_j^{(k)} - M_y)^2. \tag{6.11}$$

Definition 9 - Trace of total scatter matrix:

$$Tr(S) = Tr(S_W) + Tr(S_B). \tag{6.12}$$

Definition 10 - Trace of total within scatter matrix:

$$Tr(S_W) = \sum_{k=1}^{K} \sum_{j=1}^{n_K} (\mathrm{x}_j^{(k)} - \mathrm{m}^{(k)})^T (\mathrm{x}_j^{(k)} - \mathrm{m}^{(k)}) = \sum_{k=1}^{K} e_k^2, \tag{6.13}$$

where e_k^2 is the mean square error at each k.

For the 2-D case:

$$Tr(S_W) = \sum_{k=1}^{K} \sum_{j=1}^{n_K} (x_j^{(k)} - m_x^{(k)})^2 + (y_j^{(k)} - m_y^{(k)})^2. \tag{6.14}$$

Definition 11 - Trace of total between scatter matrix:

$$Tr(S_B) = \sum_{k=1}^{K} n_K (\mathbf{m}^{(k)} - \mathbf{M})^T (\mathbf{m}^{(k)} - \mathbf{M}). \tag{6.15}$$

For the 2-D case:

$$Tr(S_B) = \sum_{k=1}^{K} n_K ((m_x^k - M_x)^2 + (m_y^k - M_y)^2). \tag{6.16}$$

Using these scatter matrix equations, a new criterion called *sphericity* is introduced [11]. The definition of sphericity is as follows:

Definition 12 - Sphericity:

$$Sphericity = \gamma[S_W, S_B] = \frac{Tr(S_W)Tr(S_B)}{Tr(S_W) + Tr(S_B)} = \frac{Tr(S_W)Tr(S_B)}{Tr(S)}. \tag{6.17}$$

The analysis on scatter matrices and sphericity is done using Proposition 1, Proposition 2, and Proposition 3.

Proposition 1. $Tr(S_W)$ monotonically decreases with k.

Proof. Without loss of generality, assume that a given set of n distinct patterns in d dimensions has been partitioned into k clusters. When one uses the k-means algorithm, the following equation is derived:

$$\sum_{j \in L} \| x_j - m_{k+1} \|^2 < \sum_{j \in L} \| x_j - m_i \|^2 \qquad \forall i \neq k+1, \tag{6.18}$$

where L is the set of integers corresponding to the subscripts of patterns assigned to the $k+1$ th cluster. This is justified since new data patterns assigned to the $k+1$ th cluster are closer to m_{k+1} than m_i; however, E_k^2 for other clusters remain the same. Therefore,

$$E^2_{k+1}(1) < \overline{E^2_k}, \tag{6.19}$$

where $\overline{E^2_k}$ is the mean square error at a predefined threshold. Reassignment of data patterns take place from one iteration to another if and only if

$$E^2_k(t+1) < E^2_k(t), \tag{6.20}$$

and ends if and only if

$$E^2_k(t+1) = E^2_k(t). \tag{6.21}$$

Therefore,

$$E^2_{k+1}(SS) \leq E^2_{k+1}(1) < \overline{E^2_k}. \tag{6.22}$$

Thus Proposition 1 is proved.

Proposition 2. $Tr(S_B)$ monotonically increases with k.

Proof. By definition $Tr(S)$ is constant for all k. Additionally, by definition $Tr(S) = Tr(S_W) + Tr(S_B)$. Therefore, $Tr(S_B)$ has to be monotonically increasing to have a constant $Tr(S)$. Thus Proposition 2 is proved.

Proposition 3. $\gamma[S_W, S_B]$ monotonically decreases for $k \geq 2$ when $\frac{Tr(S_B)}{Tr(S_W)} > 1$.

Proof. Using definition (29),

$$\frac{\partial \gamma[S_W, S_B]}{\partial k} = \frac{\partial \gamma[S_W, S_B]}{\partial Tr(S_W)} \frac{\partial Tr(S_W)}{\partial k} = \frac{1}{Tr(S)}[Tr(S) - 2Tr(S_W)] \frac{\partial Tr(S_W)}{\partial k}. \tag{6.23}$$

We know that $\frac{1}{Tr(S)}$ is always positive from the definition, $\frac{\partial Tr(S_W)}{\partial k}$ is negative from Proposition 1, and since

$$\frac{Tr(S_B)}{Tr(S_W)} > 1$$

$$\begin{aligned} \Rightarrow \frac{Tr(S_W)}{Tr(S_W)} + \frac{Tr(S_B)}{Tr(S_W)} &> 2 \\ \Rightarrow \frac{Tr(S_B) + Tr(S_W)}{Tr(S_W)} &> 2 \\ \Rightarrow Tr(S) &> 2Tr(S_W); \end{aligned} \tag{6.24}$$

therefore,

$$\frac{\partial \gamma[S_W, S_B]}{\partial k} < 0. \tag{6.25}$$

Thus Proposition 3 is proved.

The reason $k \geq 2$ is because $Tr(S_B) = 0$ for $k = 1$. The reason for the condition $\frac{Tr(S_B)}{Tr(S_W)} > 1$, is because $\gamma[S_W, S_B]$ has the "knee" at the optimum k for only well clustered data from simulation results. The larger $\frac{Tr(S_B)}{Tr(S_W)}$, the better the patterns are clustered, since $Tr(S_B)$ shows the variance of patterns between the clusters, and $Tr(S_W)$ shows the variance of patterns within the cluster. $\frac{Tr(S_B)}{Tr(S_W)} < 1$ suggests $Tr(S_W) \geq Tr(S_B)$, which signifies that the patterns are very close to being one cluster. Thus $\frac{Tr(S_B)}{Tr(S_W)}$ has to be greater than one to have well clustered patterns of $k \geq 2$.

6.2 The algorithm

Using the characteristics of scatter matrices, the SBC algorithm could be summarized as follows:

Step 1: Compute $\mathbf{c_i(n)}$ using the FSCL algorithm, where i=1...k.

Step 2: Input patterns $\mathbf{x_d}$, where d=1...total number of data patterns are assigned to center vectors:

$$w_i = \begin{cases} 1 & \text{if } \|\mathbf{x_d} - \mathbf{c_i}(n)\|^2 \leq \|\mathbf{x_d} - \mathbf{c_j}(n)\|^2 \qquad \forall j \neq i \\ 0 & \text{otherwise.} \end{cases} \tag{6.26}$$

Step 3: The data patterns are clustered according to:

$$\mathbf{d}_i(n+1) = \mathbf{d}_i(n) + \mathbf{x}_d w_i, \tag{6.27}$$

where $\mathbf{d}_i(n)$ is the i th data cluster.

Step 4: Compute $\angle\gamma[S_W, S_B]$.

Step 5: Check if the $\angle\gamma[S_W, S_B]$ is the minimum.

Step 6: If $\angle\gamma[S_W, S_B]$ is the minimum, stop. If not, increment k and go back to step 1.

These steps indicate that for the SBC algorithm, one does not have to guess the initial number of center units. Each k is tested incrementally and the optimum number of center units is obtained from the minimum angle of sphericity.

6.3 Simulation Results

To investigate the performance of SBC, it is applied to four, five, and six clusters of data patterns, with ε=0.05, and the deviation of the data patterns equal 0.2. Fig. 6.2, Fig. 6.3, and Fig. 6.4 show the application of SBC on four data clusters. Fig. 6.2 shows the convergence of the center units toward the center of data clusters at (-1, 0), (0, -1), (0, 1), and (1, 0) from utilizing FSCL. Fig. 6.3 shows the $Tr(S)$, $Tr(S_W)$, and $Tr(S_B)$ for $k = 1 \ldots 10$. As Proposition 1 and 2 are proved, $Tr(S_B)$ is monotonically increasing, $Tr(S_W)$ is monotonically decreasing for $k \geq 2$, and $Tr(S)/2 > Tr(S_W)$ for $k \geq 2$. Therefore, these clusters are valid clusters where sphericity could be employed to obtain an optimum number of neural units for the hidden layer of RBF networks. Fig. 6.4 shows the sphericity, and as Proposition 3 proposed, $\gamma[S_W, S_B]$ is maximum at k=2. In seeking the optimum number, one could see that $\gamma[S_W, S_B]$ stablizes at $k = 4$, and the simplest method for the computer to evaluate $\gamma[S_W, S_B]$ is by letting it compute the minimum angle of $\gamma[S_W, S_B]$. The angles for $\gamma[S_W, S_B]$ is shown in Table 6.1, illustrating the minimum angle to be at $k = 4$. Fig. 6.6, Fig. 6.7, and Fig. 6.8 illustrate the utilization of SBC on five clusters of data. Fig. 6.8 shows that

$\gamma[S_W, S_B]$ stablizes at $k = 5$, and Table 6.2 illustrates that the angle is minimum at $k = 5$. Fig. 6.10, Fig. 6.11, and Fig. 6.12 illustrate the utilization of SBC on six data clusters. Fig. 6.12 shows that $\gamma[S_W, S_B]$ stablizes at $k = 6$ and Table 6.3 shows that the angle is minimum at $k = 6$. These simulation results show that the SBC algorithm is more robust compared to RPCL, always producing the optimum number whereas RPCL heavily depends on the learning rate r and fails when the number of initial center units are smaller than the actual number of clusters. Thus, using the optimum number produced by the SBC algorithm and the value of center units corresponding to that optimum number, an optimum RBF network would be produced for that data pattern.

# of cluster	angle
2	318.14°
3	206.37°
4	**96.22°**
5	174.55°
6	176.30°
7	171.84°
8	166.26°
9	168.84°

Table 6.1 Angles for the sphericity obtained for four clusters of data by using SBC algorithm

# of cluster	angle
2	276.14°
3	244.11°
4	179.21°
5	**109.12°**
6	178.21°
7	176.62°
8	179.00°
9	177.29°

Table 6.2 Angles for the sphericity obtained for five clusters of data by using SBC algorithm

# of cluster	angle
2	321.86°
3	197.00°
4	159.29°
5	177.52°
6	**129.96°**
7	174.41°
8	179.82°
9	177.82°

Table 6.3 Angles for the sphericity obtained for six clusters of data by using SBC algorithm

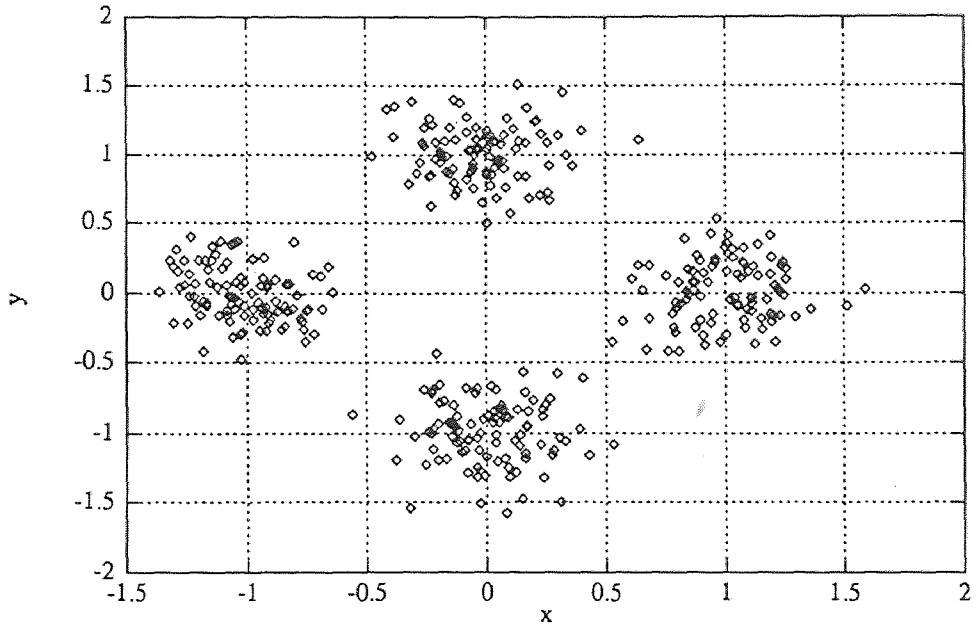

Figure 6.1 Four clusters of data that were used in this simulation centered at (-1.0,0.0), (1.0,0.0), (0.0,1.0), and (0.0,-1.0) each with 100 patterns Gaussian distributed with deviation of 0.2.

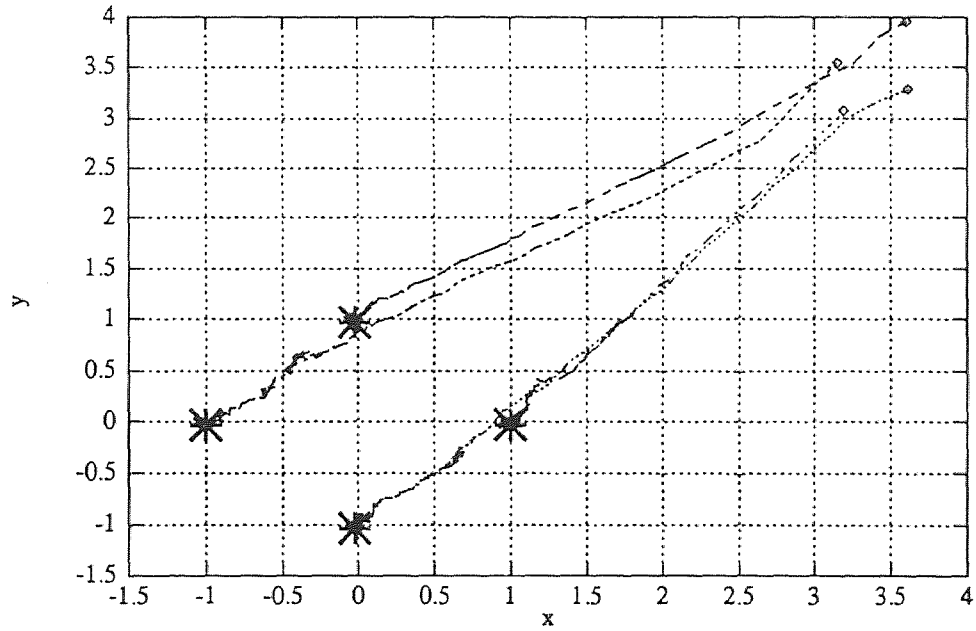

Figure 6.2 The learning trace for $k = 4$ with initial centers at (3.1,3.5), (3.3,3.5), (3.5,3.5), and (3.1,3.1).

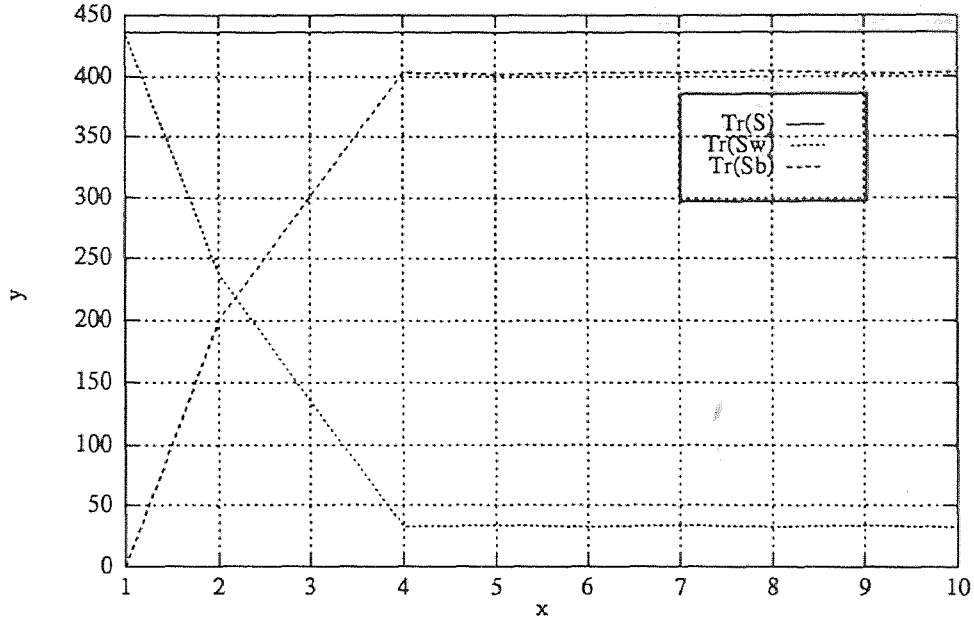

Figure 6.3 $Tr(S)$, $Tr(S_W)$, and $Tr(S_B)$ obtained for four clusters by using SBC for $k = 1 \ldots 10$.

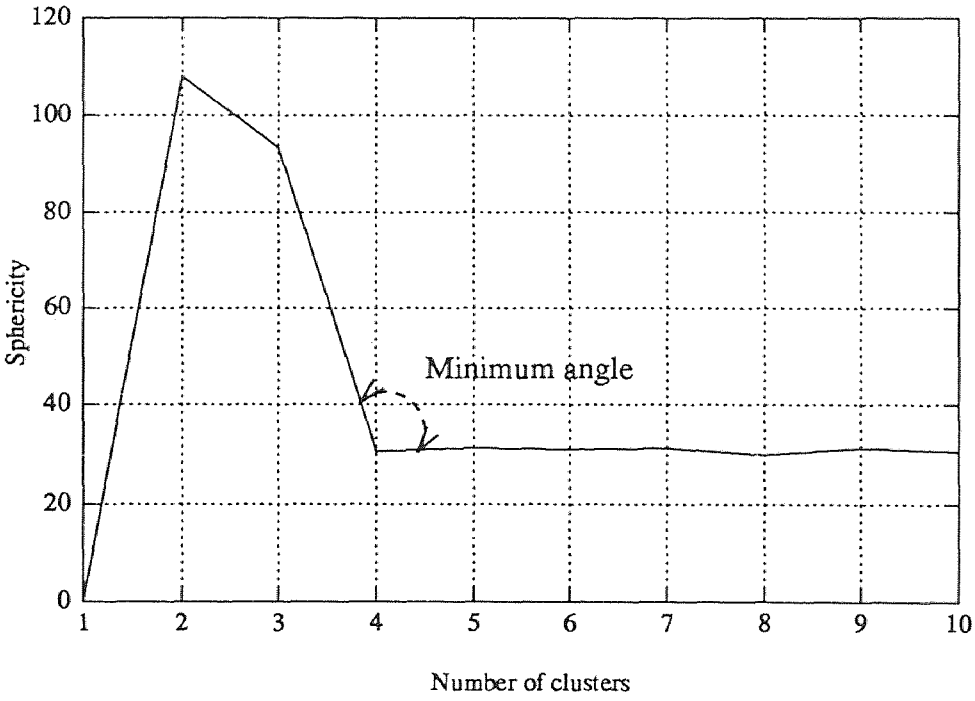

Figure 6.4 Sphericity obtained for four clusters by using SBC for $k = 1 \ldots 10$.

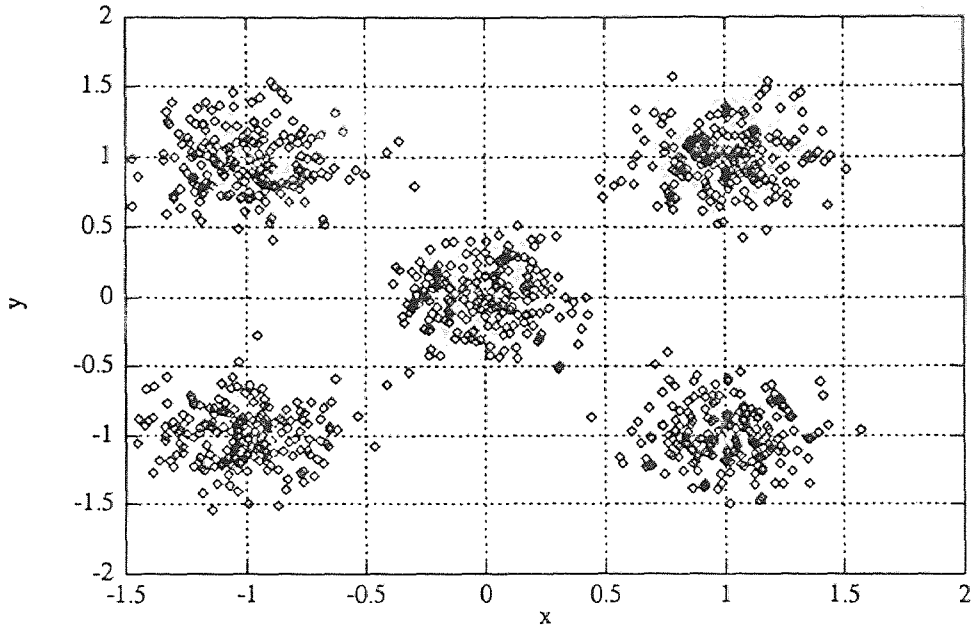

Figure 6.5 Five clusters of data that were used in this simulation centered at (-1.0,-1.0), (1.0,-1.0), (-1.0,1.0), (1.0,1.0), and (0.0,0.0) each with 100 patterns Gaussian distributed with deviation of 0.2.

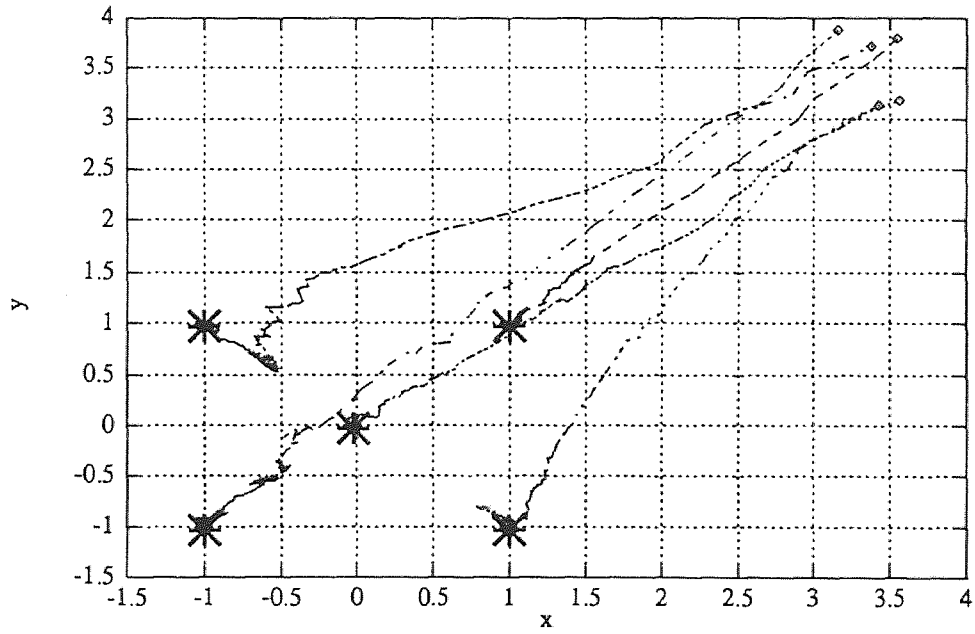

Figure 6.6 The learning trace for $k = 5$ with initial centers at (3.6,3.2), (3.2,3.9), (3.4,3.1), (3.5,3.8), and (3.4,3.7).

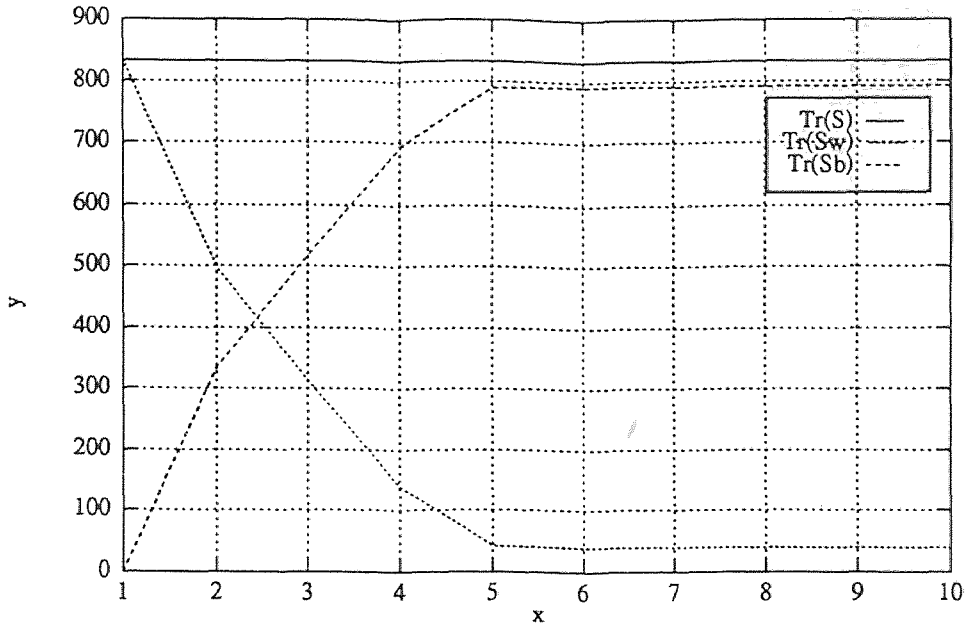

Figure 6.7 $Tr(S)$, $Tr(S_W)$, and $Tr(S_B)$ obtained for five clusters by using SBC for $k = 1 \ldots 10$.

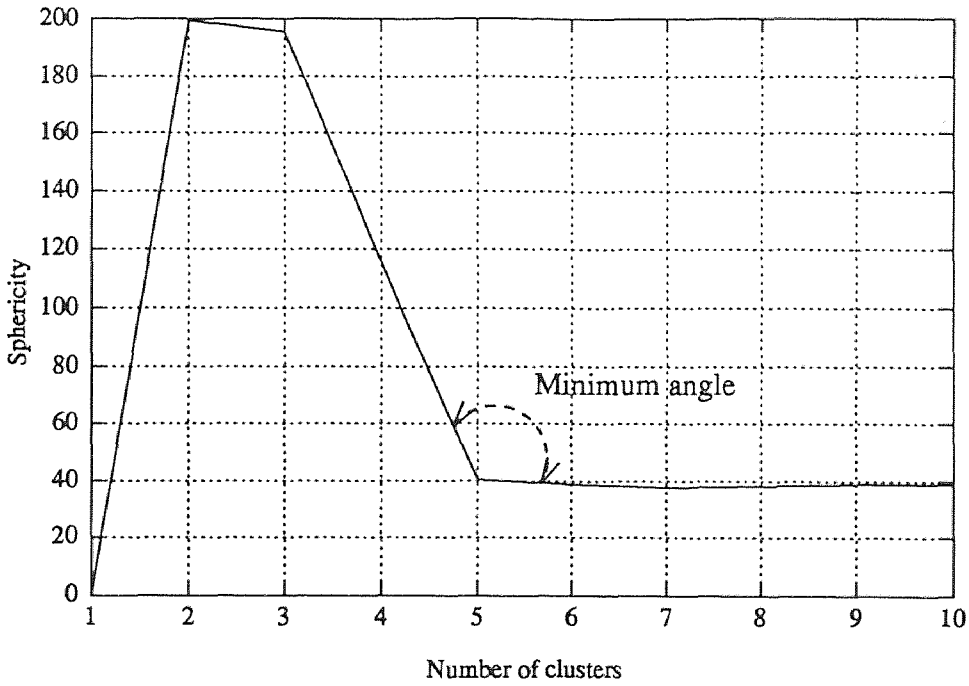

Figure 6.8 Sphericity obtained for five clusters by using SBC for $k = 1 \ldots 10$.

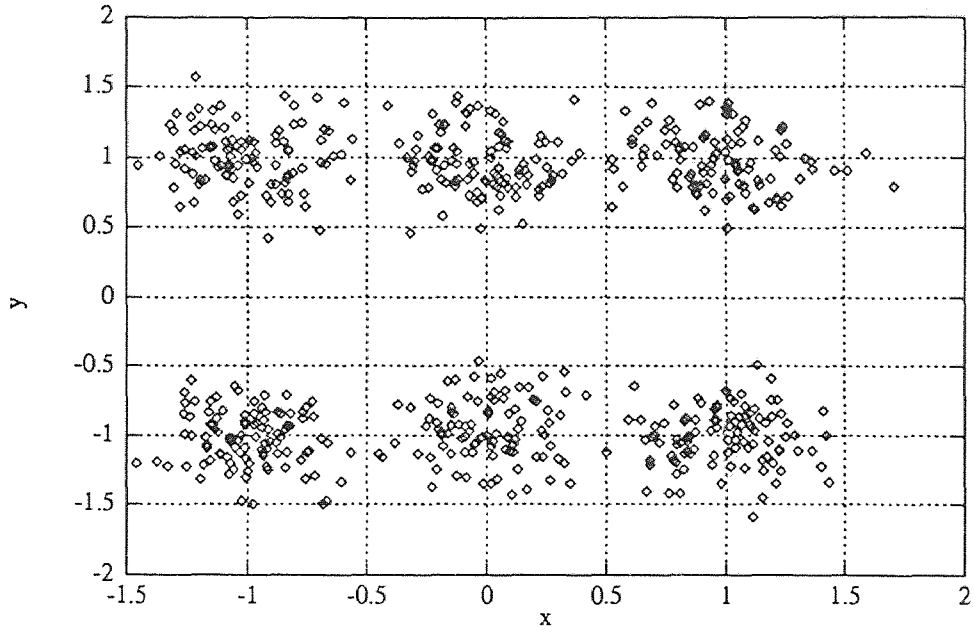

Figure 6.9 Six clusters of data that were used in this simulation centered at (-1.0,-1.0), (1.0,-1.0), (-1.0,1.0), (1.0,1.0), (0.0,1.0), and (0.0,-1.0) each with 100 patterns Gaussian distributed with deviation of 0.2.

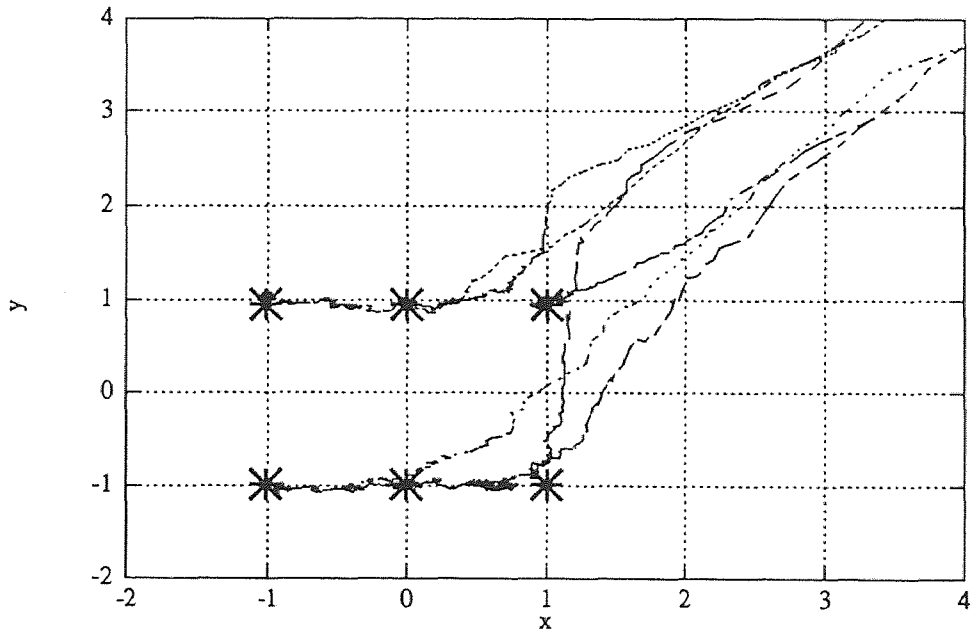

Figure 6.10 The learning trace for $k = 6$ with initial centers at (3.2,3.8), (3.3,3.9), (3.5,3.9), (3.5,3.1), (3.9,3.7), and (3.9,3.7).

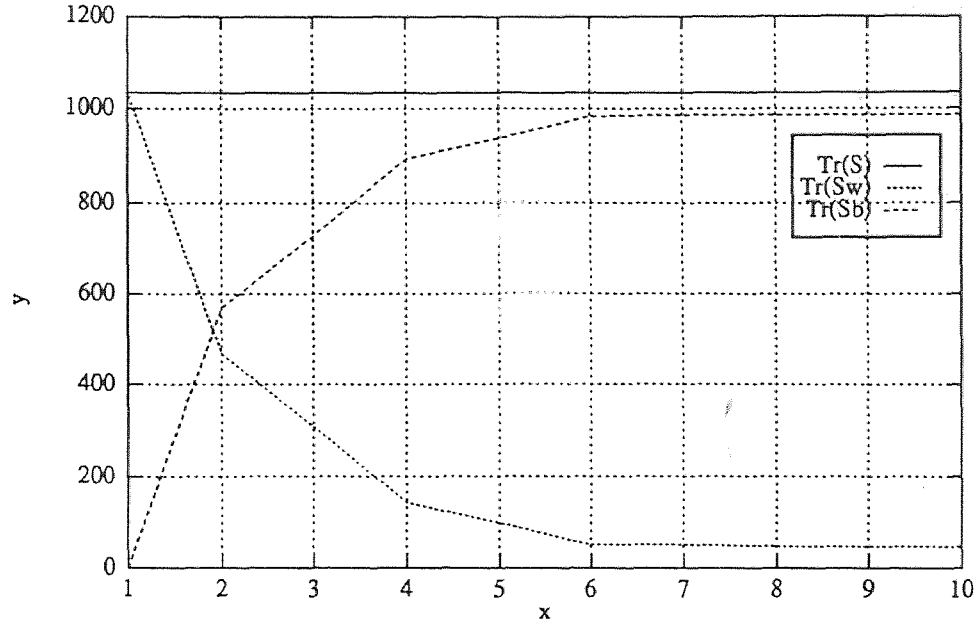

Figure 6.11 $Tr(S)$, $Tr(S_W)$, and $Tr(S_B)$ obtained for six clusters by using SBC for $k = 1 \ldots 10$.

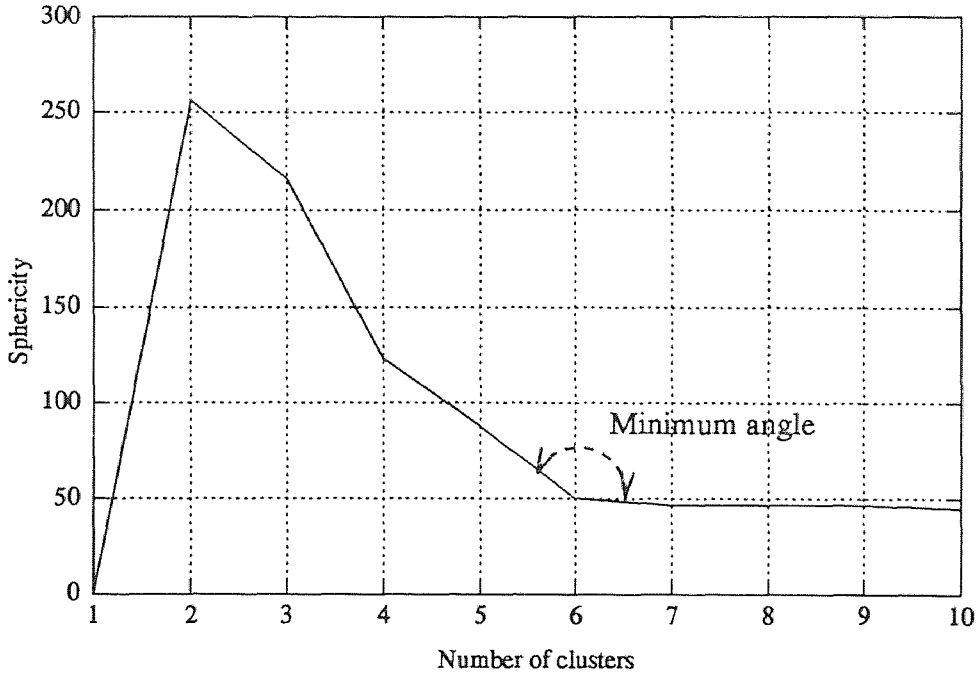

Figure 6.12 Sphericity obtained for six clusters by using SBC for $k = 1 \ldots 10$.

CHAPTER 7
SUPERVISED CLASSIFICATION THROUGH RBF NETWORKS

To illustrate the functionality of the RBF network trained by the RPCL and SBC, the RBF network is utilized to classify the patterns in a "noisy" XOR problem. The data patterns used for this XOR classification are the same as the data patterns in Fig. 6.1, where the patterns are centered at (-1.0, 0.0), (1.0, 0.0), (0.0, 1.0), and (0.0, -1.0). The deviation is the same as before, which is 0.2 with 100 patterns in each cluster. The two clusters centered at (0.0, 1.0) and (0.0, -1.0) form the first class and the other two clusters centered at (-1.0, 0.0) and (1.0, 0.0) form the second class. The basis function used for the response function of the RBF network was the Gaussian function shown previously, in equation (2), with $\sigma = 0.2$.

The first step in this experiment is to create random patterns for training purposes, centered at (-1.0, 0.0), (1.0, 0.0), (0.0, 1.0), and (0.0, -1.0), and the desired output for these patterns. If the patterns are close to (0.0,1.0) and (0.0, -1.0), the desired output of the RBF network is 1, and if the patterns are close to (-1.0, 0.0) and (1.0, 0.0), the desired output of the RBF network is 0. Next, the hidden layer is trained either using the RPCL algorithm or the SBC algorithm. From this training, center units for RBFs are obtained. Using the data pattern, the desired output for this data pattern, and the center units for RBFs, the weights for the output layer are obtained using the pseudo-inverse method. After the training, the RBF network is obtained that classifies the patterns centered at (-1.0, 0.0), (1.0, 0.0), (0.0, 1.0), and (0.0,-1.0) of the XOR problem. To test this RBF network, new test data patterns are created that are centered at the same place, but with different random sampling. These test data patterns are the input for the RBF network to produce output, which should classify the patterns into two classes. The accuracy of the RBF network could be computed by comparing the output of the RBF network with the desired output of the test data. By comparing these computations, one

could compare the functionality of the SBC algorithm with the RPCL algorithm. The results are shown in Table 7.1 through 7.4.

From the tables, correct and incorrect classifications by the RBF network are easily computed. The values under "marginally classified as" indicate how many data patterns are marginally classified as class 1 or class 2. To examine the use of the RPCL algorithm on training the hidden layer of the RBF network in reality, where one does not know the actual number of data clusters, the number of initial center units are set as five, ε=0.05, and $r = 0.005$, which is a reasonable value between 0 and 1. The result of the RBF network with these parameters is shown in Table 7.1, which shows that the recognition rate is only around 52%. Therefore, the RBF network classifies the patterns correctly only half of the time. The reason for such a low rate of recognition is because the rival penalizing force is too strong, which is similar to the simulation results shown in Fig. 5.10, Fig. 5.11, and Fig. 5.12 where too much of the rival penalizing force pushed away more center units than needed. Thus, after decreasing r several times, the optimum value of $r = 0.002$ is obtained. The results are shown in Table 7.2, where the recognition rate is around 98%. The sensitivity of r is a major hindrance toward obtaining the optimal RBF network. Table 7.3 shows the result of initial center units less than the actual number of clusters simulated with $k = 3$ and $r = 0.002$. The recognition rate is around 73%. The obvious reason for the low recognition rate is that there are not enough RBFs to represent the four data clusters. Table 7.4 shows the result of using the SBC algorithm. The optimum number of neural units for the hidden layer was found by searching k that has the minimum angle of sphericity as shown in the previous simulation results, and the value of center units corresponding to that optimal k were used. The recognition rate is around 98%.

	classified as		marginally
	class 1	class 2	classified as
class 1	176	0	24
class 2	16	31	153

Table 7.1 Classification by using RPCL when r=0.005 and $k = 5$

	classified as		marginally
	class 1	class 2	classified as
class 1	194	6	0
class 2	0	196	4

Table 7.2 Classification by using RPCL when r=0.002 and $k = 5$

	classified as		marginally
	class 1	class 2	classified as
class 1	96	102	2
class 2	0	197	3

Table 7.3 Classification by using RPCL when r=0.002 and $k = 3$

	classified as		marginally
	class 1	class 2	classified as
class 1	197	0	3
class 2	0	194	6

Table 7.4 Classification by using SBC

CHAPTER 8

CONCLUSION

Although it has been shown and proved that the RBF network is faster and more flexible compared to classical multi-layered neural networks, the major problem of using the RBF network is the appropriate selection of radial basis function centers. To address and solve this problem, a new learning method based on scatter matrices and sphericity is developed for the construction of the optimal RBF network.

From the simulation results and the application of supervised classification through the RBF network, it has been shown that the CL algorithm, the FSCL algorithm, and the RPCL algorithm are inadequate in training the hidden layer of the RBF network. Among different competitive learning methods, RPCL was the most promising, but it is too sensitive to the learning rate r, and it failed to work when the number of center units chosen was smaller than the actual one. As for the SBC algorithm, it was able to choose the optimal number of center units by selecting k with the minimum angle of sphericity and the optimal value for the center units. By using the characteristics of scatter matrices, the SBC algorithm was far superior in robustness.

REFERENCES

1. A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.

2. B. A. Whitehead and T. D. Choate, "Evolving space-filling curves to distribute radial basis functions over an input space," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 15–23, 1994.

3. D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," *Cognitive Science*, vol. 9, pp. 75–112, 1985.

4. D. M. Hummels, W. Ahmed, and M. T. Musavi, "Adaptive detection of small sinusoidal signals in non-gaussian noise using an RBF neural network," *IEEE Trans. Neural Networks*, vol. 6, no. 1, pp. 214–219, 1995.

5. D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.

6. G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54–115, 1987.

7. G. A. Carpenter and S. Grossberg, "ART 2: Self organization of stable category recognition codes for analog output patterns," *Appl. Opt.*, vol. 26, pp. 4919–4930, 1987.

8. G. A. Carpenter and S. Grossberg, "ART 3: Hierarchical searching using chemical transmitters in self-organizing pattern recognition architectures," *Neural Networks*, vol. 3, pp. 129–152, 1990.

9. L. Xu, A. Krzyzak, and E. Oja, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 636–649, 1993.

10. M. J. D. Powell, "Radial basis function approximations to polynomials," *Proc. 12th Biennial Numerical Analysis Conf.(Dundee)*, pp. 223–241, 1987.

11. N. Ansari and E. J. Delp, "On the distribution of a deforming triangle," *Pattern Recognition*, vol. 23, no. 12, pp. 1333–1341, 1990.

12. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

13. S. C. Ahalt, A. K. Krishnamurty, P. Chen, and D. E. Melton, "Competitive learning algorithms for vector quantization," *Neural Networks*, vol. 3, pp. 277–291, 1990.

14. S. Chen, B. Mulgrew, and P. M. Grant, "A Clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 570–579, 1993.

15. S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.

16. S. Grossberg, "Adaptive pattern classification and universal recording: I. parallel development and coding of neural feature detectors," *Biol. Cybern*, vol. 23, pp. 121–134, 1976.

17. S. Grossberg, "Competitive learning: from iterative activation to adaptive resonance," *Cognitive Science*, vol. 11, pp. 23–63, 1987.

18. S. Haykin, *Neural Networks, A Comprehensive Foundation*, Macmillan, New Jersey, 1994.

19. T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1988.

20. T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, pp. 1464–1480, 1990.

21. X. Yao, "A review of evolutionary artificial neural networks," *International Journal of Intelligent Systems*, vol. 8, no. 4, pp. 539–567, 1993.