EXHIBIT 5
PART
1 OF 2

Keinosuke Fukunaga

Introduction to Statistical Pattern Recognition

Second Edition

Keinosuke Fukunaga

Introduction to Statistical Pattern Recognition

Second Edition

This completely revised second edition presents an introduction to statistical pattern recognition. Pattern recognition in general covers a wide range of problems: it is applied to engineering problems, such as character readers and wave form analysis, as well as to brain modeling in biology and psychology. Statistical decision and estimation, which are the main subjects of this book, are regarded as fundamental to the study of pattern recognition. This book is appropriate as a text for introductory courses in pattern recognition and as a reference book for people who work in the field. Each chapter also contains computer projects as well as exercises.

Introduction to Statistical Pattern Recognition

Second Edition

This is a volume in
COMPUTER SCIENCE AND SCIENTIFIC COMPUTING

Editor: WERNER RHEINBOLDT

Introduction to Statistical Pattern Recognition

Second Edition

Keinosuke Fukunaga
School of Electrical Engineering
Purdue University
West Lafayette, Indiana

Morgan Kaufmann is an imprint of Academic Press

A Harcourt Science and Technology Company

San Diego San Francisco New York Boston
London Sydney Tokyo

This book is printed on acid-free paper. ♾

ACADEMIC PRESS
A Harcourt Science and Technology Company
525 B Street, Suite 1900, San Diego, CA 92101-4495 USA
http://www.academicpress.com

Academic Press
24-28 Oval Road, London NW1 7DX United Kingdom
http://www.hbuk/ap/

Morgan Kaufmann
340 Pine Street, Sixth Floor, San Francisco, CA 94104-3205
http://mkp.com

Library of Congress Cataloging-in-Publication Data
Fukunaga, Keinosuke.
Introduction to statistical pattern recognition / Keinosuke
Fukunaga. — 2nd ed.
p. cm.
Includes bibliographical references.
ISBN 0-12-269851-7
1. Pattern perception — Statistical methods. 2. Decision-making —
— Mathematical models. 3. Mathematical statistics. I. Title.
0327.F85 1990
006.4 — dc20 89-18195
CIP

PRINTED IN THE UNITED STATES OF AMERICA
03 9

Chapter 11

CLUSTERING

In the preceding chapters, we have presented a considerable body of design theory for pattern recognition. Procedures for classifier design, parameter estimation, and density estimation have been discussed in detail. We have consistently assumed the existence of a training set of classified samples. In this chapter, we will focus our attention on the classification of samples without the aid of a training set. We will refer to this kind of classification as *clustering* or *unsupervised classification*.

There are many instances where classification must and can be performed without *a priori* knowledge. Consider, for example, the biological taxonomy problem. Over the years, all known living things have been classified according to certain observable characteristics. Of course, plants and animals have never borne labels indicating their kingdoms, phylae, and so on. Rather, they have been categorized according to their observable characteristics without outside supervision.

The clustering problem is not well defined unless the resulting classes of samples are required to exhibit certain properties. The choice of properties or, equivalently, the definition of a cluster, is the fundamental issue in the clustering problem. Given a suitable definition of a cluster, it is possible to distinguish between good and bad classifications of samples.

In this chapter, two approaches to clustering will be addressed. One is called the *parametric approach* and the other is the *nonparametric approach*.

In most parametric approaches, *clustering criteria* are defined, and given samples are classified to a number of clusters so as to optimize the criteria. The most commonly used criteria are the class separability measures which were introduced in Chapter 10. That is, the class assignment which maximizes the class separability measure is considered to be the best clustering result. In this approach, the structure (parametric form) of the classification boundary is determined by the criterion. The *clustering algorithm*, which determines efficiently the best classification with respect to the criterion, is normally an iterative algorithm. In another parametric approach, a mathematical form is assumed to express the distribution of the given data. A typical example is the summation of normal distributions. In this case, the clustering problem consists of finding the parameter values for this distribution which best fit the data.

On the other hand, neither clustering criteria nor assumed mathematical forms for the distribution are used in the nonparametric approach. Instead, samples are separated according to the *valley* of the density function. The valley may be considered as the natural boundary which separates the modes of the distributions. This boundary could be complex and not expressible by any parametric form.

In addition, clustering may be viewed as the selection of representatives. In general, a density function may be approximated by the Parzen density estimate around the representatives. Then, we may try to reduce the number of representatives while maintaining the degree of approximation. An iterative procedure to choose the representatives is discussed in this chapter.

11.1 Parametric Clustering

In this section, we will present, first, a general-purpose clustering algorithm based on a generalized criterion. Then, the discussion for a specific criterion follows.

General Clustering Algorithm

The clustering algorithm developed in this section applies to a wide range of criteria. However, it is necessary to specify the form of the criterion as well as some other details of the clustering problem at the outset.

Clustering criterion: Assume that we want to classify N samples, $X_1, \dots, X_N$. These vectors are not denoted as random vectors because, in the clustering problem, they are assumed to be fixed and known. Each sample is to be placed into one of L classes, $\omega_1, \dots, \omega_L$, where L is assumed to be given. The class to which the ith sample is assigned is denoted ω_{k_i} ($i = 1, \dots, N$). For convenience, let the value of k_i be an integer between 1 and L. A *classification* Ω is a vector made up of the ω_{k_i}'s, and a *configuration* X^* is a vector made up of the X_i's, that is,

$$\Omega = [\omega_{k_1} \dots \omega_{k_N}]^T \tag{11.1}$$

and

$$X^* = [X_1^T \dots X_N^T]^T \ . \tag{11.2}$$

The clustering criterion J is a function of Ω and X^* and can be written

$$J = J(\omega_{k_1}, \dots, \omega_{k_N}; X_1, \dots, X_N) = J(\Omega; X^*) \ . \tag{11.3}$$

By definition, the best classification Ω_0 satisfies either

$$J(\Omega_0; X^*) = \max_{\Omega} \text{ or } \min_{\Omega} J(\Omega; X^*) \tag{11.4}$$

depending on the criterion. For the remainder of this section, we will discuss only the minimization problem, since the maximization is similar.

Example 1: Six vectors, $X_1, \dots, X_6$, of Fig. 11-1 are given. The problem is to find class assignments of these vectors to one of two classes so as to minimize a criterion. Let $J = \text{tr}(S_m^{-1} S_w)$ be the criterion where S_w and S_m are the within-class and mixture scatter matrices defined in (10.1) and (10.4).

For each classification, for example $\{X_1, X_2, X_5\} \in \omega_1$ and $\{X_3, X_4, X_6\} \in \omega_2$ as shown by dotted lines, or $\{X_1, X_2, X_3\} \in \omega_1$ and $\{X_4, X_5, X_6\} \in \omega_2$ as shown by solid lines, the mean vectors and covariance matrices for ω_1 and ω_2 are estimated, and S_w, S_m, and J can be computed.

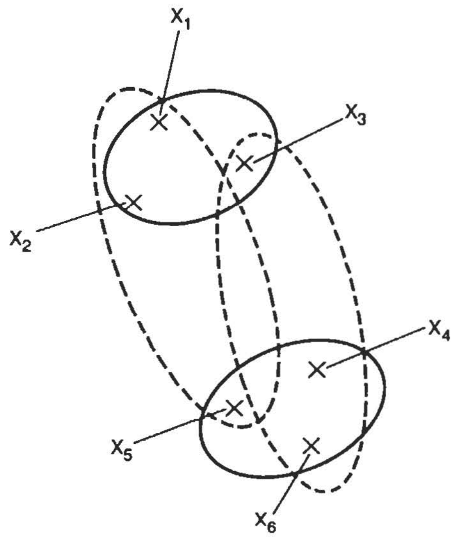

Fig. 11-1 An example of clustering.

Note that S_w varies depending on the class assignment but S_m does not. The class assignment by the solid lines would give the minimum J (the smallest within-class scatter) among all possible combinations of class assignments.

Clustering algorithm: For a given clustering problem, the configuration X^* is fixed. The clustering algorithm varies only the classification Ω. Ordinary steepest descent search techniques cannot be applied because of the discrete and unordered nature of Ω. Still, it is possible to define an iterative search algorithm based on variations in J with respect to variations in Ω.

Suppose, at the ℓth iteration, the classification is $\Omega(\ell)$, where

$$\Omega(\ell) = [\omega_{k_1(\ell)} \dots \omega_{k_N(\ell)}]^T \ . \tag{11.5}$$

If the ith sample is reclassified from its present class $k_i(\ell)$ to class j, the clustering criterion varies by an amount $\Delta J(i,j,\ell)$, which is given by

$$\Delta J(i,j,\ell) = J(\omega_{k_1(\ell)}, \dots, \omega_{k_{i-1}(\ell)}, \omega_j, \omega_{k_{i+1}(\ell)}, \dots, \omega_{k_N(\ell)}; X^*) - J(\Omega(\ell); X^*) \ . \tag{11.6}$$

If $\Delta J(i,j,\ell)$ is negative, reclassification of the ith sample to class j yields a classification that is improved in terms of J. This fact is the basis of the following algorithm:

(1) Choose an initial classification $\Omega(0)$.

(2) Given the ℓth classification $\Omega(\ell)$, calculate $\Delta J(i,j,\ell)$ for $j = 1,2, \ldots ,L$ and $i = 1,2, \ldots ,N$.

(3) For $i = 1,2, \ldots ,N$, reclassify the ith sample to class t, where

$$\Delta J(i,t,\ell) = \min_j \Delta J(i,j,\ell) . \tag{11.7}$$

Decide ties involving $j = k_i(\ell)$ in favor of the present classification. Decide other ties arbitrarily. This step forms $\Omega(\ell + 1)$.

(4) If $\Omega(\ell + 1) \neq \Omega(\ell)$, return to Step (2). Otherwise the algorithm is complete.

The algorithm is simply the iterative application of a classification rule based on the clustering criterion. Here, we adopted a procedure in which all of the samples are reclassified simultaneously at each iteration. An alternative approach is to reclassify each sample one at a time, resulting in similar but slightly different clusters. In these iterative procedures, there is no guarantee that the algorithm converges. Even if it does converge, we cannot be certain that the absolute minimum of J has been obtained. Therefore, we must depend on empirical evidence to justify the algorithm.

In contrast to these potential weaknesses, the algorithm described above is very efficient. Like any good search algorithm, it surveys a small subset of classifications in a systematic and adaptive manner. It is easily programmable for any criterion of the form of (11.3).

Determining the number of clusters: So far, we have ignored the problem of determining the number of classes L, and have assumed that L is given. However, in practice, L is rarely known. We not only need to determine L but also the proper class assignment. For that purpose, we may run the clustering procedure for the various values of L, and find the best classification for each value of L. Let J_L^* be the optimal criterion value for a given L after the clustering procedure has converged. If J_L^* decreases as L increases, and either reaches the minimum point at L_0 or becomes flat after L_0, then we may use L_0 as the proper number of classes. Unfortunately, many of the popular criteria do not have this favorable property. For example, consider $J = \mathrm{tr}(S_m^{-1} S_w)$ of Example 1. As L increases, samples are divided into smaller groups, and consequently the within-class scatter becomes smaller. This means that J might decrease monotonically with L. Finally, when L becomes N, the total number of samples, each class consists of one sample only, and there is no within-class scatter ($J = 0$). Although $L = N$ minimizes this criterion, this is obviously not the solution we want.

It appears, therefore, that some external method of controlling L is necessary. Unfortunately, no unified theory for determining L has been fully developed and accepted.

Merging and splitting: After a number of classes is obtained, we may consider the merging of two classes into a single class or the splitting of a class into a number of classes.

Basically, merging is desirable in two instances. The first is when two classes are very similar. The similarity may be measured in a number of ways. The Euclidean distance between two mean vectors is the simplest measure but is not an accurate one. The Bhattacharyya distance of (3.152), based on the normal assumption, could be a reasonable compromise between simplicity and accuracy. The second instance in which merging may be appropriate is when the population of a class is very small. In this case, the class may be merged with the most similar class, even when the similarity is not very close.

Deciding whether or not a class is to be split is a more complex problem. Too large a population suggests that the class is a candidate for a split. Multimodal and nonsymmetric distributions as well as distributions with large variances along one direction are also candidates for splitting. In order to identify these characteristics, various tests are necessary. Splitting a class may be carried out by applying the clustering procedure to the samples in the class.

It goes without saying that this kind of merging and splitting is very heuristic. Its merit lies in the fact that it is efficient and requires a minimum of human interaction.

Multiple dichotomy: It is somewhat more satisfying to adopt an approach which depends entirely on the clustering criterion J. One such approach has been suggested [1] and is outlined as follows.

Suppose that for $L = 2$ there are several distinct classifications which yield a nearly minimal value of J. If these classifications differ only in the classification of a few samples, there is no reason to suppose the existence of more than two classes. If the classifications are grossly different, however, then it is evident that several classes are actually present.

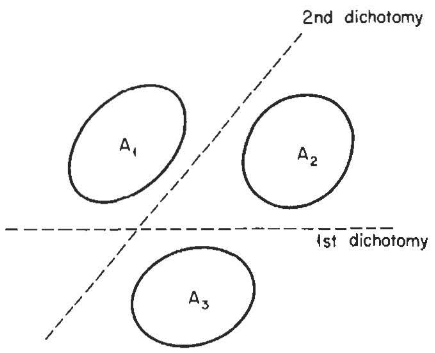

Fig. 11-2 Multiple dichotomy of three classes of samples.

Figure 11-2 illustrates two possible dichotomies of a collection of samples apparently containing three classes A_1, A_2, and A_3. One dichotomy separates the samples into $A_1 \cup A_2$ and A_3, while the other results in the two classes A_1 and $A_2 \cup A_3$. Thus, A_3, A_1, and $\overline{A_1 \cup A_3} = A_2$ are seen to be distinct classes ($\bar{A}$ is the complement of the set A.)

Now let us consider the more general case where there are k dichotomies of a collection of samples containing L classes. Each dichotomy separates these samples into two groups. Let S_{ij} be the set of all samples assigned to group j by the ith dichotomy for $j = 1,2$ and $i = 1, \ldots, k$. Assume that the following two conditions hold.

(a) A dichotomy places each class into only one group, that is, classes are not split by dichotomies.

(b) For each pair of classes, there is at least one dichotomy which does not assign the two classes to the same group.

Select one group from each of the k dichotomies and form a subset C as the intersection of these k groups. By condition (a), if C contains one sample, then it must contain all of the samples of that class. By condition (b), for any other class, there is at least one of the k selected groups to which that class does not belong. Therefore, if C is nonempty, then C contains one and only one class. Hence, in order to construct all the L classes, we consider the 2^k subsets of the form.

$$C(j_1,\ldots,j_k) = \bigcap_{i=1}^{k} S_{ij_i} \, , \tag{11.8}$$

where each j equals 1 or 2. Each nonempty C is a class. In our example, we have

$$S_{11} = A_1 \cup A_2 , \quad S_{12} = A_3 , \quad S_{21} = A_1 , \quad S_{22} = A_2 \cup A_3 \, , \tag{11.9}$$

so that

$$\begin{aligned} C(1,1) &= S_{11} \cap S_{21} = A_1 \, , \\ C(1,2) &= S_{11} \cap S_{22} = A_2 \, , \\ C(2,1) &= S_{12} \cap S_{21} = 0 \, , \\ C(2,2) &= S_{12} \cap S_{22} = A_3 \, , \end{aligned} \tag{11.10}$$

which is in agreement with our earlier argument.

The multiple dichotomy approach has a stronger theoretical basis than the merging and splitting procedure. Further, it relies on no numerical criterion other than J. However, implementation of the multiple dichotomy approach can be difficult, especially when the true number of classes is large. In addition, the conditions (a) and (b) mentioned above are rarely satisfied in practice. These difficulties may be overcome somewhat by imposing a hierarchical structure on the classes. The samples are divided into a small number of classes, each class is divided further, and so on. Under this strategy, we need not find every possible dichotomy of the entire collection of samples.

At this point, we depart from general discussion of the clustering algorithm. Obviously, the discussion is incomplete. We have a basis, however, to develop and implement clustering procedures. Therefore, let us turn our attention to the detailed derivations of clustering procedures.

Nearest Mean Reclassification Algorithm [2-5]

In this section, we will discuss clustering procedures based on parameters such as mean vectors and covariance matrices. We will show that the criteria of class separability discussed in Chapter 10 play an important role, and that the iterative algorithms of the previous section take on simple forms.

Criteria: Clustering can be considered as a technique to group samples so as to maximize class separability. Then, all of the criteria which were discussed in Chapter 10 may be used as clustering criteria. In this section only functions of scatter matrices are discussed due to the following reasons:

(1) The extension to multiclass problems is straightforward. In this respect, the Bhattacharyya distance has a severe disadvantage, since it can be applied only to two-class problems.

(2) Most clustering techniques are based on the scatter of mean vectors. Finding clusters based on covariance-differences is extremely difficult, unless some mathematical structure is imposed on the distribution. Therefore, the functions of scatter matrices fit well to clustering problems.

(3) The simplicity of the criteria is a significant advantage, because in clustering we have the additional complexity of unknown class assignment.

For feature extraction, we could choose any combination of S_b, S_w, and S_m as S_1 and S_2 to form a criterion $J = \mathrm{tr}(S_2^{-1} S_1)$. However, for clustering it is preferable to use S_m as S_2, because S_m is independent of class assignment. It would be too complicated if we had to recompute the inverse of S_2 each time the class assignment was altered in iteration. Therefore, our choice is limited to either $\mathrm{tr}(S_m^{-1} S_b)$ or $\mathrm{tr}(S_m^{-1} S_w)$. These two criteria are the same, because $\mathrm{tr}(S_m^{-1} S_b) = \mathrm{tr}\{S_m^{-1}(S_m - S_w)\} = n - \mathrm{tr}(S_m^{-1} S_w)$. In this chapter, we will use $J = \mathrm{tr}(S_m^{-1} S_w)$.

Another important consideration in selecting criteria for clustering is to ensure that the clustering procedures give the same classification for a given set of samples regardless of the coordinate system of these samples. The chosen criterion, $J = \mathrm{tr}(S_m^{-1} S_w)$, satisfies this condition, since the criterion is invariant under any nonsingular linear transformation.

Clustering algorithm: Let us assume that $M_0 = 0$ and $S_m = I$ without losing generality. If the given samples do not satisfy these conditions, we can shift the coordinate origin and whiten the data with respect to S_m. Then, using (10.1) the criterion is rewritten as

$$J = \operatorname{tr} S_w = \sum_{r=1}^{L} \frac{N_r}{N} \frac{1}{N_r} \sum_{j=1}^{N_r} (X_j^{(r)} - M_r)^T (X_j^{(r)} - M_r)$$

$$= \frac{1}{N} \sum_{r=1}^{L} \sum_{j=1}^{N_r} \| X_j^{(r)} - M_r \|^2 \ . \tag{11.11}$$

Changing the class assignment of X_i from the current class k_i to class j at the ℓth iteration, we delete from (11.11) the term $\|X_i - M_{k_i}(\ell)\|^2$ and insert a new term $\|X_i - M_j(\ell)\|^2$. Thus,

$$\Delta J(i,j,\ell) = \frac{1}{N} \{ \|X_i - M_j(\ell)\|^2 - \|X_i - M_{k_i}(\ell)\|^2 \} \ . \tag{11.12}$$

Since the second term of (11.12) is independent of j, the reclassification of X_i at the ℓth iteration can be carried out by

$$\|X_i - M_t(\ell)\| = \min_j \|X_i - M_j(\ell)\| \quad \rightarrow \quad X \in \omega_t \ . \tag{11.13}$$

In words, the algorithm becomes:

(1) Choose an initial classification, $\Omega(0)$, and calculate $M_1(0), \ldots, M_L(0)$.

(2) Having calculated sample mean vectors $M_1(\ell), \ldots, M_L(\ell)$ at the ℓth iteration, reclassify each X_i according to the nearest $M_j(\ell)$.

(3) If the classification of any X_i is changed, calculate the new sample mean vectors $M_1(\ell + 1), \ldots, M_L(\ell + 1)$ for the new class assignment, and repeat from Step (2). Otherwise, stop.

This algorithm is called the *nearest mean reclassification rule*.

Figure 11-3 shows how the iterative process works. At the ℓth step, samples are divided to three clusters, and their sample means, $M_i(\ell)$'s, are computed. All samples are now reclassified according to the nearest means. That is, the new boundary is piecewise linear, bisecting each pair of $M_i(\ell)$'s. In Fig. 11-3, there are three clearly separated clusters. We can see that the boundary is indeed improved by this operation.

From the above discussion, some properties of the nearest mean reclassification algorithm become evident. They are:

(1) Clusters are divided by piecewise linear bisectors. Only the means contribute to determine the boundary and covariance matrices do not affect the boundary.

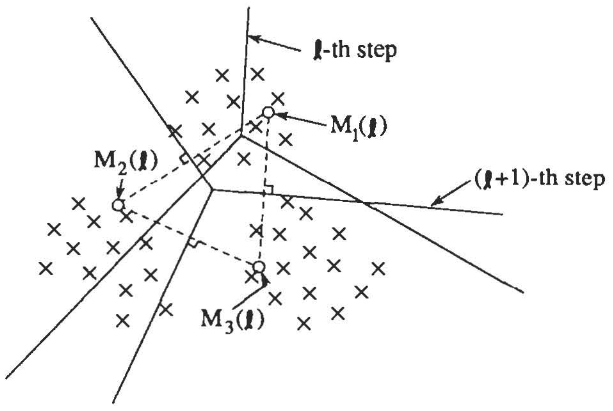

Fig. 11-3 An example of the nearest mean reclassification algorithm.

(2) The number of clusters must be preassigned.

(3) The initial classification, $\Omega(0)$, may be given randomly. No matter how $\Omega(0)$ is given, the $M_i(0)$'s are computed and the reclassification of samples according to the nearest $M_i(0)$'s results in a piecewise linear boundary. This is equivalent to selecting the number of vectors, $M_i(0)$'s, initially according to the number of clusters. Random class assignment does not impose any extra instability on the algorithm.

In order to verify the algorithm, the following experiment was conducted.

Experiment 1: One hundred samples were generated from each of Data I-Λ, and mixed together to form 200 samples. Then, these samples were classified to two clusters. Table 11-1 shows the *confusion matrix* of this experiment [5]. All 100 samples from ω_1 with 19 samples from ω_2 were assigned to one cluster, and 81 samples from ω_2 are assigned to the other cluster. The error rate is $19/200 = 0.095$. Recall that we got 5% error for this data by designing the optimum linear classifier in Chapter 4. Considering the fact that any covariance information was not used in this clustering algorithm, the error rate of 9.5% is reasonable. Furthermore, since all error samples came from one class, we could improve the error rate simply by adjusting the decision threshold.

Convergence [5]: The nearest mean reclassification algorithm is not guaranteed to converge. In this section, we will discuss the conditions under

TABLE 11-1

CONFUSION MATRIX FOR THE NEAREST MEAN RECLASSIFICATION ALGORITHM

		Assigned class	
		1	2
Actual class	1	100	0
	2	19	81

which the separating hyperplane converges for two normal distributions with equal covariance matrices.

Let us assume that two normal distributions are $N_X(M_1,\Sigma_1)$ and $N_X(M_2,\Sigma_2)$ after normalization and that $\Sigma_1 = \Sigma_2 = \Sigma$. The normalization makes $S_m = I$ and $M_0 = 0$. The Bayes classifier in this case becomes linear as

$$(M_2 - M_1)^T\Sigma^{-1}X + c = 0\ , \tag{11.14}$$

where c is a constant. Since $S_m = I = \Sigma + P_1M_1M_1^T + P_2M_2M_2^T$ and $M_0 = 0 = P_1M_1 + P_2M_2$,

$$\Sigma = I - P_1M_1M_1^T - P_2M_2M_2^T = I - \frac{P_2}{P_1}M_2M_2^T\ , \tag{11.15}$$

$$M_2 - M_1 = \frac{1}{P_1}M_2\ . \tag{11.16}$$

Using (2.160),

$$\Sigma^{-1} = I + \frac{\dfrac{P_2}{P_1}M_2M_2^T}{1 - \dfrac{P_2}{P_1}M_2^TM_2}\ . \tag{11.17}$$

Substituting (11.16) and (11.17) into (11.14), the Bayes classifier is

$$\frac{1}{P_1 - P_2M_2^TM_2}M_2^TX + c = 0\ . \tag{11.18}$$

Thus, the optimum hyperplane for the equal covariance case is always in the direction of M_2 which is the same as the mean-difference vector M_2-M_1. This property, which the original coordinate system does not have, is a significant advantage of the normalized coordinate system.

For the equal covariance case, we can show that the algorithm converges to M_2-M_1 from a wide range of initial classifications. After a given iteration, samples are separated by a hyperplane whose direction is, say V ($\|V\| = 1$), as shown in Fig. 11-4. Also, the position of the hyperplane is specified by ℓ_1 and

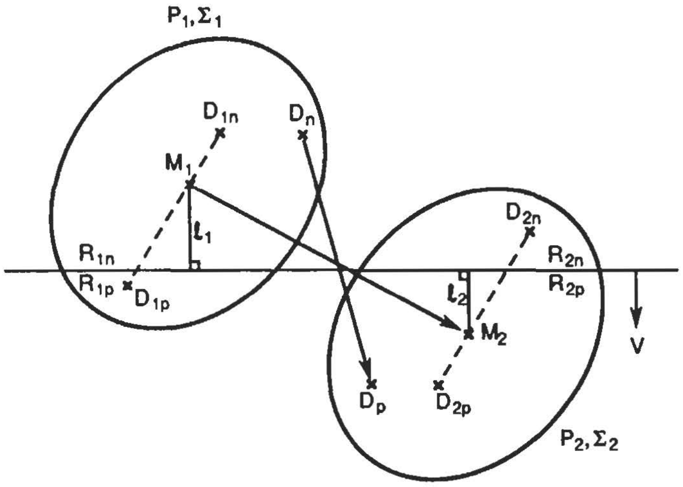

Fig. 11-4 Separation of two distributions.

ℓ_2 which are the distances of M_1 and M_2 from the hyperplane. Let D_p and D_n be the centers of probability mass for the positive and negative sides of the hyperplane. Then, following the nearest mean reclassification rule, the direction of the succeeding hyperplane will be D_p-D_n. So, our convergence proof is to show that the angle between D_p-D_n and M_2-M_1 is smaller than the angle between V and M_2-M_1.

Since the hyperplane separates each distribution into two parts, the positive and negative sides, we have four probability masses R_{ij} ($i = 1,2$; $j = p,n$), as shown in Fig. 11-4. Let D_{ij} and q_{ij} be the mean vectors and populations of these probability masses. Then,

$$D_{ip} = M_i + \frac{a_i}{\sigma_i b_i}\Sigma_i V \ , \tag{11.19}$$

$$D_{in} = M_i - \frac{a_i}{\sigma_i(1-b_i)}\Sigma_i V \ , \tag{11.20}$$

$$q_{ip} = P_i b_i \ , \tag{11.21}$$

$$q_{in} = P_i(1-b_i) \ , \tag{11.22}$$

where

$$a_i = \frac{1}{\sqrt{2\pi}}\int_{\pm\ell_i/\sigma_i}^{\infty} \zeta \exp[-\frac{1}{2}\zeta^2]d\zeta = \frac{1}{\sqrt{2\pi}}\exp[-\frac{1}{2}\left[\frac{\ell_i}{\sigma_i}\right]^2] \ , \tag{11.23}$$

$$b_i = \frac{1}{\sqrt{2\pi}}\int_{\pm\ell_i/\sigma_i}^{\infty} \exp[-\frac{1}{2}\zeta^2]d\zeta = 1 - \Phi\left[\pm\frac{\ell_i}{\sigma_i}\right] \ , \tag{11.24}$$

$$\sigma_i^2 = V^T\Sigma_i V \ , \tag{11.25}$$

and Φ is the normal error function. The sign + or − is selected, depending on whether M_i is located in R_{in} or R_{ip}. The details of the derivation of (11.19) through (11.25) are left to the reader. However, the following information could be helpful in deriving (11.19) through (11.25). Since $\mathbf{X}$ is normal, $\mathbf{y} = V^T\mathbf{X}$ is also normal with the variance of (11.25). In the y-space, the probability of mass for the positive side, b_i, can be computed by (11.24). The vector $D_{ij} - M_i$ has the direction of $\Sigma_i V$, and the projections of $D_{ij} - M_i$ on V $(j = p, n)$ are $V^T(D_{ip} - M_i) = a_i\sigma_i/b_i$ and $V^T(D_{in} - M_i) = -a_i\sigma_i/(1-b_i)$. These are obtained by computing the expected values of $\mathbf{y}$ for the positive and negative sides. From D_{ij} and q_{ij}, D_p and D_n are obtained as

$$D_p = \frac{q_{1p}D_{1p} + q_{2p}D_{2p}}{q_{1p} + q_{2p}} \ , \tag{11.26}$$

$$D_n = \frac{q_{1n}D_{1n} + q_{2n}D_{2n}}{q_{1n} + q_{2n}} \ . \tag{11.27}$$

Substituting (11.19) through (11.22) into (11.26) and (11.27),

$$D_p - D_n = \frac{P_1P_2(b_1-b_2)(M_1-M_2)+\{P_1(a_1/\sigma_1)\Sigma_1+P_2(a_2/\sigma_2)\Sigma_2\}V}{(P_1b_1+P_2b_2)\{1-(P_1b_1+P_2b_2)\}} . \tag{11.28}$$

For the equal covariance case,

$$\Sigma_1 = \Sigma_2 = \Sigma \quad \text{and} \quad \sigma_1 = \sigma_2 = \sigma . \tag{11.29}$$

Furthermore, under the normalization of $S_m = I$ and $M_0 = 0$, Σ and $M_2 - M_1$ are expressed as functions of M_2 as in (11.15) and (11.16). Therefore, (11.28) becomes

$$D_p - D_n = c_1M_2 + c_2V , \tag{11.30}$$

where

$$c_1 = \frac{P_2(b_2-b_1)-(1/\sigma)(P_2/P_1)(P_1a_1+P_2a_2)M_2^TV}{(P_1b_1+P_2b_2)\{1-(P_1b_1+P_2b_2)\}} , \tag{11.31}$$

$$c_2 = \frac{(1/\sigma)(P_1a_1+P_2a_2)}{(P_1b_1+P_2b_2)\{1-(P_1b_1+P_2b_2)\}} . \tag{11.32}$$

The normal of the new hyperplane has a component in the direction of V and another in the direction of M_2. If the coefficient of M_2, c_1, has the same sign as M_2^TV, the successive hyperplane becomes more nearly parallel to M_2. Since c_2 and the denominator of c_1 are positive, we need to show that the numerator of c_1 and M_2^TV have the same sign. We examine only the case where $M_2^TV > 0$. The discussion for $M_2^TV < 0$ is similar to the one for $M_2^TV > 0$. For $M_2^TV > 0$, we see from Fig. 11-4 that

$$\ell_1 + \ell_2 = (M_2-M_1)^TV = \frac{1}{P_1}M_2^TV . \tag{11.33}$$

Using (11.33), the condition for convergence becomes

$$b_2 - b_1 > \frac{\ell_1 + \ell_2}{\sigma}(P_1a_1 + P_2a_2) . \tag{11.34}$$

It is easily seen that the inequality of (11.34) is not satisfied for certain combinations of parameters. However, the region of parameters where (11.34) is satisfied can be calculated numerically. The result is shown in Fig. 11-5.

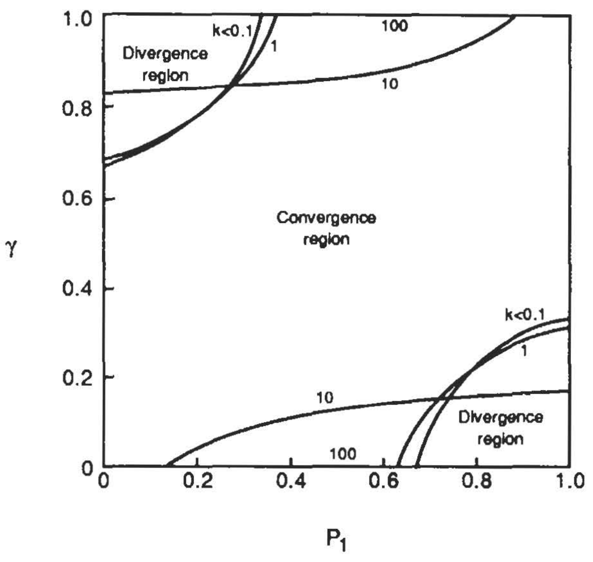

Fig. 11-5 Region of convergence.

Equations (11.23) and (11.24) with (11.34) show that we have three parameters in (11.34), ℓ_1/σ, ℓ_2/σ, and P_1 ($P_2 = 1 - P_1$), or

$$k = \frac{\ell_1+\ell_2}{\sigma}\ , \quad \gamma = \frac{\ell_1}{\ell_1+\ell_2}\ , \quad \text{and} \quad P_1\ . \tag{11.35}$$

In Fig. 11-5, the convergence regions of γ and P_1 are plotted for various values of k [5]. The figure shows that convergence is quite likely in practice, except for either extreme P_1's or γ's.

Branch and bound procedure [6]: The nearest mean reclassification algorithm works fine for many applications. However, there is no guarantee of the convergence of the iterative process. Also, the process might stop at a locally minimum point and fail to find the globally minimum point.

Since assigning a class to each sample is a combinatorial problem, the *branch and bound procedure* discussed in Chapter 10 may be applied to find the optimum class assignment.

Figure 11-6 shows a solution tree for the clustering problem with four samples and three clusters. In general, there are L^N different Ω's for classify-

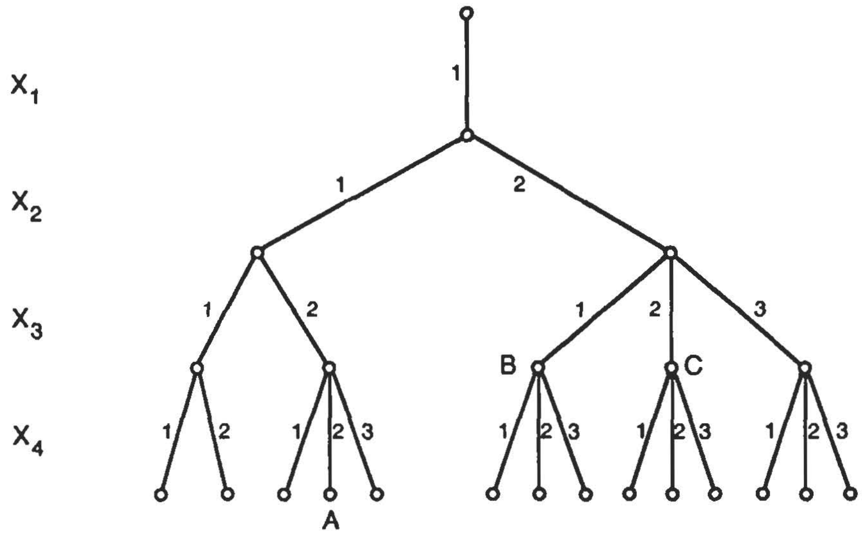

Fig. 11-6 A solution tree for clustering.

ing N sample vectors into L clusters. However, since the label of each cluster may be chosen arbitrarily, each classification could have several different expressions for Ω. For example, $\Omega = [1\ 1\ 2\ 2]$ and $\Omega = [2\ 2\ 3\ 3]$ are the same classification, both indicating that X_1 and X_2 are in one cluster and X_3 and X_4 are in one of the other clusters. In order to eliminate this duplication, we assign the first sample X_1 to cluster 1, the second sample X_2 to either cluster 1 or 2, and so on, as shown in Fig. 11-6.

In order for the branch and bound procedure to work effectively, we need to have a criterion which satisfies the *monotonicity condition*. Let $J_m(\Omega_m)$ be the criterion to be minimized for $\Omega_m = [\omega_{k_1} \ldots \omega_{k_m}]$, where the subscript m indicates the number of sample vectors involved. Then, the monotonicity condition is stated as

$$J_{m+1}(\Omega_m, \omega_{k_{m+1}}) \geq J_m(\Omega_m) \ . \tag{11.36}$$

That is, the J of a node is smaller than the J's for all nodes which are successors of the node.

Let α be the value of J_N which is the current smallest among all cases tested so far for the classification of N samples (for example, $\alpha = J(A)$). Then, the branch and bound procedure checks at each node (for example, at B) whether or not the following inequality is satisfied

$$J_m(\Omega_m) \geq \alpha \, . \tag{11.37}$$

If yes, then from (11.36), all successors of this node have J's larger than α. It means that the optimum classification does not exist in the subtree under the node. Thus, the subtree is rejected without further tests, and the search backtracks to the next node (for example, C). This elimination of subtrees makes the branch and bound procedure a very efficient tree search technique. When (11.37) is not satisfied, the search moves down to a node in the next level. The node selected for the next evaluation is determined by

$$J_{m+1}(\Omega_m, t) = \min_{\ell} J_{m+1}(\Omega_m, \ell) \, . \tag{11.38}$$

That is, X_{m+1} is assigned to cluster t, and the search goes on.

The criterion $J = \mathrm{tr}(S_m^{-1} S_w)$ satisfies the monotonicity condition with a minor modification. Again, assuming $S_m = I$, (11.11) is the criterion to be minimized. Since the number of samples is determined by the level of the solution tree and is independent of Ω, let us delete it from the criterion and rewrite the criterion for m samples, $X_1, \ldots, X_m$, as

$$J_m(\Omega_m) = \sum_{r=1}^{L} \sum_{j=1}^{m_r} \| X_j^{(r)} - M_r \|^2 \, , \tag{11.39}$$

where m_r is the number of ω_r-samples among $X_1, \ldots, X_m$. When X_{m+1} is added into cluster ℓ, M_ℓ must be modified to M_ℓ', including the effect of X_{m+1}, and $\| X_{m+1} - M_\ell' \|^2$ must be added to the summation. Thus,

$$J_{m+1}(\Omega_m, \ell) = J_m(\Omega_m) + \Delta J(\ell) \, , \tag{11.40}$$

where

$$\Delta J(\ell) = \| X_{m+1} - M_\ell' \|^2 + \sum_{j=1}^{m_\ell} \{ \| X_j^{(\ell)} - M_\ell' \|^2 - \| X_j^{(\ell)} - M_\ell \|^2 \} \, . \tag{11.41}$$

The new ℓ-class mean, M_ℓ', can be obtained as

$$\begin{aligned} M_\ell' &= \frac{1}{m_\ell + 1} \left(\sum_{j=1}^{m_\ell} X_j^{(\ell)} + X_{m+1} \right) \\ &= M_\ell + \frac{1}{m_\ell + 1} (X_{m+1} - M_\ell) \, , \end{aligned} \tag{11.42}$$

or

$$X_j^{(\ell)} - M_\ell' = (X_j^{(\ell)} - M_\ell) - \frac{1}{m_\ell + 1}(X_{m+1} - M_\ell) \,. \tag{11.43}$$

Substituting (11.43) into (11.41) and using $M_\ell = (1/m_\ell)\Sigma_{j=1}^{m_\ell} X_j^{(\ell)}$,

$$\Delta J(\ell) = \frac{m_\ell}{m_\ell + 1}\|X_{m+1} - M_\ell\|^2 \geq 0 \,. \tag{11.44}$$

Since $\Delta J(\ell) \geq 0$, from (11.40) the criterion has the monotonicity property.

Note that (11.40), (11.44), and (11.42) provide recursive equations for computing $J_{m+1}(\Omega_m,\ell)$ and M_ℓ' from $J_m(\Omega_m)$, M_ℓ, and X_{m+1}. Also, (11.38), (11.40), and (11.44) indicate that the selection of the next node can be made by minimizing $\Delta J(\ell)$ with respect to ℓ.

For a large N, the number of nodes is huge. Thus, the initial selection of α becomes critical. One way of selecting a reasonably low initial α is to apply the iterative nearest mean reclassification algorithm to get a suboptimal solution and use the resulting criterion value as the initial α. The branch and bound procedure gives the final solution which is guaranteed to be optimum globally.

Also, it is possible to make the procedure more efficient by reordering the samples [6].

Normal Decomposition

Piecewise quadratic boundary: The nearest mean reclassification rule can be extended to include more complex boundaries such as quadratic ones. Following the same iterative process, the algorithm would be stated as follows:

(1) Choose an initial classification, $\Omega(0)$, and calculate $P_i(0)$, $M_i(0)$ and $\Sigma_i(0)$ $(i = 1, \ldots, L)$.

(2) Having calculated class probabilities, $P_i(\ell)$, and sample means and covariance matrices, $M_i(\ell)$ and $\Sigma_i(\ell)$, at the ℓth iteration, reclassify each X_j according to the smallest $(1/2)(X_j - M_i)^T \Sigma_i^{-1}(X_j - M_i) + (1/2)\ln|\Sigma_i| - \ln P_i$. The class probability for ω_i is estimated by the number of ω_i-samples divided by the total number of samples.

(3) If the classification of any X_j is changed, calculate the $P_i(\ell+1)$, $M_i(\ell+1)$ and $\Sigma_i(\ell+1)$ for the new class assignment, and repeat from Step (2).

Otherwise stop.

This process is identical to the nearest mean reclassification algorithm, but results in a piecewise quadratic boundary. Also, since the estimation of the covariance matrices is involved, the process is more computer-time consuming and more sensitive to parameters such as sample size, dimensionality, distributions, and so on.

More fundamentally, the clustering techniques mentioned above may have a serious shortcoming, particularly when a mixture distribution consists of several overlapping distributions. An important goal of finding clusters is to decompose a complex distribution into several normal-like distributions. If we could approximate a complex distribution by the summation of several normal distributions, it would be much easier to discuss all aspects of pattern recognition, including feature extraction, classifier design, and so on. However, the clustering procedures discussed above decompose a mixture as in Fig. 11-7(b) rather than as in Fig. 11-7(a). The hatched distribution of cluster 1 in Fig. 11-7(b) includes the tail of the ω_2-distribution and does not include the tail of the ω_1-distribution. As a result, the mean and covariance matrix of the hatched distribution in Fig. 11-7(b) could be significantly different from the ones for the hatched distribution of Fig. 11-7(a). Thus, the representation of a complex distribution by the parameters obtained from the clustering procedures described above could be very poor.

Decomposition of a distribution into a number of normal distributions has been studied extensively [7]. The two most common approaches are the *method of moments* and *maximum likelihood estimation*. In the former method, the parameters of normal distributions are estimated from the higher order moments of the mixture distribution (for example, the third and fourth order moments [8]). This approach is complex and not very reliable for high-dimensional cases. Therefore, in this chapter, only the latter approach is presented in detail.

Maximum likelihood estimate [9-10]: In order to obtain the hatched distribution of Fig. 11-7(a) from $p(X)$, it is necessary to impose a mathematical structure. Let us assume that $p(X)$ consists of L normal distributions as

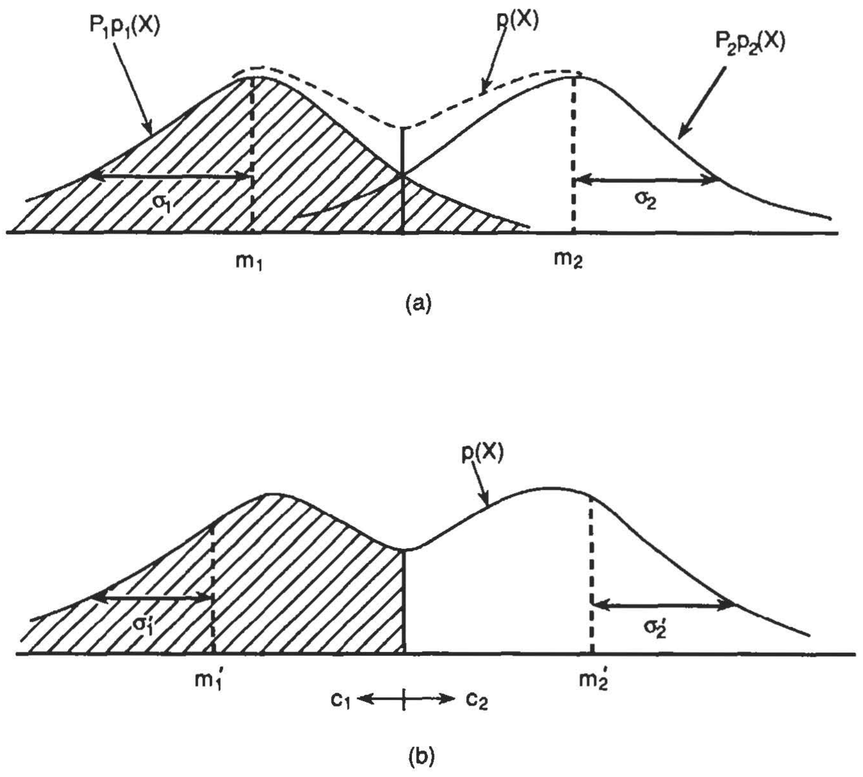

Fig. 11-7 An example of the shortcoming of the clustering technique.

$$p(X) = \sum_{i=1}^{L} P_i p_i(X) \,, \tag{11.45}$$

where $p_i(X)$ is normal with the expected vector M_i and covariance matrix Σ_i. Under this assumption, our problem is to estimate P_i, M_i, and Σ_i from N available samples, $X_1, \ldots, X_N$, drawn from $p(X)$. One way of solving this problem is to apply the *maximum likelihood estimation* technique. The maximum likelihood estimates may be obtained by maximizing $\Pi_{j=1}^{N} p(X_j)$ with respect to P_i, M_i, and Σ_i under a constraint $\Sigma_{i=1}^{L} P_i = 1$. Taking the logarithm of $\Pi_{j=1}^{N} p(X_j)$, the criterion to be maximized is

$$J = \sum_{j=1}^{N} \ln p(X_j) - \mu\left(\sum_{i=1}^{L} P_i - 1\right) \,, \tag{11.46}$$

where μ is a Lagrange multiplier.

First, computing the derivative of J with respect to P_i,

$$\frac{\partial J}{\partial P_i} = \sum_{j=1}^{N} \frac{p_i(X_j)}{p(X_j)} - \mu = \frac{1}{P_i} \sum_{j=1}^{N} q_i(X_j) - \mu = 0 \,, \tag{11.47}$$

where $q_i(X) = P_i p_i(X)/p(X)$ is the a posteriori probability of ω_i, and satisfies $\Sigma_{i=1}^{L} q_i(X) = 1$. Since $\Sigma_{i=1}^{L} P_i(\partial J/\partial P_i) = \Sigma_{j=1}^{N}(\Sigma_{i=1}^{L} q_i(X_j)) - (\Sigma_{i=1}^{L} P_i)\mu = N - \mu = 0$,

$$\mu = N \,, \tag{11.48}$$

and from (11.47)

$$P_i = \frac{1}{N} \sum_{j=1}^{N} q_i(X_j) \,. \tag{11.49}$$

Next, the derivative of J with respect to M_i can be computed as

$$\frac{\partial J}{\partial M_i} = \sum_{j=1}^{N} \frac{P_i}{p(X_j)} \frac{\partial p_i(X_j)}{\partial M_i} = \sum_{j=1}^{N} q_i(X_j) \Sigma_i^{-1}(X_j - M_i) = 0 \,. \tag{11.50}$$

Using $\Sigma_{j=1}^{N} q_i(X_j) = N P_i = N_i$ where N_i is the number of ω_i-samples, (11.50) can be solved for M_i, resulting in

$$M_i = \frac{1}{N_i} \sum_{j=1}^{N} q_i(X_j) X_j \,. \tag{11.51}$$

At last, the derivative of J with respect to Σ_i is, from (A.9) and (A.23),

$$\begin{aligned} \frac{\partial J}{\partial \Sigma_i} &= \sum_{j=1}^{N} \frac{P_i}{p(X_j)} \frac{\partial p_i(X_j)}{\partial \Sigma_i} \\ &= \sum_{j=1}^{N} q_i(X_j)[\Sigma_i^{-1}(X_j - M_i)(X_j - M_i)^T \Sigma_i^{-1} - \Sigma_i^{-1} \\ &\quad - \frac{1}{2} \text{diag}[\Sigma_i^{-1}(X_j - M_i)(X_j - M_i)^T \Sigma_i^{-1} - \Sigma_i^{-1}]] = 0, \end{aligned} \tag{11.52}$$

where diag$[A]$ is a diagonal matrix, keeping only the diagonal terms of A. Equation (11.52) can be solved for Σ_i yielding

$$\Sigma_i = \frac{1}{N_i} \sum_{j=1}^{N} q_i(X_j)(X_j - M_i)(X_j - M_i)^T \,. \tag{11.53}$$

By solving (11.49), (11.51), and (11.53), we can obtain the optimum solution. However, since $q_i(X)$ is a function of P_k, M_k, and Σ_k ($k = 1, \ldots, L$), it is very difficult to obtain the solution explicitly. Therefore, we must solve these equations iteratively. The process can be described as follows.

(1) Choose an initial classification, $\Omega(0)$, and calculate P_i, M_i, and Σ_i ($i = 1, \ldots, L$).

(2) Having calculated $P_i^{(\ell)}$, $M_i^{(\ell)}$, and $q_i^{(\ell)}(X_j)$, compute $P_i^{(\ell+1)}$, $M_i^{(\ell+1)}$, and $\Sigma_i^{(\ell+1)}$ by (11.49), (11.51), and (11.53), respectively. The new $q_i^{(\ell+1)}(X_j)$ can be calculated as

$$q_i^{(\ell+1)}(X_j) = \frac{P_i^{(\ell+1)} p_i^{(\ell+1)}(X_j)}{\sum_{k=1}^{L} P_k^{(\ell+1)} p_k^{(\ell+1)}(X_j)} , \tag{11.54}$$

where the superscript indicates the (ℓ+1)st iteration, and $p_i^{(\ell+1)}(X)$ is a normal density function with mean $M_i^{(\ell+1)}$ and covariance matrix $\Sigma_i^{(\ell+1)}$. Note that each sample X_j carries L probability values $q_1(X_j), \ldots, q_L(X_j)$ instead of being assigned to one of the L classes.

(3) When $q_i^{(\ell+1)}(X_j) = q_i^{(\ell)}(X_j)$ for all $i = 1, \ldots, L$ and $j = 1, \ldots, N$, then stop. Otherwise, increase ℓ by 1 and go to Step (2).

In the maximum likelihood estimation technique, the criterion (the first term of (11.46)) may be used to determine the number of clusters. The maximized criterion value, J_L, is obtained for a given L, and the procedure is repeated for various values of L. The criterion J_L tends to increase with increasing L, and reach a flat plateau at L_0. This means that, even if we use more normal distributions than L_0, the mixture distribution cannot be better approximated. Therefore, L_0 is the proper number of clusters to use in approximating the mixture distribution.

In order to verify the procedure, the following two experiments were run.

Experiment 2: One hundred samples per class were generated from two-dimensional normal distributions specified by

$$M_1 = M_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} , \quad \Sigma_1 = \begin{bmatrix} 1 & -0.7 \\ -0.7 & 1 \end{bmatrix} , \quad \text{and} \quad \Sigma_2 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix} . \tag{11.55}$$

The sample means and covariance matrices of the generated samples were

$$\hat{M}_1 = \begin{bmatrix} -0.06 \\ 0.21 \end{bmatrix}, \quad \hat{\Sigma}_1 = \begin{bmatrix} 0.91 & -0.65 \\ -0.65 & 1.17 \end{bmatrix},$$

$$\hat{M}_2 = \begin{bmatrix} 0.11 \\ 0.19 \end{bmatrix}, \quad \hat{\Sigma}_2 = \begin{bmatrix} 1.03 & 0.62 \\ 0.62 & 0.81 \end{bmatrix}. \tag{11.56}$$

These two hundred samples were mixed, and initially assigned to either ω_1 or ω_2 depending on $x_2 \geq 0$ or $x_2 < 0$ (x_2 is the second variable). After 10 iterations, the parameters became

$$\hat{P}_1 = 0.61 \ , \quad \hat{P}_2 = 0.39 \ ,$$

$$\hat{M}_1 = \begin{bmatrix} 0.07 \\ 0.36 \end{bmatrix}, \quad \hat{\Sigma}_1 = \begin{bmatrix} 0.87 & -0.47 \\ -0.47 & 1.04 \end{bmatrix},$$

$$\hat{M}_2 = \begin{bmatrix} -0.04 \\ -0.06 \end{bmatrix}, \quad \hat{\Sigma}_2 = \begin{bmatrix} 1.12 & 0.66 \\ 0.66 & 0.78 \end{bmatrix}. \tag{11.57}$$

Note that the two distributions share the same mean and are heavily overlapped. This means that finding clusters must be very difficult. Despite the expected difficulty, the procedure found two reasonable clusters successfully. Without imposing the mathematical structure of (11.45), no other clustering technique works properly for this example.

Experiment 3: One hundred samples per class were generated from 8-dimensional normal distributions of Data *I*-Λ, and initially assigned to either ω_1 or ω_2 depending on $x_8 \leq 0$ or $x_8 > 0$ (x_8 is the eighth variable). After 20 iterations, samples were classified to either ω_1 or ω_2, depending on whether $q_1(X) > q_2(X)$ or $q_1(X) < q_2(X)$. Table 11-2 shows the resulting confusion matrix. This error of 2.5% is very close to the Bayes error of 1.9%, and is much better than the 9.5% error of the nearest mean reclassification algorithm [see Table 11-1].

In order to confirm that the mixture distribution was properly decomposed into two normal distributions, a quadratic classifier was designed based on P_i, M_i, and Σ_i obtained from the two clusters. Independently, 100 samples per class were generated according to Data *I*-Λ, and classified by the quadratic classifier. The resulting error was 2.5%. Considering the fact that the holdout method (design and test samples are selected independently) always produces an error larger than the Bayes error, the designed classifier was very closed to the Bayes.

TABLE 11-2

CONFUSION MATRIX FOR THE MAXIMUM LIKELIHOOD ESTIMATION ALGORITHM

		Assigned class	
		1	2
Actual class	1	98	2
	2	3	97

In order to determine the proper number of clusters, the experiment was repeated for various values of L. For a given L, P_i and $p_i(X)$ $(i = 1, \ldots L)$ of (11.45) were estimated, and subsequently the first term of (11.46), $J = \Sigma_{j=1}^{N} \ln p(X_j)$, was computed. Figure 11-8 shows J/N vs. L. The curves are flat for $L \geq 2$ and $N = 400$, 800, indicating that two normal distributions are adequate to represent this data. The number of samples assigned to each cluster is N/L on the average. Therefore, when N/L becomes smaller (for example $N/L = 50$ for $N = 200$ and $L = 4$), each cluster may not have an adequate sample size to estimate the covariance matrix properly. This is the reason that the curve decreases as L increases for $N = 200$ in Fig. 11-8.

So far, we have presented the recursive equations to estimate a priori probabilities, mean vectors, and covariance matrices. However, in some applications, we can assume some of the parameter values or the relationship among the parameters as follows:

(1) all covariance matrices are equal [see Problem 2],

(2) all mean vectors are equal, or

(3) a priori probabilities are known [see Problem 3].

With the above conditions, the maximum likelihood estimation technique can be applied to estimate the remaining parameters. Because of the additional information, we can obtain a better estimate with faster convergence.

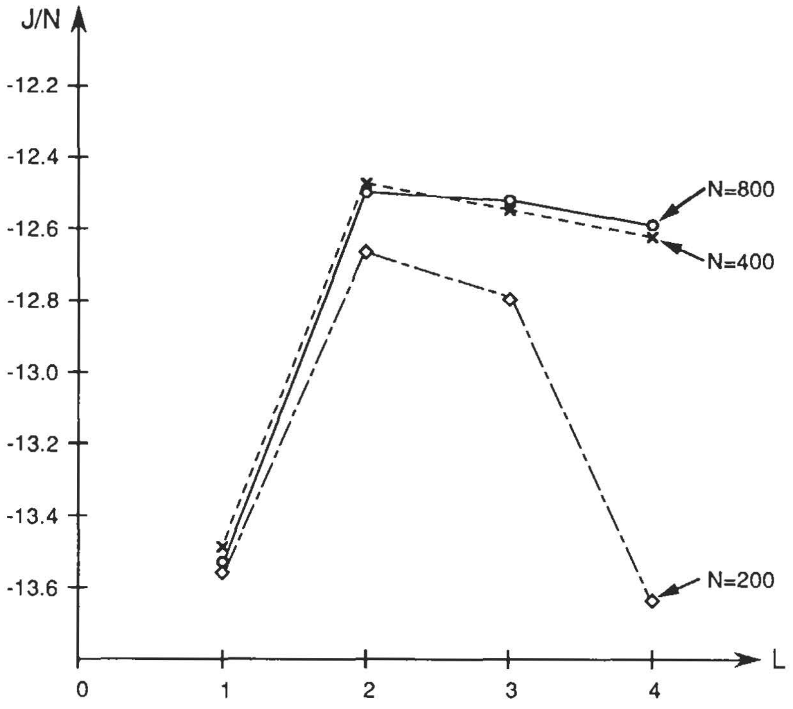

Fig. 11-8 Criterion value vs. L for Data I-Λ.

11.2 Nonparametric Clustering

When a mixture density function has peaks and valleys as shown in Fig. 11-9, it is most natural to divide the samples into clusters according to the valley. However, the valley may not have a parametric structure such as hyperplanes, quadratic surfaces, and so on. As discussed in the previous section, the parametric structure of the boundary comes from either the use of a parametric criterion or from the underlying assumption that the distribution consists of several normal distributions. For the distribution of Fig. 11-9, we cannot expect to get reasonable clusters by a parametric boundary.

In order to find the valley of a density function in a high-dimensional space, we need a nonparametric technique to characterize the local structure of the valley. There are many nonparametric clustering procedures available. However, most of them are implicitly or explicitly based on the estimate of the density gradient. In Fig. 11-9, if we estimate the gradient of the density func-

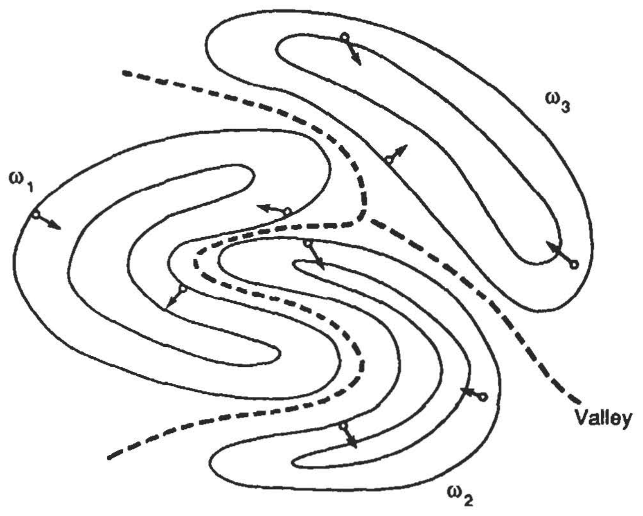

Fig. 11-9 Clusters separated by the valley.

tion at each existing sample point (as indicated by arrows) and move the sample toward the direction of the gradient, samples move away from the valley area. Repeating this process, the valley becomes wider at each iteration, and samples form compact clusters. This procedure is called the *valley-seeking* procedure.

Thus, the valley-seeking procedure consists of two problems: one is how to estimate the gradient of a density function, and the other is how to utilize the estimate to form clusters.

Estimation of Density Gradient

In this section, we develop the estimation technique of the density gradient, and discuss how to apply this technique to pattern recognition problems.

Estimation of density gradient [11]: In order to estimate the gradient of a density function at X, let us select an ellipsoidal local region $\Gamma(X)$ with radius r, specified by

$$\Gamma(X) = \{Y: d(Y,X) \leq r\} , \tag{11.58}$$

where

$$d^2(Y,X) = (Y-X)^T A^{-1}(Y-X) \tag{11.59}$$

and A is the metric to measure the distance. The expected vector of $\mathbf{Y}$ in $\Gamma(X)$, which is called the *local mean*, can be computed as

$$M(X)=E\{(\mathbf{Y}-X)\mid\Gamma(X)\}=\int_{\Gamma(X)}(Y-X)\frac{p(Y)}{u_0}dY\ , \tag{11.60}$$

where

$$u_0 = \int_{\Gamma(X)} p(Y)dY \cong p(X)v \tag{11.61}$$

and v is the volume of $\Gamma(X)$. The term u_0 is the coverage of $\Gamma(X)$, and $p(Y)/u_0$ of (11.60) gives the conditional density function of $\mathbf{Y}$ given $\Gamma(X)$. Expanding $p(Y)$ around X by a Taylor series

$$p(Y) \cong p(X) + (Y-X)^T \nabla p(X)\ . \tag{11.62}$$

Substituting (11.61) and (11.62) into (11.60),

$$M(X)\cong\int_{\Gamma(X)}(Y-X)(Y-X)^T\frac{1}{v}dY\frac{\nabla p(X)}{p(X)} = \frac{r^2}{n+2}A\frac{\nabla p(X)}{p(X)} \tag{11.63}$$

or

$$\frac{\nabla p(X)}{p(X)} \cong \frac{n+2}{r^2}A^{-1}M(X)\ , \tag{11.64}$$

where the integration of (11.63) is obtained from (B.6). Equation (11.64) indicates that, by measuring the local mean $M(X)$ in $\Gamma(X)$, $\nabla p(X)/p(X)$ can be estimated. Particularly, the formula becomes simpler if the Euclidean metric $A = I$ is used.

The normalization of $\nabla p(X)$ by $p(X)$ has an advantage, particularly in clustering. In clustering, it is desirable that samples around the valley area have stronger signal as to which direction the gradients point. Since $p(X)$ is low at the valley, $\nabla p(X)$ is amplified by being divided by $p(X)$. On the other hand, at the peak area, $p(X)$ is high and $\nabla p(X)$ is depressed by being divided by $p(X)$.

Figure 11-10 illustrates how the local mean is related to the gradient of a density function. In $\Gamma(X)$, we tend to have more samples from the higher density side than from the lower density side. As a result, the sample mean of local samples generally points in the direction of the higher density side.

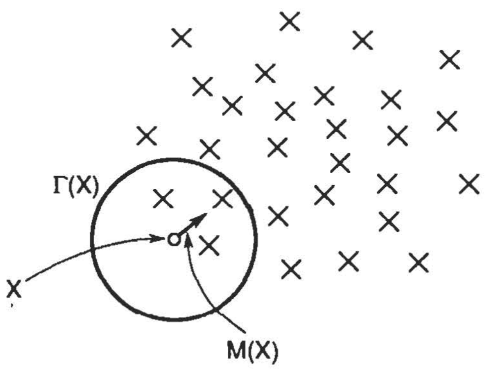

Fig. 11-10 Local mean as the gradient estimate.

The estimate of the density gradient can be applied to many pattern recognition problems besides clustering. They are briefly discussed as follows.

Gradient of $q_i(X)$: The Bayes classifier is the hypersurface on which X satisfies $q_1(X) = q_2(X) = 0.5$ for two-class problems. The vector perpendicular to this hypersurface at X is the gradient of $q_1(X)$, $\nabla q_1(X)$, which indicates the local linear classifier, classifying local samples Y around X as

$$\nabla q_1^T(X)(Y-X) \underset{\omega_2}{\overset{\omega_1}{\lessgtr}} 0 . \tag{11.65}$$

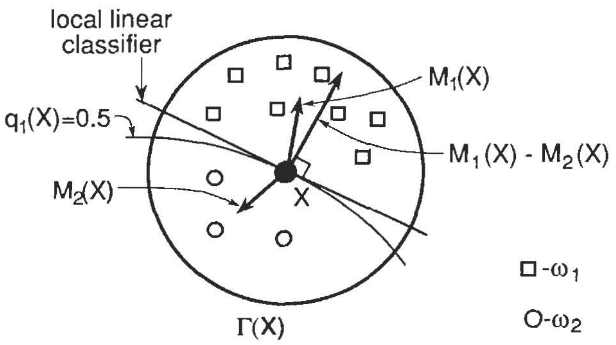

Fig. 11-11 The gradient of $q_1(X)$.

Figure 11-11 shows an example. Note that $\nabla q_2(X) = -\nabla q_1(X)$, since $q_1(X) + q_2(X) = 1$.

The gradient $\nabla q_1(X)$ can be estimated from the local means as

$$\begin{aligned}\nabla q_1(X) &= \nabla \left[\frac{P_1 p_1(X)}{p(X)} \right] \\ &= q_1(X) q_2(X) \left[\frac{\nabla p_1(X)}{p_1(X)} - \frac{\nabla p_2(X)}{p_2(X)} \right] \\ &= \frac{n+2}{r^2} q_1(X) q_2(X) [M_1(X) - M_2(X)] \,, \end{aligned} \tag{11.66}$$

where the Euclidean metric $A = I$ is used. Since $q_1(X)q_2(X)$ is a scalar, the direction of the vector $\nabla q_1(X)$ is determined by $M_1(X) - M_2(X)$, as shown in Fig. 11-11.

Normality test [12]: When $p(X)$ is normal with zero mean and covariance matrix I, $\nabla p(X)/p(X)$ can be obtained by differentiating $\ln p(X)$ with respect to X [see (B.11)], resulting in

$$\frac{\nabla p(X)}{p(X)} = -X \,. \tag{11.67}$$

Equation (11.67) suggests that, by adding the estimate of $\nabla p(X)/p(X)$ to X, the resulting vector should point toward the coordinate origin if **X** is normally distributed. This property can be used to test the normality of a given set of samples. The procedure is described as follows.

(1) Whiten the samples. After the whitening process, the samples have zero mean and covariance matrix I.

(2) Estimate $\nabla p(X)/p(X)$ by the local mean $M(X)$ of (11.60), and add it to X. Use $A = I$.

(3) The data passes the normality test by satisfying

$$\frac{1}{N} \sum_{i=1}^{N} \| X_i + \frac{n+2}{r^2} M(X_i) \|^2 < t \,, \tag{11.68}$$

where t is a threshold. Various properties of this test as well as the selection procedures of related parameters, including t, can be found in [12].

Data filter [11]: A data filter eliminates noise from a given set of samples and produces the *skeleton hypersurface*. The filter could be an effective tool for determining the intrinsic dimensionality of samples. Figure 11-12

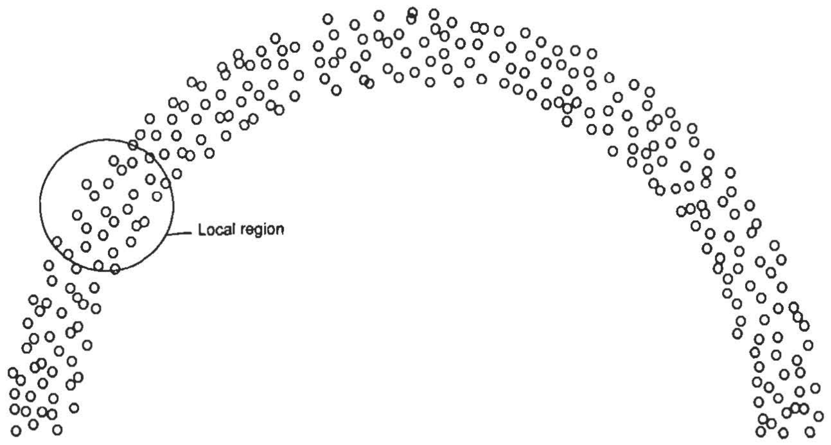

Fig. 11-12 Noisy data set.

shows a distribution of samples which, judged subjectively, is intrinsically one- or two-dimensional. Let us assume that the distribution is one-dimensional, and that unwanted noise is responsible for the two-dimensional scatter. As discussed in Chapter 6, the intrinsic dimensionality is determined by observing the local dimensionalities. Selecting a local region, as shown in Fig. 11-12, the dimensionality in the local region is two, because the two-dimensional scatter of noise has the same order of magnitude as the size of the local region. In order to eliminate the noise, we can measure the density gradient at each sample point X_j, and move X_j toward the direction of the gradient. The amount of the move could be controlled by another parameter, which is multiplied to the local mean vector $M(X_j)$. Repeating this process, samples are merged to a curve having little two-dimensional scatter. This curve is the skeleton of the distribution. After obtaining the skeleton, the local dimensionality is measured, which is one in the example of Fig. 11-12.

Clustering Algorithms

After estimating the gradient of a density function, we now need an algorithm to find clusters. As discussed in data filter, one way of finding clusters is to move samples toward the direction of the gradient by $\rho M(X)$ where ρ is a control parameter. The procedure must be repeated until all samples in each cluster converge to one vector. This is conceptually simple, but computationally cumbersome. So, if possible, we would like to change only class assignment without altering sample vectors. Also, it is preferable to avoid iterative operations. There are many ways to accomplish this. However, in this section we present only two: one is a non-iterative process, and the other is an iterative one.

Graph theoretic approach [13]: One way to avoid an iterative operation is to form trees as shown in Fig. 11-13. In this figure a node representing

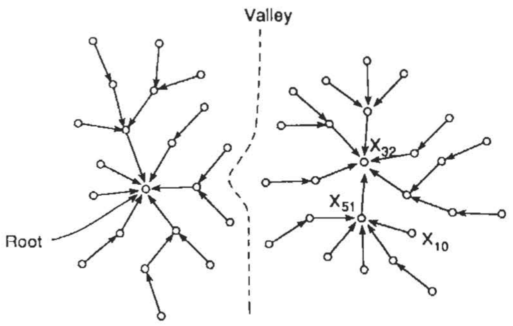

Fig. 11-13 Graph theoretic clustering.

X_{10} initiates a branch (or an arrow) pointing another node X_{51}, which is called the *predecessor* of X_{10}. Then, X_{51} initiates another branch to point to X_{32}, and so on. Thus, each sample becomes an initial node and leads into a final node through a series of branches, each branch pointing from one node to its predecessor. A series of branches is called a *directed path*. We will implement an algorithm such that there is no directed path from a node to itself (i.e. no cycle). At the top of a tree, the *final node* (such as X_{32}) does not have a predecessor and is called the *root* of a tree. Note that each node except the final node has one and only one predecessor, but each could be the predecessor of a number of nodes, including zero. This type of tree is called a *directed tree*. Since the concept of this tree-formation comes from graph theory, we call this the *graph theoretic approach.*

In order to form directed trees for the clustering problem, we need an algorithm to select the predecessor of each sample. If we could select, as the predecessor of X, a sample along the steepest ascent path starting from X, samples will be divided by the valley of the density function, and a tree is formed for each cluster as shown in Fig. 11-13. The quality of the result depends wholly on the quality of the estimation technique for the density gradient, particularly in the low density area of the valley.

The density gradient at X can be estimated by the local mean $M(X)$ as in (11.64). Asymptotically, the sample at the local mean can be the predecessor of X. However, in practice, with a finite number of samples, none of the existing local samples in $\Gamma(X)$ is located exactly at the local mean. Therefore, we need a procedure to pick up an existing local sample which is closest to the local mean.

When two samples are located close together, the steepness of the slope from X_j to X_ℓ can be measured by

$$s_{\ell j} = \frac{p(X_\ell) - p(X_j)}{\|X_\ell - X_j\|} . \tag{11.69}$$

Then, the predecessor X_k is the one which has the steepest slope from X_j among samples in $\Gamma(X_j)$, satisfying

$$s_{kj} = \max_{X_\ell \in \Gamma(X_j)} s_{\ell j} . \tag{11.70}$$

Equation (11.69) has another interpretation. Expanding $p(X_\ell)$ around X_j by a Taylor series

$$\begin{aligned} p(X_\ell) &\cong p(X_j) + \nabla p^T(X_j)(X_\ell - X_j) \\ &= p(X_j) + \|X_\ell - X_j\| \nabla^T p(X_j) \frac{X_\ell - X_j}{\|X_\ell - X_j\|} . \end{aligned} \tag{11.71}$$

Thus

$$s_{\ell j} \cong \nabla^T p(X_j) \frac{X_\ell - X_j}{\|X_\ell - X_j\|} = \|\nabla p(X_j)\| \cos\theta_{\ell j} , \tag{11.72}$$

where $\theta_{\ell j}$ is the angle between the two vectors, $\nabla p(X_j)$ and $(X_\ell - X_j)$. Since $\|\nabla p(X_j)\|$ is independent of ℓ, (11.70) and (11.72) suggest that X_k is the sample which gives the smallest angle between $\nabla p(X_j)$ and $(X_k - X_j)$ among all local samples in $\Gamma(X_j)$. That is, X_k is the closest sample to the steepest ascent line from X_j. Thus, the predecessor of X_j can be determined by measuring the angle between the local mean and $(X_\ell - X_j)$.

In addition, when nodes approach the root of the tree and s_{kj} of (11.70) becomes either zero or negative, we need rules to identify the root as follows.

(1) $s_{kj} < 0$: X_j is a root.

(2) $s_{kj} = 0$: Consider the set $\pi(X_j) = \{X_k \mid X_k \in \Gamma(X_j), s_{kj} = 0\}$. Eliminate from $\pi(X_j)$ any X_k, from which there exists a directed path to X_j. If the resulting $\pi(X_j)$ is empty, X_j is a root. Otherwise, the predecessor of X_j is X_t which satisfies

$$\|X_t - X_j\| = \min_{X_k \in \pi(X_j)} \|X_k - X_j\| . \tag{11.73}$$

The similar result may be obtained without computing local means. The density values of (11.69) can be estimated by using any nonparametric technique such as the Parzen or *kNN* approach as discussed in Chapter 6. For example, if the Parzen approach with a uniform kernel function is used, $\hat{p}(X_j) = \hat{\tau}(X_j)/Nv$, where $\hat{\tau}(X_j)$ is the number of samples in $\Gamma(X_j)$, N is the total number of samples, and v is the volume of $\Gamma(X_j)$. Since N and v are independent of j, we may ignore them and replace $\hat{p}(X_j)$ by $\hat{\tau}(X_j)$. For the *kNN* approach, $\hat{p}(X_j) = (k-1)/N\hat{v}(X_j)$, where k is a preset number and $\hat{v}(X_j)$ must be measured. Since k and N are independent of j in this case, $\hat{p}(X_j)$ is replaced by $1/\hat{v}(X_j)$ in (11.69). Thus, using $\hat{p}(\cdot)$ in the place of $p(\cdot)$ in (11.69), (11.70) determines the predecessor of each sample.

The graph theoretic approach has a number of advantages. It is a non-iterative process, and does not require an initial class assignment. Also, the number of clusters needs not be preassigned. After the predecessor of each sample is found, a computer keeps track of the connections of samples to identify the number of isolated trees automatically.

In the graph theoretic approach, a crucial parameter is the size of the local region $\Gamma(X)$. A density function is not a smooth function with a few peaks, but a noisy function with many local peaks and valleys. With a small $\Gamma(X)$, the algorithm tends to pick up many clusters separated by the local valleys due to noise. On the other hand, if $\Gamma(X)$ is too large, all peaks and valleys are smoothed out and the algorithm produces only one cluster. In order to find a proper size for $\Gamma(X)$, it is suggested to run the algorithm for various sizes of $\Gamma(X)$, and observe the resulting number of clusters. Normally, as the size of $\Gamma(X)$ is changed from a small value to a large one, the number of clusters starts from a large value, drops down and stays at a certain level for a while, and then starts to drop again. The plateau at the middle is a reasonable and stable operating range, from which we can determine the size of $\Gamma(X)$ and the number of clusters.

Iterative valley-seeking: A nonparametric version of the nearest mean reclassification algorithm can be developed by defining a *nonparametric within-class scatter matrix* as

$$\mathcal{S}_w = \sum_{i=1}^{L} P_i \frac{1}{N_i} \sum_{j=1}^{N_i} (X_j^{(i)} - \mathcal{M}_i(X_j^{(i)}))(X_j^{(i)} - \mathcal{M}_i(X_j^{(i)}))^T \,, \tag{11.74}$$

where $\mathcal{M}_i(X_j^{(i)})$ is the sample mean of kNN's to $X_j^{(i)}$ from ω_i as

$$\mathcal{M}_i(X_j^{(i)}) = \frac{1}{k} \sum_{\ell=1}^{k} X_{\ell NN}^{(i)} \,. \tag{11.75}$$

We will call $\mathcal{M}_i(X_j^{(i)})$ the *local* ω_i*-mean* of $X_j^{(i)}$. This is the kNN version of the local mean. On the other hand, the local ω_i-mean for the Parzen approach is the sample mean of ω_i-samples in the local region $\Gamma(X_j)$ with a fixed radius. Comparing (11.74) with (10.99) and (10.100), we note that the weighting coefficients of (10.99) and (10.100) are not included in (11.74). Since w_ℓ requires knowledge of the true class assignment of the samples, their use is deemed inappropriate for clustering.

The criterion for class separability is set as $J = \text{tr}(S_m^{-1}\mathcal{S}_w)$ just as $J = \text{tr}(S_m^{-1}S_w)$ is used for the parametric counterpart. When k approaches N_i, the local ω_i-mean becomes the global ω_i-mean M_i, and consequently (11.74) becomes the parametric within-class scatter matrix S_w. Thus, the nearest mean reclassification algorithm is a special case of the optimization of $J = \text{tr}(S_m^{-1}\mathcal{S}_w)$. On the other hand, when $k << N_i$, we can develop the nonparametric reclassification algorithm by repeating the derivation of (11.13) with $\mathcal{M}_i(X_j)$ this time instead of M_i then, resulting in

$$\|X_j - \mathcal{M}_t(X_j)\| = \min_i \|X_j - \mathcal{M}_i(X_j)\| \quad \rightarrow \quad X_j \in \omega_t \,. \tag{11.76}$$

Note that (11.76) is applied only after the data is whitened with respect to S_m. This procedure may be called the *nearest local-mean reclassification* algorithm. In this algorithm, the local ω_i-means must be updated each time the class assignment is changed.

Another possible definition of the nonparametric within-class scatter matrix is

$$\mathcal{S}_w = \sum_{i=1}^{L} P_i \frac{1}{N_i} \sum_{j=1}^{N_i} (X_j^{(i)} - X_{kNN}^{(i)})(X_j - X_{kNN}^{(i)})^T \,, \tag{11.77}$$

where $X_{kNN}^{(i)}$ is the kth NN of $X_j^{(i)}$ from ω_i. This time, we use the kth NN itself instead of the sample mean of the *kNN*'s. Then, by a derivation similar to before, $X_j^{(i)}$ is reclassified to ω_t by

$$\|X_j - X_{kNN}^{(t)}\| = \min_{\ell} \|X_j - X_{kNN}^{(\ell)}\| \quad \rightarrow \quad X_j \in \omega_t \tag{11.78}$$

after whitening the data with respect to S_m.

Under the current class assignment, the density function of ω_ℓ at X_j can be estimated by using the *kNN* approach as

$$\hat{P}_\ell \hat{p}_\ell(X_j) = \frac{N_\ell}{N} \frac{k-1}{N_\ell \hat{v}_\ell(X_j)} = \frac{k-1}{Nc\|X_j - X_{kNN}^{(\ell)}\|^n} , \tag{11.79}$$

where c is a constant relating the radius to the volume. Selecting the smallest $\|X_j - X_{kNN}^{(\ell)}\|$ means selecting the largest $\hat{P}_\ell \hat{p}_\ell(X_j)$. Therefore, the reclassification algorithm of (11.78) suggests that we evaluate $\hat{P}_\ell \hat{p}_\ell(X_j)$ by (11.79) at each X_j, and classify X_j to the class which has the largest $\hat{P}_\ell \hat{p}_\ell(X_j)$.

When the Parzen approach with a uniform kernel function is used, $\hat{P}_\ell \hat{p}_\ell(X_j)$ is estimated by

$$\hat{P}_\ell \hat{p}_\ell(X_j) = \frac{N_\ell}{N} \frac{k_\ell(X_j)}{N_\ell v} = \frac{k_\ell(X_j)}{Nv} , \tag{11.80}$$

where v is a fixed volume of the local region around X_j, and $k_\ell(X_j)$ is the number of ω_ℓ-samples in the local region. Then, (11.78) is converted to

$$k_t(X_j) = \max_{\ell} k_\ell(X_j) \quad \rightarrow \quad X_j \in \omega_t \ . \tag{11.81}$$

The formulas of (11.78) and (11.81) have a computational advantage over the formula of (11.76). When (11.76) is used, we need to recompute the local means at each iteration. This is not required for (11.78) and (11.81).

When (11.81) is used, we set the local region around each sample with a fixed volume v, and tabulate samples in the local regions with their current class assignments, as shown in Fig. 11-14. Then, finding the class with the highest population, each sample is reclassified to that class. For example, in Fig. 11-14, X_1 is reclassified to ω_2 because the local region of X_1 contains one ω_1-sample and two ω_2-samples. After all samples are reclassified, the class labels of samples in the table are revised, and the same operation is repeated. In this iteration, only class labels are processed and no mean vector computation is involved. The same is true for (11.78). In the operation of (11.78),

		1st NN		2nd NN		3rd NN		4th NN		
	ω		ω		ω		ω		ω	
X_1	1	X_5	2	X_8	2	X_{13}	1			$X_1 \rightarrow \omega_2$
X_2	3	X_6	1	X_{21}	1					$X_2 \rightarrow \omega_1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$					$\vdots$
X_{1000}	5	X_{35}	5	X_{15}	5	X_{210}	6	X_{361}	5	$X_{1000} \rightarrow \omega_5$

Fig. 11-14 Iterative class assignment.

after tabulating neighbors of each sample, the sample is classified to ω_i when the kth NN from ω_i appears first in the sequence of $1NN$, $2NN$, ... For example, in Fig. 11-14 with $k = 2$, X_1 has a sequence of neighbors as $\omega_2, \omega_2, \omega_1, \ldots$, and the second NN from ω_2 appears first in the sequence. Accordingly, X_1 is reclassified to ω_2. Again, no mean computation is involved in each iteration.

Because of the above computational advantage, let us use (11.81) as the updating scheme of class assignment, and study the properties of the valley-seeking algorithm. The algorithm can be stated as follows [14-15].

(1) Whiten the data with respect to S_m.

(2) Assign the number of clusters, L. Choose an initial classification, $\Omega(0)$.

(3) Set a local spherical region around each sample, $\Gamma(X_j)$, with a fixed radius, and list samples in $\Gamma(X_j)$ with the current class assignment, as in Fig. 11-14.

(4) Reclassify X_j according to the majority of classes among all neighboring samples in $\Gamma(X_j)$.

(5) If any change in class assignment occurs, revise the class labels of all neighbors in the table and go to Step (4). Otherwise stop.

In order to understand how this procedure works, let us study the one-dimensional example of Fig. 11-15. Suppose that, at the ℓth iteration, samples are divided into 7 clusters as shown. A sample X_1 is not reclassified from ω_2 because all neighboring samples of X_1 in $\Gamma(X_1)$ belong to ω_2 currently. On the other hand, X_2 on the boundary between ω_5 and ω_6 is most likely reclassified to ω_6, because the number of neighbors from ω_6 tends to be larger than the number of ω_5-neighbors. This is due to the fact that the density function on the ω_6-side is higher than the one on the ω_5-side. Reclassifying X_2