EXHIBIT 5
PART

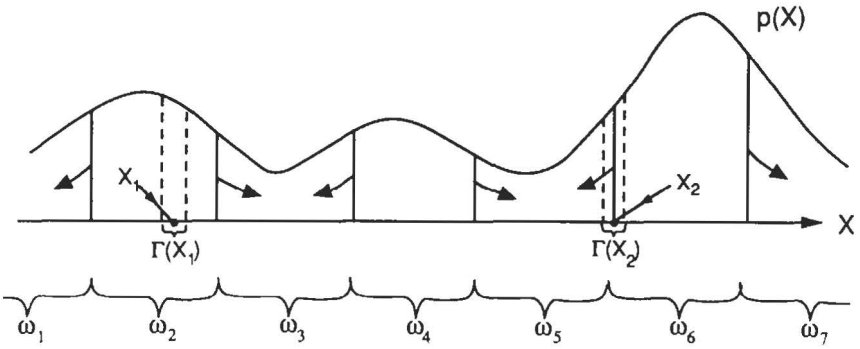

Fig. 11-15 An example of the valley-seeking clustering.

into ω_6 results in a shift of the boundary toward the ω_5-side, as the arrow indicates. This is equivalent to saying that the direction of the density gradient is estimated by using the numbers of samples on both sides of the boundary, and the boundary is moved toward the lower density side. Applying this process to all boundaries repeatedly, the leftmost boundary of Fig. 11-15 moves out to $-\infty$, leaving the ω_1-cluster empty, the second and third leftmost boundaries merge to one, making the ω_3-cluster empty, and so on. As a result, at the end of iteration, only ω_2, ω_4, and ω_6 keep many samples and the others become empty. The procedure works in the same way even in a high-dimensional space.

A number of comments can be made about this iterative valley-seeking procedure. The procedure is nonparametric, and divides samples according to the valley of a density function. The density gradient is estimated, but in a crude way. The number of clusters must be preassigned, but we can always assign a larger number of clusters than we actually expect to have. Many of initial clusters could become empty, and only true clusters separated by the valleys will keep samples. As far as computation time is concerned, it takes a lot of computer time to find neighbors for each sample and form the table of Fig. 11-14. However, this operation is common for all nonparametric clustering procedures, including the graph theoretic approach. The iterative process of this algorithm revises only the class assignment according to the majority of classes in $\Gamma(X)$. This operation does not take much computation time.

The volume of $\Gamma(X)$ affects the performance of this algorithm, just as it did in the graph theoretic approach. The optimum volume should be determined experimentally, as was done for the graph theoretic approach.

Experiment 4: Seventy five samples per class were generated according to

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = 20 \begin{bmatrix} \cos\boldsymbol{\theta} \\ \sin\boldsymbol{\theta} \end{bmatrix} + \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} + \begin{bmatrix} \mathbf{n}_1 \\ \mathbf{n}_2 \end{bmatrix}, \tag{11.82}$$

where $\mathbf{n}_1$ and $\mathbf{n}_2$ are independent and normally distributed with zero mean and unit variance for both ω_1 and ω_2, $[m_1\ m_2]$ is [0 0] for ω_1 and [0 −20] for ω_2, and $\boldsymbol{\theta}$ is normally distributed with $E\{\boldsymbol{\theta}|\omega_1\} = \pi$, $E\{\boldsymbol{\theta}|\omega_2\} = 0$, and $\text{Var}\{\boldsymbol{\theta}|\omega_1\} = \text{Var}\{\boldsymbol{\theta}|\omega_2\} = (\pi/4)^2$. After the data was whitened with respect to the mixture covariance matrix, both graph theoretic and iterative valley-seeking algorithms were applied, resulting in the same clustering result, as shown in Fig. 11-16 [13-14].

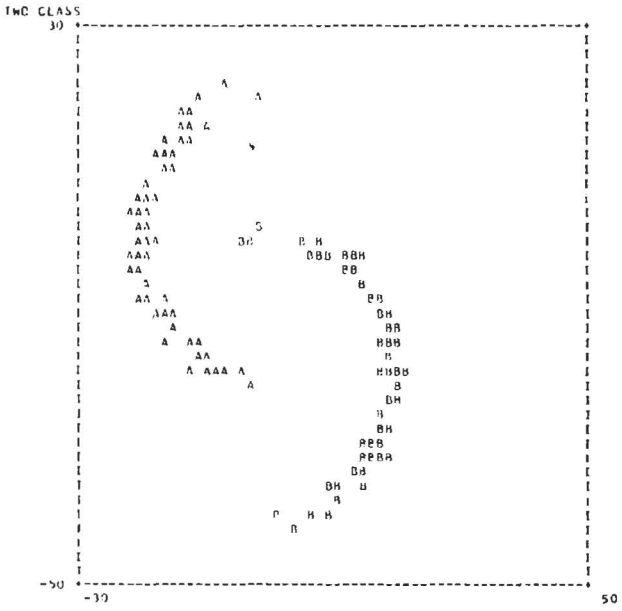

Fig. 11-16 Classification of a two-class example.

Table 11-3 shows the selected radius for $\Gamma(X)$, and the number of iterations required to reach the final clusters in the iterative valley-seeking algorithm [14].

Experiment 5: Fifty samples per class were generated from three classes. Two of them were the same ones as Experiment 4 except

TABLE 11-3

PERFORMANCE OF THE VALLEY-SEEKING ALGORITHM

Experiment	Number of samples	Number of clusters	Radius of $\Gamma(X)$	Number of iterations
4	150	2	1.0	8
5	150	3	0.75	10

$[m_1\ m_2] = [20\ 0]$ for ω_2 instead of $[0\ -20]$. The third distribution was normal with the mean and covariance matrix

$$M_3 = \begin{bmatrix} 10 \\ 0 \end{bmatrix} \quad \text{and} \quad \Sigma_3 = \begin{bmatrix} 16 & 0 \\ 0 & 1 \end{bmatrix}. \tag{11.83}$$

Again, after the data was whitened with respect to the mixture covariance matrix, both the graph theoretic and iterative valley-seeking algorithms produced the same clustering result, as shown in Fig. 11-17 [13-14]. The

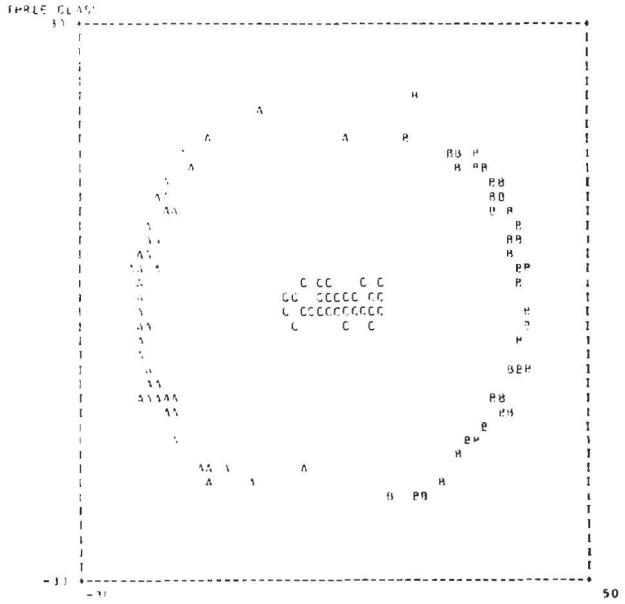

Fig. 11-17 Classification of a three-class example.

radius of $\Gamma(X)$ and the number of iterations in the latter algorithm are listed in Table 11-3 [14].

General comments: We have discussed both parametric and nonparametric clustering techniques. The question here is which technique is better under what conditions. Generally speaking, nonparametric techniques do not require the knowledge of the number of clusters beforehand. This is certainly true for the graph theoretic approach. Even for the iterative valley-seeking procedure, we can preassign a larger number of clusters than what is actually needed, and let the procedure select automatically the necessary number of clusters. Therefore, if we do not have any a priori knowledge about the data structure, it is most natural to adopt a nonparametric clustering technique and find out how the data is naturally divided into clusters.

However, nonparametric procedures are in general very sensitive to the control parameters, especially the size of the local region. Therefore, it is necessary to run experiments for a wide range of sizes, and the results must be carefully examined. Also, nonparametric procedures are normally very computationally intensive.

Furthermore, nonparametric clustering techniques have two fundamental flaws described as follows.

(1) We cannot divide a distribution into a number of clusters unless the valley actually exists. For example, the ω_1-distribution of Fig. 11-9 could be obtained as the result of the valley-seeking procedure, but the distribution cannot be divided into 2 or 3 clusters by any nonparametric method even if it is desirable to do so. When a distribution is wrapped as the ω_1-distribution of Fig. 11-9, it is sometimes prefered to decompose the distribution into several normal distributions for further analysis of data structure or designing a classifier.

On the other hand, the parametric procedures do not depend on the natural boundary of clusters, but depend only on the criterion. With a preassigned number of clusters, the procedures seek the boundary to optimize the criterion value. Therefore, we can divide the ω_1-distribution of Fig. 11-9 into 2, 3, or 4 clusters as we like. After examining the clustering results for various numbers of clusters, we may decide which number of clusters is most appropriate for a particular application. Previously, we stated that it is a disadvantage

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.